A Novel Approach to Efficient Monte-Carlo Localization in RoboCup

Patrick Heinemann, Jürgen Haase, and Andreas Zell

Wilhelm-Schickard-Institute, Department of Computer Architecture,
University of Tübingen, Sand 1, 72076 Tübingen, Germany
{heinemann,jhaase,zell}@informatik.uni-tuebingen.de

Abstract. Recently, efficient self-localization methods have been developed, among which probabilistic Monte-Carlo localization (MCL) is one of the most popular. However, standard MCL algorithms need at least 100 samples to compute an acceptable position estimation. This paper presents a novel approach to MCL that uses an adaptive number of samples that drops down to a single sample if the pose estimation is sufficiently accurate. Experiments show that the method remains in this efficient single sample *tracking mode* for more than 90% of the cycles.

1 Introduction

Self-localization has been a major research task in mobile robotics for several years. Especially in the Middle-Size-League of RoboCup where robots are expected to carry out cooperative tasks, it is crucial to know the robot's position in a common global coordinate system. Many different approaches to the localization task in RoboCup environments have been investigated over the last years. Nearly all of these methods are based on the relative distance and angle of features to the robot in two-dimensional field coordinates.

When walls still surrounded the RoboCup field, some teams used line segments extracted from the distance data of a laser range finder for a very fast localization ([4]). Today, the idea of extracting lines from distance data found its way into several other approaches, now using distances to field markings generated by camera systems. Iocchi *et al.* [6] as well as Marques *et al.* [7], and Jong *et al.* [1] presented algorithms where the lines are extracted using the Hough Transform. Instead of matching the lines to a model, Utz *et al.* [11] computed a distance to the model lines at given positions, to exploit the advantages of Monte-Carlo localization (MCL) [3]. Although it is obvious to extract lines as landmarks in a mostly polygonal environment like RoboCup, algorithms were developed, that were able to handle the raw distance data from the sensors. Probabilistic approaches like Markov localization use sensor data to assess given position estimates. As this is much easier than generating a position hypothesis through feature extraction and model matching, all algorithms based on raw distance data used Markov localization or more precise MCL, which became one of the most popular localization methods. Enderle *et al.* [2] presented an approach using the distance to walls extracted from camera images, while Hundelshausen

G. Lakemeyer et al. (Eds.): RoboCup 2006, LNAI 4434, pp. 322–329, 2007.

et al. [12], Röfer *et al.* [9,10] and Menegatti *et al.* [8] used the distance to the field markings. These algorithms mainly differ in the efficiency of the assessment of position estimates and the number of samples needed for the localization.

The self-localization algorithm presented in this paper is based on line points of the field's line markings. The fitness evaluation of different position estimates is based on a two-dimensional look-up table containing the distance to the next marking line for every position on the field [12,9,10]. To maintain a high accuracy a local search is used to iteratively improve the position estimation, as presented by Hundelshausen *et al.* [12]. In contrast to Hundelshausen *et al.*, however, who use their ideas only for dead-reckoning after an initial global localization, we combine the advantages of MCL with the iterative improvement of the position estimation. This algorithm uses an adaptive number of samples that is reduced to a single sample, if the position estimation of the previous cycle was sufficiently accurate. Experiments show that our algorithm remains in this efficient single sample *tracking mode* for more than 90% of the cycles. Nevertheless, it is still able to cope with the kidnapped robot problem.

The remainder of this paper is organized as follows: Section 2 is a detailed description of the proposed algorithm. Results concerning efficiency and accuracy of the algorithm are presented in section 3 and section 4 concludes the paper with an outlook on the future work.

2 Improved Monte-Carlo Localization

The major steps of the proposed algorithm are shown in Fig. 1 compared to a typical Monte-Carlo localization algorithm.

2.1 Initialization

When the algorithm starts the maximum number of samples $N_{\max}$ is generated and randomly distributed over the state space, which in RoboCup consists of the whole playable area of the field. If there is previous knowledge of the robot's pose, this knowledge can be represented by a different non-random initialization.

2.2 Application of the Motion Model

In the motion model the odometry information from the robot is incorporated. First, the samples of the set S are translated and rotated according to the observed motion a. Then a random gaussian noise is added proportional to the motion. In the tracking mode, i.e. only a single sample is used, this step only consists of translating and rotating the pose of the single sample.

2.3 Evaluation of the Sensor Model

The proposed algorithm uses the marking lines on a RoboCup soccer field as features for the self-localization. Several pixels in an omnidirectional camera image are identified as marking line points as shown by Heinemann *et al.* [5].

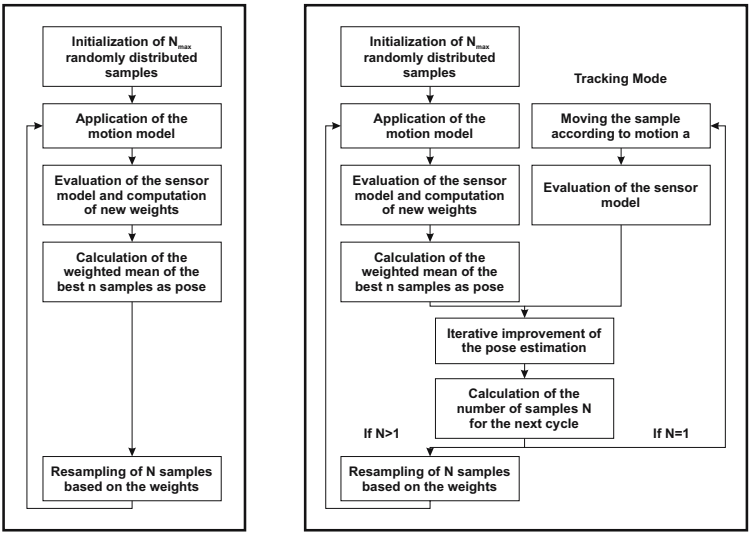

Fig. 1. The major steps of a typical Monte-Carlo localization algorithm (left) and our improved Monte-Carlo localization algorithm (right)

These pixels transformed into robot centered coordinates (x_j, y_j) serve as input for the sensor model introduced in this section.

For an efficient evaluation of the sensor model Röfer *et al.* [9,10] and Hundelshausen *et al.* [12] presented an idea of transforming the line points to the pose $l_i = (x_i, y_i, \theta_i)$ of the sample s_i, such that the coordinate system of the line points is located at position (x_i, y_i) and oriented according to θ_i. Denoting the new location of the line points as $(x_{i,j}, y_{i,j})$ and the vector from these points to their nearest model line in a model of the field as

$$f_{i,j} = (x_{m,j} - x_{i,j}, y_{m,j} - y_{i,j}), \tag{1}$$

an overall distance $D_{L,i}$ per sample can be calculated by summing over the squared distances to the model as

$$D_{L,i} = \frac{1}{j} \sum_{j} \| f_{i,j} \|^2. \tag{2}$$

As these distances only depend on the position on the field they can be precomputed on a discrete grid and easily stored in a two-dimensional look-up table (*distance matrix*). In contrast to the original methods the proposed algorithm uses the squared distances to let line points with a higher distance to the next model line have an even greater influence than line points that are almost perfectly matched. As $D_{L,i}$ is only based on the symmetric marking lines on a RoboCup soccer field, it would be the same for at least two poses in each cycle. Thus, to resolve the symmetry the angle to the two differently coloured goals

was introduced as an extra feature. From the omnidirectional image the angles $\widehat{\phi}_{1,i}$ and $\widehat{\phi}_{2,i}$ to the two goals are extracted. Comparing these angles with the expected angles at the pose of a sample $\phi_{1,i}$ and $\phi_{2,i}$ results in a goal distance

$$D_{G,i} = \left(\left\| \widehat{\phi}_1 - \phi_{1,i} \right\| + \left\| \widehat{\phi}_2 - \phi_{2,i} \right\| \right)^2, \tag{3}$$

where $\|\cdot\|$ is the absolute value of the smaller angle difference accounting for the 2π period of angles. Again this distance is squared to let higher angular differences have a greater influence. The total distance value is computed as

$$D_i = (1 - \lambda) D_{L,i} + \lambda D_{G,i}, \tag{4}$$

with $\lambda \in [0, 1]$ representing the balance of the two distance terms, and finally, the weights are updated as

$$w_{i,t} = \alpha \frac{1}{D_i}\ , \tag{5}$$

with α such that $\sum_i w_i = 1$. In the tracking mode the distances are only computed for the calculation of the number of samples used in the next step N_{t+1}.

2.4 Iterative Improvement of the Pose Estimation

A preliminary pose estimation is calculated as weighted mean over all samples

$$\widehat{p} = (x, y, \theta) = \sum_n w_n l_n, \tag{6}$$

and is used as starting pose for the iterative improvement. In the tracking mode $\widehat{p}$ is the pose of the single sample.

In addition to the *distance matrix* Hundelshausen *et al.* [12] proposed a dead-reckoning approach for self-localization. By applying forces exerted on the transformed line points by the model lines an estimated position (x, y) is iteratively improved in both directions. Using the same forces a torque is computed which iteratively improves the orientation θ. Again, these forces can be precomputed and stored in a look-up table (*force matrix*). Here we use the force matrix to improve the pose estimation $\widehat{p}$ from the MCL in a number of iterations k. It contains the two-dimensional vectors $f_{i,j}$ from equation (1) that can be interpreted as a force exerted by the nearest model line proportional to the distance. A mean force acting on the pose estimation $\widehat{p}$ can be computed as

$$F = \frac{1}{j} \sum_j f_{i,j}. \tag{7}$$

A fraction of this force can be added to the pose estimation $\widehat{p}$ in each iteration to improve it regarding the position. In contrast to Hundelshausen *et al.* we compute a mean torque according to the estimated pose to improve the orientation estimation. It is computed over all line points as

$$M = \frac{1}{j} \sum_j (x_{i,j}, y_{i,j}) \times f_{i,j}. \tag{8}$$

Thus, in each iteration k a new pose estimation $\widehat{p}_k = (x_k, y_k, \theta_k)$ is generated by

$$(x_k, y_k) = (x_{k-1}, y_{k-1}) + \mu F \tag{9}$$

$$\theta_k = \theta_{k-1} + \nu M, \tag{10}$$

starting with the preliminary estimation

$$(x_0, y_0, \theta_0) = \widehat{p}. \tag{11}$$

The iterations can be seen as a local search that minimizes F and M and thus stabilizes the pose estimation by removing the noise from the weighted mean when $N_t > 1$ and reducing the tracking errors when $N_t = 1$. The search continues until a maximum number of iterations k_{max} is reached or the improvement between the iterations was too low. The final pose estimation p is the pose resulting from the last iteration. Please note that apart from inserting the improved pose estimation p into the sample set S_{t+1} the stochastic process of the Monte-Carlo Localization is not influenced by the iterative improvement.

2.5 Calculation of the Number of Samples

The number of samples needed for the next cycle is calculated depending on the distance D of the final pose estimation p according to equation (4) as

$$N_{t+1} = \begin{cases} N_{\max} & : \text{ if } \xi D \geq N_{\max} \\ \xi D & : \text{ if } 1 < \xi D < N_{\max} \\ 1 & : \text{ if } \xi D \leq 1 \end{cases}, \tag{12}$$

where ξ is a factor that controls how fast the number of samples n is reduced.

2.6 Resampling

If $N_t > 1$ the cycle ends with an importance resampling from the set of samples S with probability $w_{i,t}$ for resampling an old sample $s_{i,t}$. The sampling continues until the number of samples N_{t+1} for the next cycle was reached. To represent the improved pose information p in the sample set, this pose is inserted as new sample into the sample set S_{t+1} for the next cycle.

3 Results

This section presents results obtained by two experiments made in our robot lab on a half field of $7m$ width and $4m$ length. Throughout this section positions and orientations are given in meters and radians, respectively. In all experiments presented in this section we used the parameters given in table 1. The maximum number of samples N_{max} used was chosen such that it is comparable to a standard MCL approach with a fixed number of samples.

In a first experiment we compared the localization algorithm with a fixed number of samples $N = 200$, $N = 100$, $N = 50$ and the proposed method. As a

Table 1. Parameter set used for the experiments

Nmax	λ	ξ	μ	ν	k_{max}
200	0.1	2500	0.001	0.0003	20

database for the comparison we located the robot at a pose $p_1 = (1.04, 1.07, 2.1)$ and stored the detected line points of 98 images from the omnidirectional camera system. Afterwards, the robot was relocated to pose $p_2 = (1.64, 2.68, 0.0)$ where the line points from another 98 images were stored. The line points were used as input to 196 cycles of the localization algorithm without any odometry information, resulting in a kidnapped robot problem. Fig. 2 shows the estimation error of the algorithms. Independent of N the algorithms compute a very good pose estimation after at most 15 cycles, where the number of samples in the adaptive method drops to $N = 1$ in only 6 cycles. From cycle 20 to 90 the estimation error and the number of samples stay at the same level. In cycle 99 where the relocation of the robot happened the proposed algorithm immediately generates $N = N_{\max}$ samples, reacting to the high estimation error. Apart from the algorithm with $N = 50$ samples all methods regenerate a good pose estimation after at most 10 cycles, whereas the number of samples in the adaptive method returns to $N = 1$ after 8 cycles, thus using only a single sample in 92.87% of the cycles. A fixed number of $N = 50$ samples without the iterative improvement is not able to handle the kidnapped robot problem in this case, as the estimation error does not recover after the relocation in cycle 98. Although the other three algorithms show comparable results concerning the estimation error, the proposed algorithm performes much better if the computation time is considered. Table 2 lists the mean computation time and estimation error.

With the second experiment we show that the method is also able to correctly track the pose of a moving robot. As a ground truth we used a laser scanner to record the true position of the robot. The robot was then manually controlled around the field. In the first run the mean speed was at 1 m/s, in the second run we raised the speed to 2 m/s. To show that the algorithm works for both differential drive and omnidirectional systems we first controlled the robot like a

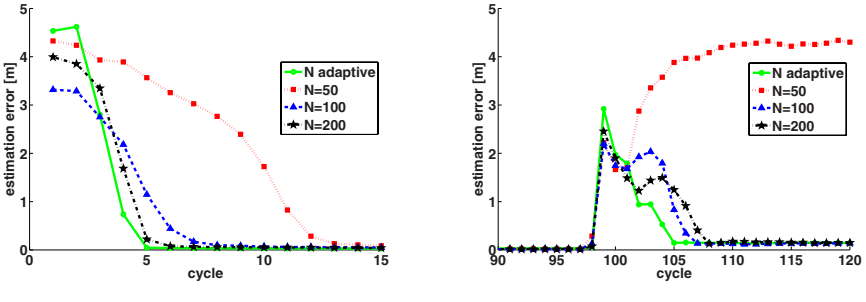

Fig. 2. Comparison of our algorithm to MCL with fixed numbers of samples

Table 2. Results of 196 cycles using different numbers of particles N

	N adaptive	$N = 50$	$N = 100$	$N = 200$
mean time	1.7632ms	3.6426ms	6.8223ms	14.2508ms
mean error	0.1936m	2.2515m	0.2075m	0.2057m

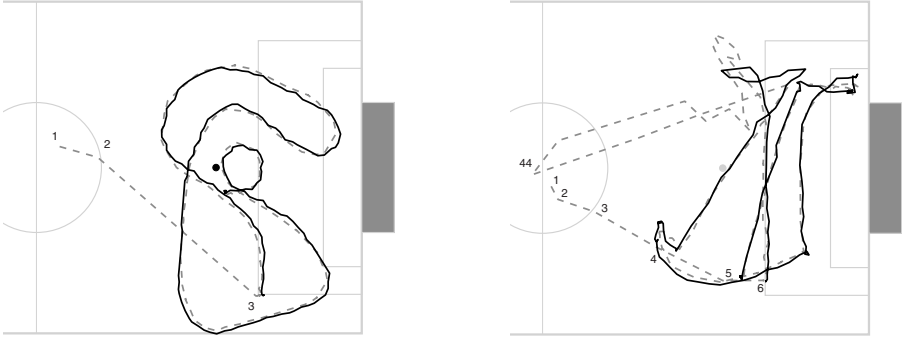

Fig. 3. This figure compares the position estimates of the proposed algorithm (dashed grey line) to the ground truth (solid black line). The algorithm needs up to 6 cycles to find the initial pose and then correctly tracks the robot. In the second run, the algorithm fails to track the position once, but relocalizes only a few cycles later.

differential drive and then the orientation was nearly fix in the second run. Fig. 3 (left) shows the results of the first run. In the beginning the weighted mean over the rndomly distributed samples results in a pose near the center of the field. After 3 cycles the starting pose of the robot is correctly estimated. Throughout the rest of the 212 cycles the estimated pose follows the path on the field with a mean accuracy of 9.89cm, using only a single sample in 97.17% of the cycles. The mean computation time for a cycle of the algorithm in this experiment was 1.5731ms on an Athlon XP 1800+ system. In the second run (Fig. 3, right) the algorithm needed 6 cycles to correctly estimate the starting position. The estimated position then follows the real position until the robot changes its direction very quickly two times in a row. Here the algorithm temporarily looses the track of the robot and distributes a higher number of samples (cycle 44). Thus, the mean accuracy without the initialization cycles was 23.97cm and the mean computation time increased to 4.32ms as only 90.64% of the cycles used the tracking mode.

4 Conclusion and Future Work

This paper presents an efficient combination of global Monte-Carlo localization with an adaptive number of samples and local position tracking. With a fast estimation of the samples' fitness and a local search for iterative improvement of

the estimated pose the number of samples was reduced to a single sample resulting in a smooth transition between global localization and local pose tracking. We showed that the algorithm was able to handle the kidnapped robot problem and to track a moving robot. In all experiments the mean cycle time of the algorithm was leaving enough time for other important tasks like object detection and planning to be done in real-time.

References

1. de Jong, F., Caarls, J., Bartelds, R., Jonker, P.: A Two-Tiered Approach to Self-Localization. In: Birk, A., Coradeschi, S., Tadokoro, S. (eds.) RoboCup 2001. LNCS (LNAI), vol. 2377, pp. 405–410. Springer, Heidelberg (2002)
2. Enderle, S., Ritter, M., Fox, D., Sablatnög, S., Kraetzschmar, G., Palm, G.: Vision-based Localization in RoboCup Environments. In: Stone, P., Balch, T., Kraetzschmar, G.K. (eds.) RoboCup 2000. LNCS (LNAI), vol. 2019, pp. 291–296. Springer, Heidelberg (2001)
3. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In: Proceedings of the National Conference on Artificial Intelligence, pp. 343–349 (1999)
4. Gutmann, J., Weigel, T., Nebel, B.: Fast, Accurate, and Robust Self-Localization in Polygonal Environments. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '99) (1999)
5. Heinemann, P., Rückstieß, T., Zell, A.: Fast and Accurate Environment Modelling using Omnidirectional Vision. In: Dynamic Perception 2004, Infix (2004)
6. Iocchi, L., Nardi, D.: Self-Localization in the RoboCup Environment. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup-99: Robot Soccer World Cup III. LNCS (LNAI), vol. 1856, pp. 318–330. Springer, Heidelberg (2000)
7. Marques, C., Lima, P.: A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor. In: Proceedings of EuRoboCup Workshop 2000 (2000)
8. Menegatti, E., Pretto, A., Pagello, E.: A New Omnidirectional Vision Sensor for Monte-Carlo Localization. In: Nardi, D., Riedmiller, M., Sammut, C., Santos-Victor, J. (eds.) RoboCup 2004. LNCS (LNAI), vol. 3276, pp. 97–109. Springer, Heidelberg (2005)
9. Röfer, T., Jüngel, M.: Vision-Based Fast and Reactive Monte-Carlo Localization. In: Proceedings of the 2003 IEEE International Conference on Robotics & Automation, pp. 856–861. IEEE Computer Society Press, Los Alamitos (2003)
10. Röfer, T., Jüngel, M.: Fast and Robust Edge-Based Localization in the Sony Four-Legged Robot League. In: RoboCup-2003: Robot Soccer World Cup VII. LNCS, vol. 3020, pp. 262–273. Springer, Heidelberg (2004)
11. Utz, H., Neubeck, A., Mayer, G., Kraetzschmar, G.: Improving Vision-Based Self-localization. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 25–40. Springer, Heidelberg (2003)
12. von Hundelshausen, F., Schreiber, M., Wiesel, F., Liers, A., Rojas, R.: MATRIX: A force field pattern matching method for mobile robots. Technical Report B-08-03, Free University of Berlin (2003)