An Experimental Comparison of Localization Methods Continued

Jens-Steffen Gutmann
Digital Creatures Laboratory
Sony Corporation, Tokyo, Japan
Email: gutmann@ieee.org

Dieter Fox
Department of Computer Science & Engineering
University of Washington, Seattle, WA
Email: fox@cs.washington.edu

Abstract

Localization is one of the fundamental problems in mobile robot navigation. Past experiments show that in general grid-based Markov localization is more robust than Kalman filtering while the latter can be more accurate than the former. Recently new methods for localization employing particle filters became popular. In this paper we compare different localization methods using Kalman filtering, grid-based Markov localization, Monte Carlo Localization (MCL), and combinations thereof. We give experimental evidence that a combination of Markov localization and Kalman filtering as well as a variant of MCL outperform the other methods in terms of accuracy, robustness, and time needed for recovering from manual robot displacement, while requiring only few computational resources.

1 Introduction

Self-localization is the task of estimating the pose (position and orientation) of a mobile robot given a map of the environment and a history of sensor readings and executed actions. It is one of the fundamental problems in mobile robot navigation and many solutions have been presented in the past including approaches employing Kalman filtering [14, 15, 17, 18], grid-based Markov localization [4, 10], or Monte Carlo methods [9, 16, 20]. For an overview see [7, 11, 19].

By performing localization experiments with a mobile robot it has been ascertained that grid-based Markov localization is more robust than Kalman filtering while the latter – given good inputs – is more efficient and accurate than the former [13]. A combination of both approaches is likely to inherit the advantages of the underlying techniques. More recently, new localization methods employing particle filters have been presented [9]. Variants such as Sensor Resetting Localization [16] or Mixture MCL [20, 11] further improved the performance of this localization framework. However, it is an open question how the Monte Carlo approaches perform in comparison with previous methods such as Kalman filters, grid-based Markov localization, or combinations thereof.

This work is the continuation of the localization experiments reported in [13] with the latest methods available to the authors [12, 16, 20]. Using a similar setup as in the RoboCup Sony legged robot league, where a quadruped robot observes landmarks with a CMOS camera, we experimentally compare Kalman filtering (EKF) [2, 18], a combination of grid-based Markov localization and Kalman filtering (ML-EKF) [12], Sensor Resetting localization (SRL) [16], Mixture MCL (Mix-MCL) [20] and a previously unreported variant of SRL called adaptive MCL (A-MCL) with each other.

The contribution of our work is as follows:

1. Experimental evidence that ML-EKF and MCL methods outperform simple Kalman filters in terms of accuracy, robustness, and time for recovering from manual robot displacement.
2. Results which indicate that ML-EKF compares very well to MCL methods, outperforming some but not all of them.
3. Experiments showing that adaptive MCL partially outperforms sophisticated localization algorithms such as ML-EKF.
4. All methods perform equally well when using only a fraction of the available sensor data, if the sensor provides measurements at high frame rates.

The remainder of this paper is organized as follows. The next section provides a brief overview of the different localization methods compared in our experiments. Section 3 describes the experimental setup, followed by a detailed discussion of our results. Finally, Section5 provides a discussion of our findings.

2 Probabilistic Localization

In probabilistic terms, localization is the process of determining the probability of the robot being at pose l given sensor inputs s_n and executed actions a_n $(n = 1 \ldots N)$. By assuming independence between observations and between executed actions, the following update rules can be formulated in the Markov localization framework [10]:

$$p(l) \quad \leftarrow \quad \alpha\, p(s_n \mid l)\, p(l) \qquad (1)$$

$$p(l) \quad \leftarrow \quad \int p(l \mid a_n, l')\, p(l')\, dl' \qquad (2)$$

where (1) is performed on observations evaluating sensor model $p(s_n \mid l)$, which returns the likelihood of an observation for a given pose, multiplied by a normalizing factor α, and (2) is employed on robot motion evaluating motion model $p(l \mid a_n, l')$ which delivers the probability of the robot being at pose l given it was at l' and executed action a_n.

Depending on the application and the type of robot, different motion models can be considered. Throughout this paper we make use of 3 motion models sketched in Fig. 1 where (a) refers to a Gaussian model that is widely used in all kinds of robot systems, (b) is the *max distance* model that places equal probability to positions up to a certain range and is useful e.g. for legged robots that might get obstructed at obstacles, and (c) allows a robot to be manually displaced (kidnapped) to any position in the environment with a certain displacement probability.

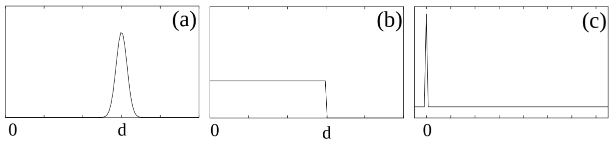


Fig. 1. *Motion models: (a) Gaussian, (b) max distance, (c) displacement model.*

Depending on the representation of $p(l)$ localization methods can be categorized as Kalman filters, grid-based Markov localization, or Monte Carlo methods.

2.1 Kalman Filter

Kalman filtering emerges when representing all densities by Gaussians: $p(l) \sim N(\hat{l}, \Sigma_l)$, $p(s_n|l) \sim N(\hat{s}_n, \Sigma_{s_n})$, and $p(l|a_n, l') \sim N(\hat{a}_n, \Sigma_{a_n})$. The update rules (1, 2) become:

$$\hat{l} \leftarrow \hat{l} + Wv \quad , \quad \Sigma_l \leftarrow \Sigma_l - W\Sigma_v W^T \qquad (3)$$

$$\hat{l} \leftarrow f(\hat{l}, \hat{a}_n) \quad , \quad \Sigma_l \leftarrow \nabla f \Sigma_l \nabla f^T + \Sigma_{a_n} \qquad (4)$$

where $W = \Sigma_l \nabla h^T \Sigma_v^{-1}$ is the filter gain, $v = \hat{s}_n - h(\hat{l})$ the innovation, $\Sigma_v = \nabla h \Sigma_l \nabla h^T + \Sigma_{s_n}$ its covariance, h and f are the measurement and process equations, and ∇h and ∇f their Jacobians. Note that the EKF does not necessarily assume Gaussian densities but is a linear estimation tool for any random variable given it can be adequately represented by the first and second moments of its density function [2].

Kalman filtering has been successfully applied for mobile robot localization in many systems [13, 14, 15, 17]. The inherent problem in this approach is that only one pose hypothesis can be represented making the method in general unable to globally localize the robot or to recover from total localization failures.

2.2 Grid-based Markov localization and ML-EKF

If $p(l)$ is represented by a piece-wise linear function, we obtain grid-based Markov localization [10]. Advantages of this method are its global search space and flexibility for different motion and sensor models. Depending on the dimension, resolution and size of the grid, the method might not be feasible for real-time applications without further optimizations.

Recently a novel approach combining Markov localization and Kalman filtering (ML-EKF) for localizing robots observing known landmarks has been presented. For details about the ML-EKF algorithm, see [12].

The basic idea is that Markov localization is used for global search of the robot position providing high robustness on sensor noise and fast recovering from manual robot displacement, whereas Kalman filtering is used locally for precisely computing the robot pose. The ML-EKF system consists of 3 modules depicted in Fig. 2.

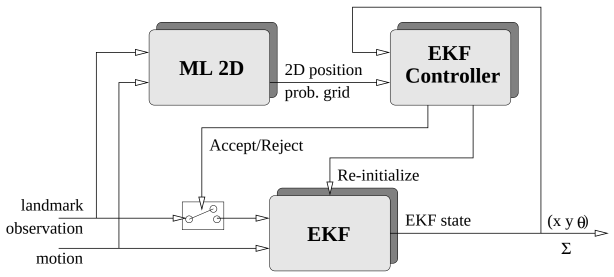


Fig. 2. *Markov-Kalman (ML-EKF) localization system*

A 2D Markov localization grid at coarse resolution represents possible (x, y) positions of the robot but does not contain information about orientation. Because of being 2D, on observations only the distance to landmarks are considered and on motion all directions are treated with equal probability. This setup allows for very fast updates as we will see in Section 4.

The heart of this localization system is the EKF controller that filters observations and reinitializes the EKF when necessary. All robot motions are integrated into the Markov grid and the EKF. The latter uses a Gaussian motion model (optimistic assumption, Fig. 1(a)) whereas the Markov grid employs a mixture of max distance and displacement motion model (pessimistic assumption, Fig. 1(b) and 1(c)).

Landmark observations are first integrated into the Markov grid. If the given observation is plausible based on the Markov state, it is also integrated into the EKF. The plausibility check examines $p(s_n|\tilde{l})$, where $\tilde{l}$ is the maximum likely cell in the grid. If this probability is smaller than a threshold t_{obs}, the observation is rejected for the EKF. After accepting and integrating an observation into the EKF, the distributions of Markov grid and EKF are compared using a χ^2 test. If the test exceeds a threshold t_{χ^2}, the Kalman filter is reinitialized with $\tilde{l}$ and the maximum likely orientation computed by projecting the last unfiltered observation to $\tilde{l}$.

The output of the ML-EKF system is the EKF state.

A limitation of this approach is that integration of observations into the 2D Markov grid must be feasible, i.e.

$$p(s_n \mid l_{xy}) = \int_0^{2\pi} p(s_n \mid l_x, l_y, l_\theta)\, dl_\theta \qquad (5)$$

can be computed efficiently. This is true for landmark based navigation but might not be the case for methods using dense sensor matching [10, 13, 20].

2.3 Monte Carlo Localization (MCL)

The key idea of Monte Carlo localization (MCL) is to represent $p(l)$ by sets of n weighted samples $\langle l_i, w_i \rangle$ [9]. Each l_i corresponds to a robot position, and the w_i are non-negative numerical factors called importance weights, which sum up to one. The prediction and correction update of the sample sets is achieved by a procedure often referred to as sequential importance sampling with resampling [6]. The basic algorithm takes as input a sample set S representing the current position estimate $p(l)$, a sensor measurement s_n, and action information a_n. Each sample representing the posterior distribution for $p(l)$ is generated according to the following three steps:

Resampling: Draw with replacement a random sample l' from the sample set S according to the (discrete) distribution defined through the importance weights w_i. This sample corresponds to an instance of $p(l')$ in (2).

Sampling: Use l' and the control information a_n to sample l from the distribution $p(l \mid a_n, l')$. This sample represents $p(l)$ on the left side (2).

Importance sampling: Weight the sample l by the importance weight $p(s_n \mid l)$, the likelihood of the sample l given the measurement s_n.

After n iterations, the importance weights of the newly generated samples are normalized so that they sum up to 1. It can be shown that the sample set consisting of these samples in fact approximates the posterior density for $p(l)$ [6]. While these steps suffice to efficiently track a robot's position and solve the global localization problem, this basic algorithm is highly inefficient in solving the kidnapped robot problem. Fox and colleagues [9] showed how adding random samples at each iteration allows the algorithm to efficiently recover from localization failures.

In this paper we compare three different methods for adding samples. The first heuristic, sensor resetting localization, adds samples drawn according to landmark observations [16]. Sensor resetting determines the number $\tilde{n}$ of additional samples based on the likelihood of the current observation:

$$\tilde{n} = n \cdot \max(0,\ 1 - \frac{\tilde{p}}{p_t}) \qquad (6)$$

Here $\tilde{p} = \sum_i p(s_n \mid l_i)/n$ is the average likelihood of the observation, and p_t is a threshold determined manually. Whenever $\tilde{p}$ is above this threshold, no samples are added, and whenever $\tilde{p} \leq p_t$, a small fraction of samples is added from the observation. The second approach to adding samples from observations, mixture MCL, additionally weighs these samples with the current probability density $p(l)$ [20]. This approach has been developed specifically for extremely accurate sensor information, and the weighting makes it consistent with the recursive posterior estimation. In our experiments, mixture MCL adds a fixed number of samples to the distribution.

The third approach uses sensor resetting in combination with a more elaborate strategy to determine the number of samples to be added from observations. This approach has been applied by the University of Washington's entry in RoboCup 2002 [5]. The key idea of this approach is to use a combination of two *smoothed estimates* of the observation likelihoods. The first estimate, $\bar{p}_l$, determines the long-term average of observation likelihoods, and the second estimate, $\bar{p}_s$, determines the short-term average observation likelihood. While $\bar{p}_l$ estimates the slow changing noise level in the environment and sensors, $\bar{p}_s$ is used to detect rapid changes in the likelihood due to a failure of the position estimate. The number $\tilde{n}$ of samples to be added is determined as follows:

$$\begin{aligned} \bar{p}_l &\leftarrow \bar{p}_l + \eta_l(\tilde{p} - \bar{p}_l) \\ \bar{p}_s &\leftarrow \bar{p}_s + \eta_s(\tilde{p} - \bar{p}_s) \\ \tilde{n} &= n \cdot \max(0, 1 - \nu \frac{\bar{p}_s}{\bar{p}_l}) \end{aligned} \qquad (7)$$

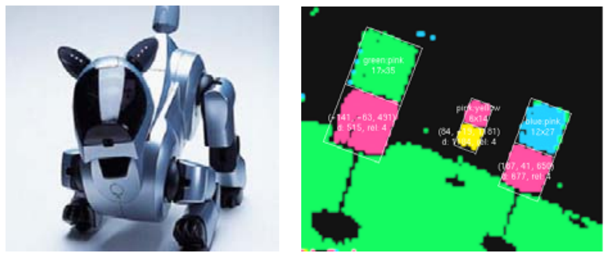
As in (6), $\tilde{p}$ represents the likelihood of the current observation. The only difference between $\bar{p}_l$ and $\bar{p}_s$ lies in the smoothness factors η_l and η_s. The desired smoothing is achieved by setting $0 \leq \eta_l << \eta_s \leq 1$. The parameter ν allows to adjust the level at which samples are added. In essence, this approach only adds samples if the short-term estimate of the observation likelihood is less than $1/\nu$ the long-term average.

In the remainder of this paper, we denote the first method SRL, the second Mix-MCL, and the third A-MCL (adaptive MCL).

3 Experimental Setup

For our experiments we employed an ERS 2100 robot system, a developer version of the commercial AIBO robot (see Fig. 3(a)), connected to a Linux PC by wireless LAN. We programmed the robot to observe color landmarks as shown in Fig. 3(b) that are also used in RoboCup competitions in the Sony legged robot league.

The detection of color tubes is realized by employing the CMVision software library [3] on color labeled images provided by the robot's vision hardware. By taking into account the kinematic chain of the robot's head, distance and bearing to landmarks in 2D coordinates are computed. Due to the rastering onto pixels and noise in the joint-angle sensors, the obtained distance and bearing values are erroneous and have been modeled as Gaussian



(a) (b)

Fig. 3. *(a) AIBO Entertainment Robot System and (b) color landmark tubes as observed by the robot's CMOS camera.*

densities [12]. For landmark distance, a standard deviation of about 15 % of the measured distance and for bearing, a fixed value of 10° was used throughout the experiments. Motion and sensor models are implemented on the robot providing their estimates to the localization system on the PC.

For evaluation, we built an environment of size 3x2m with 6 landmarks (see Fig. 4) almost identical to the setup used in the RoboCup Sony legged robot league. Several positions inside the field were marked by tape and the robot was joysticked around while swinging its head for about 1h. All sensor and motion data has been logged to a file adding a special tag each time the robot passed over one of the tape markers (as observed by the operator).

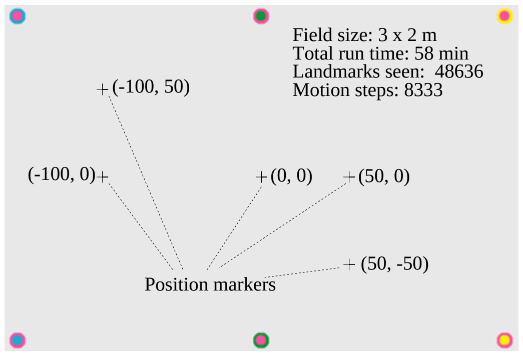


Fig. 4. *Environment for performing localization experiments.*

4 Results

Using the recorded data we conducted a series of experiments to determine accuracy, robustness and relocalization speed of the following localization methods:

EKF: A simple extended Kalman filter integrating all motions and observations.

ML-EKF: The ML-EKF system using the parameters cell size = 10 cm, t_{obs} = 0.001, and t_{χ^2} = 9.

SRL (1/2): Sensor Resetting Localization according to [16]. The experiments were conducted with different values for the threshold p_t used in (6) to determine the number of additional samples. We report results for two different values: SRL1 for $p_t = 0.000025$ and SRL2 for $p_t = 0.0000025$.

Mix-MCL: Mixture MCL has been presented in [20]. We found the best performance by adding 2 observation samples per iteration. The probability of these observation samples was determined by a grid with cell size = 30cm and angular resolution = 20deg.

A-MCL: The adaptive MCL method determined the number of observation samples based on (7). We used values of 0.1 and 0.001 for η_s and η_l, respectively. The parameter ν was set to 2.

All sample-based approaches used 30 samples to represent $p(l)$. Note that we experimentally tuned the parameters of all methods in order to obtain best results.

The leftmost data points in Fig. 5 show the mean absolute position error of the different localization methods when processing the original data of the logged motions and observations. In this and all following figures error bars indicate 95 % confidence intervals.

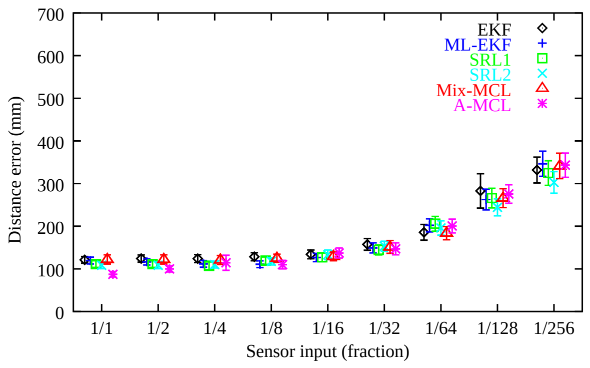


Fig. 5. *Accuracy of localization methods under different levels of sensor data sparsity.*

The absolute error of all methods is between 87 mm (A-MCL) and 122 mm (Mix-MCL). It should be noted that part of the absolute error is due to the problem of joysticking the robot exactly onto the tape markers, human observation of the robot being on a marker, and a marginal uncertainty in time when adding a tag to the log. For these reasons, the true absolute errors are slightly smaller than the ones reported here.

By discarding landmark observations from sensor data we can infer how accurate the different methods are under sensor data sparsity. Fig. 5 plots the mean position error when only a fraction of available sensor data is presented to the localization methods. All algorithms show almost equal performance with the mean error increasing when reducing the number of landmark observations.

To find out about robustness under sensor noise, we replaced a certain fraction of landmark observations by random landmark data. Fig. 6 shows the mean error for all methods under different levels of such sensor noise. Whereas ML-EKF, SRL1, and A-MCL are still capable of providing accurate position estimates for sensor noise up to 50%, EKF and SRL2 produce significantly worse results. For the parameters used in this experiment, Mix-MCL produces results lying between SRL1 and SRL2.

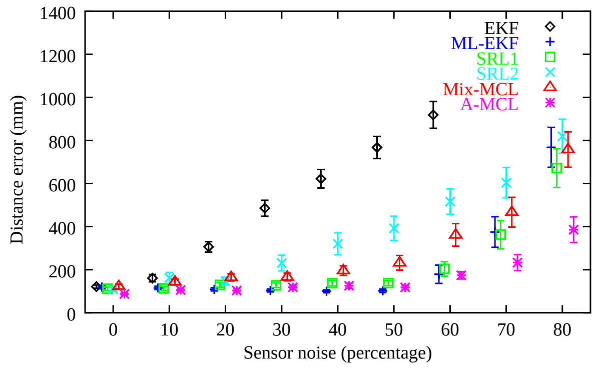


Fig. 6. *Accuracy of localization methods under different levels of sensor noise. Results for EKF have been cut at a noise level of 70 % as the error was above 1 m.*

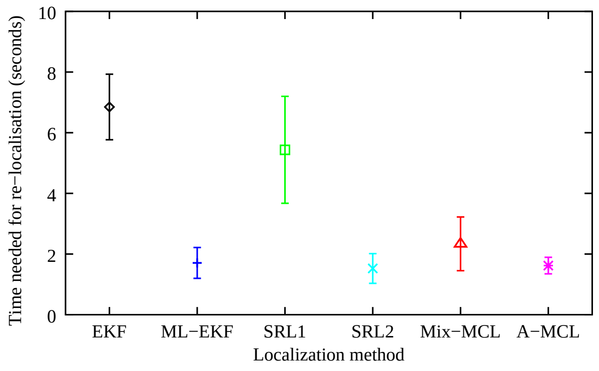


Fig. 7. *Time for recovering from manual robot displacement.*

The extended Kalman filter is not able to deal with such noise, most probably because of its limited density representation and the linearizations performed in the update equations (additional noise filter techniques should increase the noise robustness of the EKF). SRL2 fails because the parameter setting forces the algorithm to add observation samples on each noisy observation. These additional samples inject further uncertainty into the sample sets and cause larger localization errors. Note that Mix-MCL also adds observation samples at each iteration, but in contrast to SRL, Mix-MCL additionally weighs these samples with the current density $p(l)$, thereby increasing robustness to noise. The robustness of ML-EKF is due to the sensor filtering based on the Markov grid. On higher noise levels all methods significantly degrade, with A-MCL producing better estimates than all others. This superior performance of A-MCL is due to the ability to adapt to different levels of noise.

In the next set of experiments we analyzed the ability of the methods to solve the kidnapped robot problem. To do so, we computed the average time the methods needed for relocalizing the robot after it has been manually displaced. Fig. 7 shows that ML-EKF, SRL2, Mix-MCL and A-MCL recover in a very short time (about or less than 2 seconds) and are significantly faster than EKF and SRL1. Whereas ML-EKF and the MCL methods use explicit motion models for manual robot displacement, Kalman filtering runs into severe problems most probably because the Gaussian motion model does not account for such replacements. SRL1 fails because of its parameter setting, allowing almost no samples to be drawn from the observations (which caused SRL1 to be pretty robust to sensor noise, cf. Fig. 6). The parameter settings in SRL are a problem in general as we found no good trade-of between robustness against sensor noise and ability for fast recovery from robot displacement.

As a final measure we computed the computational complexity of the different algorithms when processing the 1 hour log file containing about 48,000 landmark observations and about 8,300 motion steps performed by the robot. Table 1 shows the run times of the different algorithms. The numbers in the second and third column are the times needed per prediction and per observation, respectively. The last column contains the total time when processing all data in the log. Note that these values only include the time for updating the state of each system, not included, e.g., is time for performing i/o.

Method	Prediction	Observation	Sample
EKF	2 μs	6 μs	0.41 s
ML-EKF	36 μs	69 μs	4.13 s
SRL	48 μs	63 μs	4.78 s
Mix-MCL	48 μs	122 μs	7.32 s
A-MCL	51 μs	64 μs	4.93 s

Table 1: *Run times of the different algorithms on a PIII@1GHz broken down for prediction and observation. Last column shows sample total time when processing an 1h log file. See text for comments.*

As can be seen the algorithms are extremely fast needing only a small fraction of the overall time the robot was moving in the environment. Mixture MCL needs slightly more time than the other algorithms as the weighting of the sensor sampled particles with the previous state $p(l)$ requires an explicit representation of $p(l)$ (here realized by a grid). Nevertheless the total numbers for processing the data become less relevant if we consider that the algorithms produce more or less the same output if many observations are simply omitted (as seen in Fig. 5) plus the fact that in the complete robot navigation system, most of the time is spent in the recognition of landmarks anyway.

5 Discussion and Conclusion

We presented an experimental comparison of localization methods involving Kalman filter and particle filter based methods in an environment containing artificial landmarks. Our experiments show that a combination of grid-based Markov localization and Kalman filtering (ML-EKF) as well as particle filters outperform the vanilla extended Kalman filter. We expect that additional noise filtering can significantly increase the robustness of the EKF. However, to speed up the recovery from localization failures, the EKF seems to require more complex approaches such as fault detection or multi hypothesis tracking [1]. The ML-EKF method and adaptive MCL are very well suited for the environment used in our experiments and outperform mixture MCL (in robustness in noise) and SRL (either in robustness in noise or in the time for recovering from manual robot displacement). While SRL can be tuned to be robust to noise (SRL1) or to allow quick recovery from localization failures (SRL2), it is not able to solve both problems simultaneously. This is mostly due to the fact that the number of observation samples is based solely on the likelihood of the current observation. Adaptive MCL, on the other hand, uses smoothing to get good estimates of the environment noise and of the current localization performance (see (7)). Thereby, A-MCL significantly outperforms SRL wrt. robustness and failure recovery. Furthermore, A-MCL is as good as ML-EKF in dealing with sparse sensor data and in localization recovery, but provides better robustness to high noise levels.

We consider ML-EKF and the sample-based methods discussed in this paper to be good candidates for landmark based localization systems. Another highly interesting result is the fact that all approaches perform extremely well when applied to sparse sensor data. These findings suggest that it is not necessary to process all sensor data, which allows to save major fractions of processor time.

How do these results transfer to complex environments with non-unique landmarks or other sensor types such as sonar or laser range-finders? Extended Kalman filters and sample-based methods have already been applied successfully to large indoor environments in combination with a variety of sensors [9, 13]. Recently, Fox showed that the efficiency of particle filters can be increased drastically by adapting the size of sample sets on-the-fly [8]. The heuristic of sensor resetting [16] is not directly applicable to these environments and sensors, but Thrun and colleagues showed how to use kd-trees to generate samples from arbitrary sensors [20]. However, it is not clear whether the benefits outweigh the additional computational and implementational costs.

An experimental comparison conducted in an office environment [13] showed that EKF is not capable to efficiently solve the global localization problem. Further experiments have to show whether the progress achieved through ML-EKF transfers directly to such environments. It is unclear whether the 2D grid is expressive enough to support the Kalman filter in arbitrary environments. The experiments in [13] also suggest that approaches based on 3D grids are extremely robust in the general case [4]. Their major drawback is computational complexity, even when considering efficient implementations as in [10]. Another highly promising approach for increasing the robustness of Kalman filters is multi hypothesis tracking [1]. This approach maintains multiple hypotheses for a robot's location, and each hypothesis is tracked with a Kalman filter. We would like to see a comparison of such a system with the ones reported in this work in the future.

References

[1] D. Austin and P. Jensfelt. Using multiple Gaussian hypotheses to represent probability distributions for mobile robot localization. 2000.

[2] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[3] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2000.

[4] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. 14th National Conference on Artificial Intelligence (AAAI'96)*, pages 896–901, Aug. 1996.

[5] Z. Crisman, E. Curre, C. Kwok, N. Ratliff, L. Tsybert, and D. Fox. Team description: UW Huskies-02. In G. Kaminka, P. Lima, and R. Rojas, editors, *RoboCup-2002: Robot Soccer World Cup VI*. Springer Verlag, 2003.

[6] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo in Practice*. Springer-Verlag, 2001.

[7] T. Duckett and U. Nehmzow. Knowing your place in real world environments. In *European Workshop on Advanced Mobile Robots (EUROBOT'99)*, 1999.

[8] D. Fox. KLD-sampling: Adaptive particle filters and mobile robot localization. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[9] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proc. National Conference on Artificial Intelligence (AAAI'99)*, 1999.

[10] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11, 1999.

[11] D. Fox, S. Thrun, F. Dellaert, and W. Burgard. Particle filters for mobile robot localization. In Doucet et al. [6].

[12] J.-S. Gutmann. Markov-Kalman localization for mobile robots. In *Int. Conf. on Pattern Recognition (ICPR)*, 2002.

[13] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 1998.

[14] J.-S. Gutmann, T. Weigel, and B. Nebel. A fast, accurate, and robust method for self-localization in polygonal environments using laser range finders. *Advanced Robotics Journal*, 14(8):651–668, 2001.

[15] L. Iocchi, D. Mastrantuono, and D. Nardi. A probabilistic approach to Hough localization. In *Int. Conf. on Robotics and Automation (ICRA)*, Seoul, Korea, 2001.

[16] S. Lenser and M. Veloso. Sensor resetting localization for poorly modeled mobile robots. In *Int. Conf. on Robotics and Automation (ICRA)*, 2000.

[17] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transaction on Robotics and Automation*, 7(3):376–382, 1991.

[18] P. S. Maybeck. The Kalman filter: An introduction to concepts. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 194–204. Springer-Verlag, 1990.

[19] C. F. Olson. Probabilistic self-localization for mobile robots. *IEEE Transactions on Robotics and Automation*, 16(1):55–66, Feb. 2000.

[20] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.