2014

Multi-Sensor Mobile Robot Localization For Diverse Environments

Joydeep Biswas
University of Massachusetts Amherst

Manuela M. Veloso
Carnegie Mellon University

Multi-Sensor Mobile Robot Localization For Diverse Environments

Joydeep Biswas and Manuela Veloso

School Of Computer Science, Carnegie Mellon University,
Pittsburgh, Pennsylvania 15213, USA
{joydeepb,veloso}@cs.cmu.edu

Abstract. Mobile robot localization with different sensors and algorithms is a widely studied problem, and there have been many approaches proposed, with considerable degrees of success. However, every sensor and algorithm has limitations, due to which we believe no single localization algorithm can be "perfect," or universally applicable to all situations.
Laser rangefinders are commonly used for localization, and state-of-the-art algorithms are capable of achieving sub-centimeter accuracy in environments with features observable by laser rangefinders. Unfortunately, in large scale environments, there are bound to be areas devoid of features visible by a laser rangefinder, like open atria or corridors with glass walls. In such situations, the error in localization estimates using laser rangefinders could grow in an unbounded manner. Localization algorithms that use depth cameras, like the Microsoft Kinect sensor, have similar characteristics. WiFi signal strength based algorithms, on the other hand, are applicable anywhere there is dense WiFi coverage, and have bounded errors. Although the minimum error of WiFi based localization may be greater than that of laser rangefinder or depth camera based localization, the maximum error of WiFi based localization is bounded and less than that of the other algorithms.
Hence, in our work, we analyze the strengths of localization using all three sensors - using a laser rangefinder, a depth camera, and using WiFi. We identify sensors that are most accurate at localization for different locations on the map. The mobile robot could then, for example, rely on WiFi localization more in open areas or areas with glass walls, and laser rangefinder and depth camera based localization in corridor and office environments.

Keywords: Localization, Mobile Robots, Sensor Fusion

1 Introduction And Related Work

Localization is an ability crucial to the successful deployment of any autonomous mobile robot, and has been a subject of continuous research. As technologies have evolved, we have seen a parallel evolution of algorithms to use new sensors, from SONAR, to infrared rangefinders, to vision, laser rangefinders, wireless

beacons, vision, and most recently depth cameras. The wide-scale deployment of autonomous robots has for a long time been hampered by the limitations of the sensors and the algorithms used for robot autonomy. This has led to work on robots like Minerva [1] that used a "coastal planner" to avoid navigation paths with poor information content perceivable to the robot. Exploration and navigation approaches that account for perceptual limitations of robots [2] have been studied as well. These approaches acknowledge the limitations in perception, and seek to avoid areas where perception is poor.

A number of robotics projects have been launched over the years in pursuit of the goal of long-term autonomy for mobile service robots. Shakey the robot [3] was the first robot to actually perform tasks in human environments by decomposing tasks into sequences of actions. Rhino [4], a robot contender at the 1994 AAAI Robot Competition and Exhibition, used SONAR readings to build an occupancy grid map [5], and localized by matching its observations to expected wall orientations. Minerva [1] served as a tour guide in a Smithsonian museum. It used laser scans and camera images along with odometry to construct two maps, the first being an occupancy grid map, the second a textured map of the ceiling. For localization, it explicitly split up its observations into those corresponding to the fixed map and those estimated to have been caused by dynamic obstacles. Xavier [6] was a robot deployed in an office building to perform tasks requested by users over the web. Using observations made by SONAR, a laser striper and odometry, it relied on a Partially Observable Markov Decision Process (POMDP) to reason about the possible locations of the robot, and to reason about the actions to choose accordingly. A number of robots, Chips, Sweetlips and Joe Historybot [7] were deployed as museum tour guides in the Carnegie Museum of Natural History, Pittsburgh. Artificial fiducial markers were placed in the environment to provide accurate location feedback for the robots. The PR2 robot at Willow Garage [8] has been demonstrated over a number of milestones, where the robot had to navigate over 42 km and perform a number of manipulation tasks. The PR2 used laser scan data along with Inertial Measurement Unit (IMU) readings and odometry to build an occupancy grid map using GMapping [9], and then localized itself on the map using KLD-sampling [10]. The Collaborative Robots (CoBots) project [11] seeks to explore the research and engineering challenges involved in deploying teams of autonomous robots in human environments. The CoBots autonomously perform tasks on multiple floors of our office building, including escorting visitors, giving tours and transporting objects. The annual RoboCup@Home competition [12] aims to promote research in mobile service robots deployed in typical human home environments.

Despite the localized successes of these robotic projects, we still are short of the goal of having indefinitely deployed robots without human intervention. To have a robot be universally accessible, it needs to be able to overcome individual sensor limitations. This has led to the development of a number of algorithms for mobile robot localization by "fusing" multiple sensor feeds. There are a number of approaches to sensor fusion [13] for robot localization, including merging multiple sensor feeds at the lowest level before being processed homogeneously,

and hierarchical approaches to fuse state estimates derived independently from multiple sensors.

Our approach, instead of merging sensor feeds into a single algorithm, is to select one out of a set of localization algorithms, each of which independently processes different sensory feeds. The selection is driven by the results of evaluating the performance of the individual algorithms over a variety of indoor scenarios that the mobile robot will encounter over the course of its deployment. We use three localization algorithms, each using a different sensor: a laser rangefinder based localization algorithm that uses a Corrective Gradient Refinement [14] particle filter, a depth camera based localization algorithm [15], and a WiFi localization algorithm that is a cross between graph-based WiFi localization [16] and WiFi localization using Gaussian Processes [17]. Algorithm selection [18] is a general problem that has been applied to problems like SAT solving [19], and spawned the field of "meta-learning" [20]. Here we apply the problem of algorithm selection to the problem of mobile robot localization.

By running each localization algorithm independently, we can decide to pick one over the others for different scenarios, thus avoiding the additional complexity of fusing dissimilar sensors for a single algorithm. To determine which algorithm is most robust in which scenario, we collected sensor data over the entire map, and evaluated each of the algorithms (with respect to ground truth) over the map and over multiple trials. This data is used to determine which algorithm is most robust at every location of the map. Additionally, by comparing the performance of the algorithms while the robot navigates in corridors and while it navigates in open areas, we show that different algorithms are robust and accurate for different locations on the map.

2 Multi-Sensor Localization

We use three sensors for localization - a laser rangefinder, a depth camera, and the WiFi received signal strength indicator (RSSI). Each sensor, along with robot odometry, is processed independently of the others. For the laser rangefinder, we use a Corrective Gradient Refinement (CGR) [14] based particle filter. The depth camera observations are processed by the Fast Sampling Plane Filtering [15] algorithm to detect planes, which are then matched to walls in the map to localize the robot using a CGR particle filter. The WiFi RSSI values observed by the robot are used for localization using a Gaussian Processes-Learnt WiFi Graph. This WiFi localization algorithm combines the strengths of graph-based WiFi localization [16] with Gaussian Processes for RSSI based location estimation [17]. We first review each of these sensor-specific localization algorithms.

2.1 Corrective Gradient Refinement

For localization using a laser rangefinder sensor, the belief of the robot's location is represented as a set of weighted samples or "particles", as in Monte Carlo

Localization (MCL)[21]: $Bel(x_t) = \left\{x_t^i, w_t^i\right\}_{i=1:m}$. The Corrective Gradient Refinement (CGR) algorithm iteratively updates the past belief $Bel(x_{t-1})$ using observation y_t and control input u_{t-1} as follows:

1. Samples of the belief $Bel(x_{t-1})$ are evolved through the motion model, $p(x_t|x_{t-1}, u_{t-1})$ to generate a first stage proposal distribution q^0.
2. Samples of q^0 are "refined" in r iterations (which produce intermediate distributions $q^i, i \in [1, r-i]$) using the gradients $\frac{\delta}{\delta x}p(y_t|x)$ of the observation model $p(y_t|x)$.
3. Samples of the last generation proposal distribution q^r and the first stage proposal distribution q^0 are sampled using an acceptance test to generate the final proposal distribution q.
4. Samples x_t^i of the final proposal distribution q are weighted by corresponding importance weights w_t^i, and resampled with replacement to generate $Bel(x_t)$.

Thus, given the motion model $p(x_t|x_{t-1}, u_{t-1})$, the observation model $p(y_t|x)$, the gradients of the observation model $\frac{\delta}{\delta x}p(y_t|x)$ and the past belief $Bel(x_{t-1})$, the CGR algorithm computes the latest belief $Bel(x_t)$.

2.2 Depth Camera Based Localization

Depth cameras provide, for every pixel, color and depth values. This depth information, along with the camera intrinsics (horizontal field of view f_h, vertical field of view f_v, image width w and height h in pixels) can be used to reconstruct a 3D point cloud. Let the depth image of size $w \times h$ pixels provided by the camera be I, where $I(i, j)$ is the depth of a pixel at location $d = (i, j)$. The corresponding 3D point $p = (p_x, p_y, p_z)$ is reconstructed using the depth value $I(d)$ as $p_x = I(d)\left(\frac{j}{w-1} - 0.5\right)\tan\left(\frac{f_h}{2}\right)$, $p_y = I(d)\left(\frac{i}{h-1} - 0.5\right)\tan\left(\frac{f_v}{2}\right)$, $p_z = I(d)$.

With limited computational resources, most algorithms (*e.g.* localization, mapping *etc.*) cannot process the full 3D point cloud at full camera frame rates in real time with limited computational resources on a mobile robot. The naïve solution would therefore be to sub-sample the 3D point cloud for example, by dropping (say) one out of N points, or sampling randomly. Although this reduces the number of 3D points being processed by the algorithms, it ends up discarding information about the scene. An alternative solution is to convert the 3D point cloud into a more compact, feature - based representation, like planes in 3D. However, computing optimal planes to fit the point cloud for every observed 3D point would be extremely CPU-intensive and sensitive to occlusions by obstacles that exist in real scenes. The Fast Sampling Plane Filtering (FSPF) algorithm combines both ideas: it samples random neighborhoods in the depth image, and in each neighborhood, it performs a RANSAC based plane fitting on the 3D points. Thus, it reduces the volume of the 3D point cloud, it extracts geometric features in the form of planes in 3D, and it is robust to outliers since it uses RANSAC within the neighborhood.

Algorithm 1 Fast Sampling Plane Filtering

1: **procedure** PLANEFILTERING(I)
2: $P \leftarrow \{\}$ ▷ Plane filtered points
3: $R \leftarrow \{\}$ ▷ Normals to planes
4: $O \leftarrow \{\}$ ▷ Outlier points
5: $n \leftarrow 0$ ▷ Number of plane filtered points
6: $k \leftarrow 0$ ▷ Number of neighborhoods sampled
7: **while** $n < n_{max} \wedge k < k_{max}$ **do**
8: $k \leftarrow k + 1$
9: $d_0 \leftarrow (\text{rand}(0, h-1), \text{rand}(0, w-1))$
10: $d_1 \leftarrow d_0 + (\text{rand}(-\eta, \eta), \text{rand}(-\eta, \eta))$
11: $d_2 \leftarrow d_0 + (\text{rand}(-\eta, \eta), \text{rand}(-\eta, \eta))$
12: Reconstruct p_0, p_1, p_2 from d_0,d_1,d_2
13: $r = \frac{(p_1 - p_0) \times (p_2 - p_0)}{||(p_1 - p_0) \times (p_2 - p_0)||}$ ▷ Compute plane normal
14: $\bar{z} = \frac{p_{0z} + p_{1z} + p_{2z}}{3}$
15: $w' = w\frac{S}{\bar{z}}\tan(f_h)$
16: $h' = h\frac{S}{\bar{z}}\tan(f_v)$
17: $[\text{numInliers}, \hat{P}, \hat{R}] \leftarrow \text{RANSAC}(d_0, w', h', l, \epsilon)$
18: **if** numInliers $> \alpha_{in} l$ **then**
19: Add $\hat{P}$ to P
20: Add $\hat{R}$ to R
21: numPoints $\leftarrow$ numPoints + numInliers
22: **else**
23: Add $\hat{P}$ to O
24: **end if**
25: **end while**
26: **return** P, R, O
27: **end procedure**

Fast Sampling Plane Filtering (FSPF) [15] takes the depth image I as its input, and creates a list P of n 3D points, a list R of corresponding plane normals, and a list O of outlier points that do not correspond to any planes. Algorithm1 outlines the plane filtering procedure. It uses the helper subroutine $[\text{numInliers}, \hat{P}, \hat{R}] \leftarrow \text{RANSAC}(d_0, w', h', l, \epsilon)$, which performs the classical RANSAC algorithm over the window of size $w' \times h'$ around location d_0 in the depth image, and returns inlier points and normals $\hat{P}$ and $\hat{R}$ respectively, as well as the number of inlier points found. The configuration parameters required by FSPF are listed in Table1.

FSPF proceeds by first sampling three locations d_0,d_1,d_2 from the depth image (lines 9-11). The first location d_0 is selected randomly from anywhere in the image, and then d_1 and d_2 are selected from a neighborhood of size η around d_0. The 3D coordinates for the corresponding points p_0, p_1, p_2 are then computed (line 12). A search window of width w' and height h' is computed based on the mean depth (z-coordinate) of the points p_0, p_1, p_2 (lines 14-16) and the minimum expected size S of the planes in the world. Local RANSAC is then performed in the search window. If more than $\alpha_{in} l$ inlier points are produced

Parameter	Value	Description
n_{max}	2000	Maximum total number of filtered points
k_{max}	20000	Maximum number of neighborhoods to sample
l	80	Number of local samples
η	60	Neighborhood for global samples (in pixels)
S	$0.5m$	Plane size in world space for local samples
ϵ	$0.02m$	Maximum plane offset error for inliers
α_{in}	0.8	Minimum inlier fraction to accept local sample

Table 1: Configuration parameters for FSPF

as a result of running RANSAC in the search window, then all the inlier points are added to the list P, and the associated normals (computed using a least-squares fit on the RANSAC inlier points) to the list R. This algorithm is run a maximum of m_{max} times to generate a list of maximum n_{max} 3D points and their corresponding plane normals.

With the list P and R, non-vertical planes are ignored, and the remaining planes are matched to a 2D vector map. These remaining matched points are then used for localization using CGR.

2.3 WiFi Based Localization

WiFi localization using a graph based WiFi map [16] provides fast, accurate robot localization when constrained to a graph. In this approach, the mean and standard deviations of WiFi RSSI observations are approximated by linear interpolation on a graph. This leads to a computationally efficient observation likelihood function, with the computation time of each observation likelihood being independent of the number of training instances. The disadvantage is that to construct the WiFi graph map, the robot needs to be accurately placed at every vertex location of the graph to collect WiFi RSSI training examples.

RSSI based localization using Gaussian Processes [17], on the other hand, does not require training observations from specific locations. Given an arbitrary location x_* and the covariance vector k_* between the training locations and x_*, the expected mean μ_{x_*} and variance is given by,

$$\mu_{x_*} = k_*^T (K + \sigma_n^2 I)^{-1} y \tag{1}$$

$$\sigma_{x_*}^2 = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_*. \tag{2}$$

Here, σ_n^2 is the Gaussian observation noise, y the vector of training RSSI observations, and $k(\cdot, \cdot)$ the kernel function used for the Gaussian Process (most commonly a squared exponential). The drawback with this approach is that the computational complexity for Eq.1-2 grows quadratically with the number of training samples, which is more than $100,000$ samples per access point for our single floor map. We introduce an approach that combines the advantages of both algorithms while overcoming each of their limitations. In our approach, training

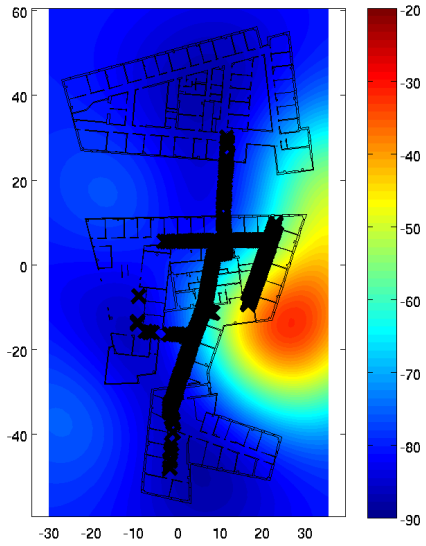
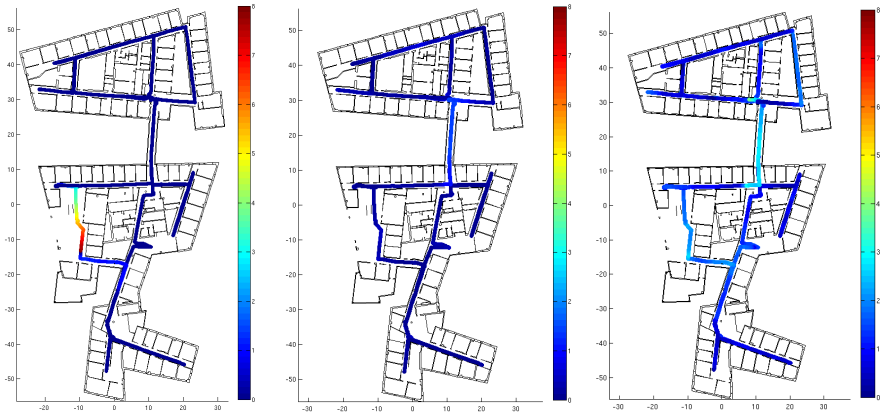


Fig. 1: Gaussian Process-learnt WiFi graph for a single access point. The mean RSSI values are color-coded, varying from $-90dBm$ to $-20dBm$. The locations where the robot observed signals from the access point are marked with crosses.

examples are first collected across the map while driving the robot around (not from any specific locations). Next, using Gaussian Processes, the WiFi mean and variance for the nodes of a WiFi Graph (with regularly spaced sample locations on the map) are learnt offline. Once the WiFi graph is learnt, WiFi mean and variance at locations during run time are estimated by bicubic interpolation across nodes of the graph. Figure 1 shows a Gaussian Process-learnt WiFi graph for a single access point on the floor.

3 Comparing Localization Algorithms

To evaluate the robustness of the different localization algorithms over a variety of environments, we collected laser scanner, depth camera, and WiFi sensor data while our mobile robot autonomously navigated around the map. The navigation trajectory covered each corridor multiple times, and also covered the open areas of the map. The sensor data thus collected was then processed offline for 100 times by each of the algorithms (since the algorithms are stochastic in nature), and the mean localization error across trials evaluated for each sensor. Figure 2 shows the errors in localization when using the three different sensors. It is seen that in corridors, the laser rangefinder based localization algorithm is the most accurate, with mean errors of a few centimeters. The depth camera (Kinect)



(a) (b) (c)

Fig. 2: Errors in localization using (a) a laser rangefinder, (b) the Kinect, and (c) WiFi, while autonomously driving across the map. The error in localization for every true location of the robot is coded by color on the map. All units are in meters.

based localization has slightly higher errors of about ten centimeters while the robot travels along corridors. The WiFi localization algorithm consistently had errors of about a meter across the map.

We then ran additional trials in a large open area of the map, where the dimensions of the area exceed the maximum range of the laser rangefinder and depth image sensors. Figure 3 shows a photograph of this open area. In these trials, the robot started out from the same fixed location, and was manually driven around the open area. The robot was then driven back to the starting location, and the error in localization noted. Figure 4a shows part of such a trajectory as the robot was driven around the open area. This was done 20 times for each localization algorithm, and the combined length of all the trajectories was over 2Km, with approximately 0.7Km per algorithm. Figure 4b shows a histogram of the recorded errors in localization at the end of the driven trajectories for each localization algorithm. The following table tabulates the minimum, maximum, and median errors for the different algorithms:

	Laser Rangefinder	Kinect	WiFi
Min Error	0.01	0.17	0.20
Max Error	14.77	3.47	1.88
Median Error	1.83	1.08	0.76

The laser rangefinder has the smallest minimum error during trajectories that the algorithm tracked successfully, but also has the largest maximum error. The WiFi localization algorithm, however, was consistently able to track the robot's trajectory for every trial with a median error of less than a meter.

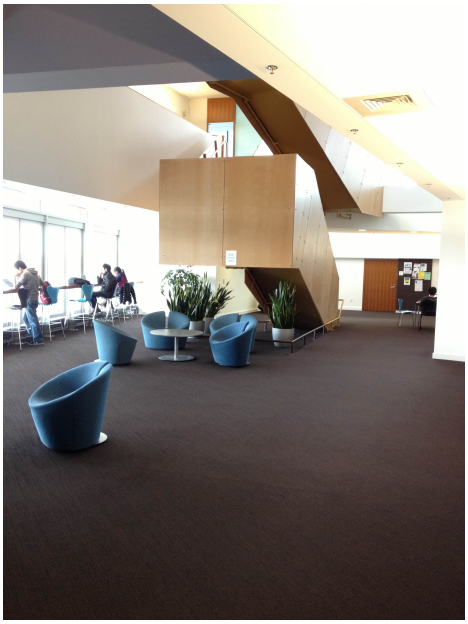


Fig. 3: The open area in which localization using different sensors were compared.

4 Choosing Between Sensors For Localization

A robot equipped with multiple sensors and running different localization algorithms for each sensor needs to decide when to rely on each of the algorithms. There are three broad approaches to choosing between sensors and algorithms for localization: computed uncertainty based selection, sensor mismatch based selection, and data-driven selection.

4.1 Computed Uncertainty Based Selection

Every localization algorithm maintains a measure of uncertainty of the robot's pose estimates. The uncertainty of a particle filter based localization algorithm is captured by the distribution of its particles, while in an Extended Kalman Filter (EKF) based localization algorithm the covariance matrix captures the uncertainty of the robot's pose. Given a measure of the uncertainty of localization using the different sensors, the robot could choose the algorithm (and hence the sensor) with the least uncertainty at any given point of time. However, it is not possible to directly compare the uncertainty measures from different algorithms because of the assumptions and simplifications specific to each algorithm. For example, an EKF based algorithm assumes a unimodal Gaussian distribution of the robot's pose, which is not directly comparable to the sample based multimodal distribution of poses of a particle filter.

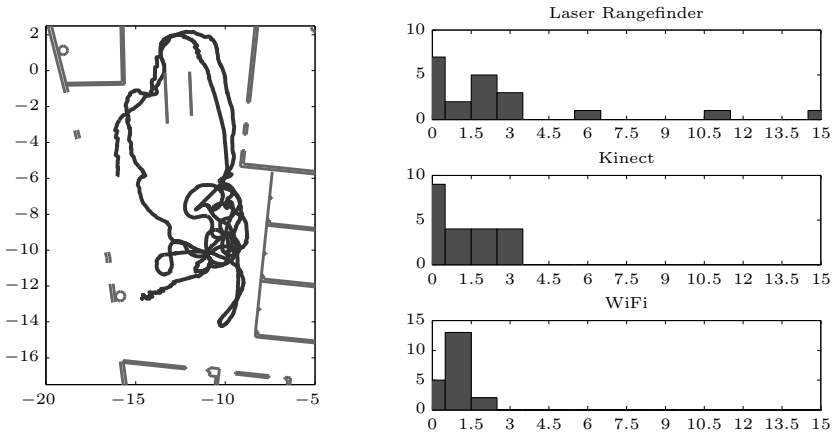


Fig. 4: Trials in the open area: (a) part of a test trajectory of the robot, and (b) histogram of errors in localization across trials for each algorithm.

4.2 Sensing Mismatch Based Selection

A more algorithm-agnostic approach to computing a measure of uncertainty for localization algorithms is to compute the mismatch between the observation likelihood function $P(l|s)$ and the belief $Bel(l)$. Sensor Resetting Localization (SRL) [22] evaluates this mismatch by sampling directly from the observation likelihood function $P(l|s)$, and adds the newly generated samples to the particle filter proportional to the mismatch. KLD Resampling [10] similarly evaluates the mismatch by sampling from the odometry model $P(l_i|l_{i-1}, u_i)$ and computing the overlap with the belief $Bel(l)$. With the estimates of sensor mismatch between each localization algorithm and their respective belief distributions, the robot could select the algorithm with the best overlap between the observation likelihood function and the belief. Such an approach would work well for choosing between different algorithms that use similar sensors, like depth cameras and laser rangefinders. However, algorithms using different types of sensors like WiFi signal strength measurements as compared to rangefinder based measurements, cannot be compared directly based on their sensor meismatch due to the very different forms of the observation likelihood functions for the different sensors.

4.3 Data - Driven Selection

If neither uncertainty based nor sensor mismatch based selection is feasible, a data-driven selection of the localization algorithms might be the only recourse. In order to select the best localization algorithm for a given location, the data-driven approach requires experimental evaluation of every localization algorithm

(as described in Section 3) over all areas in the environment. During deployments, the robot would then select (at each location) that algorithm which exhibited least variance during the experimental evaluation at the same location. Such an approach would work for any localization algorithm, and does impose any algorithmic limitations on the individual localization algorithms, unlike uncertainty or sensing mismatch based selection. However, this approach does require experimental evaluation of all localization algorithms in all the areas of the environment, which might not be feasible for large areas of deployment.

5 Conclusion And Future Work

In this work, we tested the robustness of localization using three different sensors over an entire map. We showed that in areas where there are features are visible to the laser rangefinder and Kinect, they are both more accurate than WiFi. However, in open areas, localization using a laser rangefinder or Kinect is susceptible to large errors. In such areas, the maximum error of WiFi is bounded, and its median error is lower than that using a laser rangefinder or a Kinect. This indicates that although no single sensor is universally superior for localization, by selecting different sensors for localization for different areas of the map, the localization error of the robot is bounded. Based on these results, we are now interested in learning how to predict a priori which sensors would be most robust for localization for different areas of the map.

References

1. Thrun, S., Bennewitz, M., Burgard, W., Cremers, A., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., et al.: Minerva: A second-generation museum tour-guide robot. In: Robotics and Automation, 1999 IEEE International Conference on. Volume 3.
2. Romero, L., Morales, E., Sucar, E.: An exploration and navigation approach for indoor mobile robots considering sensor's perceptual limitations. In: Robotics and Automation, 2001 IEEE International Conference on. Volume 3., IEEE (2001) 3092–3097
3. Nilsson, N.: Shakey the robot. Technical report, DTIC Document (1984)
4. Buhmann, J., Burgard, W., Cremers, A., Fox, D., Hofmann, T., Schneider, F., Strikos, J., Thrun, S.: The mobile robot rhino. AI Magazine **16**(2) (1995) 31
5. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. Computer **22**(6) (1989) 46–57
6. Koenig, S., Simmons, R.: Xavier: A robot navigation architecture based on partially observable markov decision process models. In: Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems. (1998) 91 – 122
7. Nourbakhsh, I., Kunz, C., Willeke, T.: The mobot museum robot installations: A five year experiment. In: Intelligent Robots and Systems, 2003 IEEE/RSJ International Conference on. Volume 4. 3636–3641
8. Oyama, A., Konolige, K., Cousins, S., Chitta, S., Conley, K., Bradski, G.: Come on in, our community is wide open for robotics research! In: The 27th Annual conference of the Robotics Society of Japan. Volume 9. (2009) 2009

9. Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. Robotics, IEEE Transactions on **23**(1) (2007) 34–46
10. Fox, D.: KLD-sampling: Adaptive particle filters and mobile robot localization. Advances in Neural Information Processing Systems (NIPS) (2001)
11. Rosenthal, S., Biswas, J., Veloso, M.: An effective personal mobile robot agent through symbiotic human-robot interaction. In: Autonomous Agents and Multiagent Systems, 9th International Conference on. (2010) 915–922
12. Van der Zant, T., Wisspeintner, T.: Robocup@ home: Creating and benchmarking tomorrows service robot applications. Robotic Soccer (2007) 521–528
13. Kam, M., Zhu, X., Kalata, P.: Sensor fusion for mobile robot navigation. Proceedings of the IEEE **85**(1) (1997) 108–119
14. Biswas, J., Coltin, B., Veloso, M.: Corrective gradient refinement for mobile robot localization. In: Intelligent Robots and Systems, 2011 IEEE/RSJ International Conference on, IEEE 73–78
15. Biswas, J., Veloso, M.: Depth camera based indoor mobile robot localization and navigation. In: Robotics and Automation, 2012 IEEE International Conference on. 1697–1702
16. Biswas, J., Veloso, M.: Wifi localization and navigation for autonomous indoor mobile robots. In: Robotics and Automation, 2010 IEEE International Conference on. 4379–4384
17. Ferris, B., Hähnel, D., Fox, D.: Gaussian processes for signal strength-based location estimation. In: In Proc. of Robotics Science and Systems, Citeseer (2006)
18. Rice, J.: The algorithm selection problem. Advances in Computers **15** (1976) 65–118
19. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Satzilla: portfolio-based algorithm selection for sat. Journal of Artificial Intelligence Research **32**(1) (2008) 565–606
20. Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys (CSUR) **41**(1) (2008) 1–25
21. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: Robotics and Automation, 1999 IEEE International Conference on. Volume 2., IEEE 1322–1328
22. Lenser, S., Veloso, M.: Sensor resetting localization for poorly modelled mobile robots. In: Robotics and Automation, 2000 IEEE International Conference on. 1225–1232