A Combined Monte-Carlo Localization and Tracking Algorithm for RoboCup

Patrick Heinemann*, Juergen Haase* and Andreas Zell*
* Wilhelm-Schickard-Institute
Department of Computer Architecture
University of Tübingen
Email: {heinemann, jhaase, zell}@informatik.uni-tuebingen.de

***Abstract*— Self-localization is a major research task in mobile robotics for several years. Efficient self-localization methods have been developed, among which probabilistic Monte-Carlo localization (MCL) is one of the most popular. It enables robots to localize themselves in real-time and to recover from localization errors. However, even those versions of MCL using an adaptive number of samples need at least a minimum in the order of 100 samples to compute an acceptable position estimation. This paper presents a novel approach to MCL based on images from an omnidirectional camera system. The approach uses an adaptive number of samples that drops down to a single sample if the pose estimation is sufficiently accurate. We show that the method enters this efficient *tracking mode* after a few cycles and remains there using only a single sample for more than 90% of the cycles. Nevertheless, it is still able to cope with the kidnapped robot problem.**

I. Introduction

Self-localization has been a major research task in mobile robotics for several years. Especially in the Middle-Size-League of RoboCup [8] 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.

Back in 1999, when walls still surrounded the RoboCup field, the CS-Freiburg and the Attempto Tübingen team used line segments extracted from the distance data of a laser range finder for a very fast localization ([5],[11]). However, as soon as the walls were removed, these sensors became useless for self-localization. Nevertheless, 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.* [7] presented an algorithm where the lines are extracted using the Hough Transform as well as Marques *et al.* [9], and Jong *et al.* [1]. Instead of matching the lines to a model, Utz *et al.* [14] computed a distance to the model lines at given positions, to exploit the advantages of Monte-Carlo localization (MCL) [4].

Although it is obvious to use lines as landmarks in a mostly polygonal environment like RoboCup, the line extraction of most of these algorithms is too slow to process the full 25fps of typical PAL camera systems, which would be neccessary to compete in such a highly dynamic environment. Therefore, algorithms were developed, that were able to handle the raw distance data from the sensors, without the need for generating a position hypothesis. Probabilistic approaches like Markov localization use sensor data only to assess given position estimates. As this is much easier than generating a position hypothesis out of some features, 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 perspective camera images. Hundelshausen *et al.* [15], Röfer *et al.* [12], [13] and Menegatti *et al.* [10] instead used the distance to the field markings. These versions of MCL mainly differ in the efficiency of the assessment of position estimates and the number of samples needed for an accurate localization.

The self-localization algorithm presented in this paper is based on line points of the field's line markings. They result from preprocessing images of an omnidirectional camera [6]. 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 [15], [12], [13]. To maintain a high accuracy the idea of comparing the line points to their nearest marking line in the model resulting in a distance measure is extended to forces and a torque exerted by the model lines trying to match the line points correctly. With these forces the estimated position can be iteratively improved, as presented by Hundelshausen *et al.* [15]. In contrast to Hundelshausen *et al.*, however, who use their ideas only for dead-reckoning after an initial global localization, we present an algorithm that combines the advantages of Monte-Carlo localization with the iterative improvement of the position estimation using forces. 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. We show that our algorithm remains in this efficient *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 II briefly presents the basics of probabilistic Monte-Carlo localization. Section III is a detailed description of the proposed algorithm . Finally, results concerning efficiency and accuracy of the algorithm are presented in section IV and section V concludes the paper with an outlook on the future work.

II. Monte-Carlo Localization

As Monte-Carlo localization is based on Markov Localization, both methods are briefly introduced in this section.

In Markov Localization [3] the robot's belief for being located at a distinct pose $l = (x, y, \theta)$, where (x, y) is a position and θ is an orientation in the reference coordinate system, is expressed as a continuous probability function $Bel(l)$ over all possible poses, called *belief function*. This function can model multi-modal distributions and is thus able to represent more than one pose hypothesis. If new information about the current pose is available, the belief function is updated as follows:

$$Bel(l) = \int P(l|l', a) Bel(l') dl', \qquad (1)$$

if the robot made a movement a and

$$Bel(l) = \alpha P(s|l) Bel(l), \qquad (2)$$

if the robot made a new sensor reading s. Here $P(l|l', a)$ models the conditional probability of being at pose l given that the robot made the movement a and was at pose l' before, called *motion model*. $P(s|l)$ models the conditional probability for receiving a sensor measurement s given that the robot is located at pose l, called *sensor model*. α is a normalization factor ensuring

$$\int Bel(l) dl = 1. \qquad (3)$$

There are several grid-based methods for approximating the continuous belief function but they are computationally expensive and the accuracy of the pose estimation depends on the a priori defined resolution of the grid [3].

Monte-Carlo Localization (MCL) [4] is an efficient sample-based approximation of the belief function using a *sampling/importance resampling* (SIR) approach that assigns the majority of samples to poses of high probability. Given a set of samples

$$S = \{s_i | s_i = ((x_i, y_i, \theta_i), w_i)\}, \qquad (4)$$

where each sample s_i contains a pose information (x_i, y_i, θ_i) and a weight w_i, the update for a motion a is an importance resampling according to the motion model. A sample is chosen with a probability equal to w_i and the new sample is generated by drawing a random sample from $P(l|l', a)$. For a new sensor measurement the weights of all samples are updated according to the sensor model

$$w'_i \propto P(s|l) \qquad (5)$$

and normalized

$$w_i = \alpha w'_i \qquad (6)$$

with α such that $\sum_i w_i = 1$. Finally, the pose estimation is calculated as the weighted mean over the best n samples. The advantage of the sample-based approximation is that through the importance resampling only samples with a high probability of representing the correct pose are used as input for the next steps. Thus, only few samples are needed to approximate the belief function at the local maxima with a high accuracy and major parts of the belief function with a low probability are not sampled at all, saving computational resources. The number of samples used for the approximation of the belief function can be chosen as a fixed number or can be adapted according to the quality of the pose estimation after each cycle to save even more resources.

III. Improved Monte-Carlo Localization

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

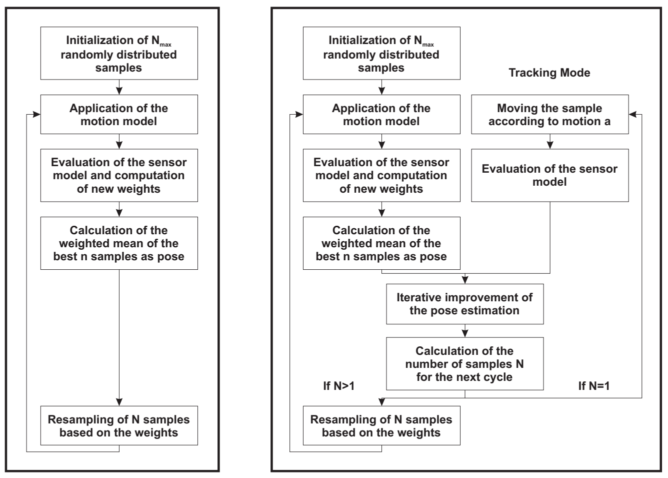
Initialization of N_{max} randomly distributed samples
Application of the motion model
Evaluation of the sensor model and computation of new weights
Calculation of the weighted mean of the best n samples as pose
Resampling of N samples based on the weights
Initialization of N_{max} randomly distributed samples
Tracking Mode
Application of the motion model
Moving the sample according to motion a
Evaluation of the sensor model and computation of new weights
Evaluation of the sensor model
Calculation of the weighted mean of the best n samples as pose
Iterative improvement of the pose estimation
Calculation of the number of samples N for the next cycle
If N>1
If N=1
Resampling of N samples based on the weights

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

A. Initialization

When the algorithm starts there is no previous knowledge of the robot's pose, therefore the maximum number of samples N_{max} is generated and randomly distributed over the state space, which in RoboCup consists of the soccer field plus small parts beyond the touch and goal lines that are used to manoeuvre. If there is previous knowledge of the pose, e.g. if the robot always starts at the same position, this knowledge can be represented by a different non-random initialization.

B. Application of the motion model

In each cycle of the control system a robot usually gets new information of its motion from internal odometry sensors and new data from the external sensors. The odometry information is incorporated in the motion model. 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 sample.

C. 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 if a sequence of green-white-green segments is detected as shown by Heinemann *et al.* [6]. These pixels transformed into robot centered coordinates (x_j, y_j) serve as input for the sensor model introduced in this section.

For a fast and efficient evaluation of the sensor model $P(s|l)$ Röfer *et al.* [12], [13] and Hundelshausen *et al.* [15] 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 base coordinate system of the line points is located at position (x_i, y_i) and oriented according to θ_i as shown in Fig. 3. 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 an a priori known model of the field as

$$f_{i,j} = (x_{m,j} - x_{i,j}, y_{m,j} - y_{i,j}), \tag{7}$$

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

$$D_{L,i} = \frac{1}{j} \sum_j \|f_{i,j}\|^2. \tag{8}$$

As these distances only depend on the position on the field they can be precomputed on a discrete grid (here a resolution of $5cm$ was used) and easily stored in a two-dimensional look-up table (*distance matrix*), which is shown in Fig. 2 as a height map. 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 colour segmentation of 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 to the goal at the pose of a sample $\phi_{1,i}$ and $\phi_{2,i}$ results in a goal distance

$$D_{G,i} = \left(\left\| \widehat{\phi}_1 - \phi_{1,i} \right\| + \left\| \widehat{\phi}_2 - \phi_{2,i} \right\| \right)^2, \tag{9}$$

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 by a linear combination as

$$D_i = (1 - \lambda) D_{L,i} + \lambda D_{G,i}, \tag{10}$$

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

$$w_{i,t} = \alpha \frac{1}{D_i} \ , \tag{11}$$

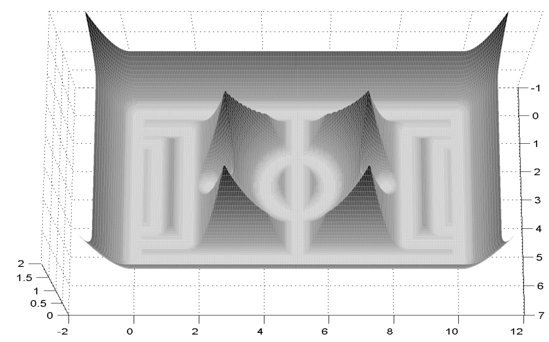

Fig. 2. The *distance matrix*. The height map visualizes the squared distance of each position on the field to the next field marking line.

with α being the normalization factor from equation (6). 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}.

D. Iterative improvement of the pose estimation

The weighted mean over n samples is calculated as the preliminary pose estimation

$$\widehat{p} = (x, y, \theta) = \sum_n w_n l_n, \tag{12}$$

which 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.* [15] proposed a dead-reckoning approach for self-localization. By applying forces exerted by the model lines on the transformed line points an estimated position (x, y) is iteratively improved in both directions. Using the same forces a torque according to the origin of the robot frame 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 adopted this idea and use a force matrix precomputed on a regular grid with $5cm$ resolution to improve the pose estimation $\widehat{p}$ in a number of iterations k. It contains the two-dimensional vectors $f_{i,j}$ from equation (7) that can be interpreted as a force which is exerted onto a line point by the nearest model line proportional to the distance to the model line. A mean force acting on the pose estimation $\widehat{p}$ can be computed as

$$F = \frac{1}{j} \sum_j f_{i,j}. \tag{13}$$

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

$$M = \frac{1}{j} \sum_j (x_{i,j}, y_{i,j}) \times f_{i,j}. \tag{14}$$

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

$$(x_k, y_k) = (x_{k-1}, y_{k-1}) + \mu F \quad (15)$$

$$\theta_k = \theta_{k-1} + \nu M, \quad (16)$$

starting with the preliminary estimation

$$(x_0, y_0, \theta_0) = \widehat{p}. \quad (17)$$

The iterations can be seen as steps of a controller that minimizes F and M. Therefore, if the control parameters μ and ν are set too high, the pose estimation oscillates around the optimum or diverges. We continue to iterate 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. It is important to 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 at all. The iterative improvement can be seen as a local search for a minimum in the belief function which stabilizes the pose estimation by removing the noise from the weighted mean when $N_t > 1$ and reducing the tracking errors when $N_t = 1$.

E. 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 (10) as

$$N_{t+1} = \begin{cases} N_{\max} & : \text{ if } \xi D \geq N_{\max} \\ \xi D & : \text{ if } 1 < \xi D < N_{\max} \\ 1 & : \text{ if } \xi D \leq 1 \end{cases}, \quad (18)$$

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

F. 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. In this step it is possible to insert a number of r randomly distributed samples to ensure a faster recovery if a localization error should occur. 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.

IV. Results

This section presents results obtained by three experiments made in our robot lab on a field of $7m$ width and $4m$ length. As this is very small compared to a real RoboCup soccer field with up to $16m$ width and $12m$ length, we have only the field markings for one half of the field. Thus, our experiments are all situated on the side with the blue goal. 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 I. The mean was computed over all samples, i.e. $n = N$ to avoid a sorting of the samples, which would lower the efficiency of the algorithm. The maximum number of samples N_{max} used was chosen such that it is comparable to the number of samples used in a standard MCL approach with a fixed number of samples. As the two distance measures D_L and D_G are of different units it is very hard to derive a good value for λ from the algorithm itself. Preliminary experiments showed, however, that the chosen value is a good compromise between a stable pose estimation by relying upon the distance to the lines and underestimating the orientation of the robot. The value of ξ results in a fast reduction of the number of samples to only one sample while leaving enough samples to recover from errors when the fitness of the pose estimation is low. The control parameters μ and ν were empirically estimated to prevent the estimation from oscillating around a minimum while still converging as fast as possible.

n	$N_{\max}$	r	λ	ξ	μ	ν	k_{max}
N	200	0	0.1	2500	0.001	0.0003	20

TABLE I
PARAMETER SET USED FOR THE EXPERIMENTS

In the first experiment the influence of the iterative improvement is analysed. The left side of Fig. 3 shows the line points extracted from a typical camera image taken by the robot at a predefined pose $(3.16, 1.37, -0.52)$ and transformed to the preliminary pose estimation $\widehat{p} = (3.17297, 2.11676, -0.43336)$. This estimation was generated by several cycles of the improved MCL. Using the iterative improvement the distance D_L and the estimation error continually drop in each iteration. The final pose estimation $p = (3.15078, 1.36494, -0.479711)$ after the maximum number of iterations is shown on the right. These results show that the iterative improvement can highly refine the pose estimation in situations where the weighted mean is a good approximation of the pose and lead to a fast reduction of the number of samples used. Furthermore, the distance measure D_L is well suited as a means for evaluating the pose estimation error. Although it is possible that the iterations do not improve the pose estimation in situations where the preliminary estimation is completely wrong, this does not reduce the accurracy of the algorithm as the iterations are stopped if the change in distance was low or even negative.

In a second experiment we compared the localization algorithm with a fixed number of samples $N = 200$, $N = 100$, $N = 50$ and the proposed method with adaptive number of samples including the iterative improvement. Our algorithm shows comparable results in terms of estimation accuracy while using only a fraction of the computational resources of a MCL with fixed number of samples. As a 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,

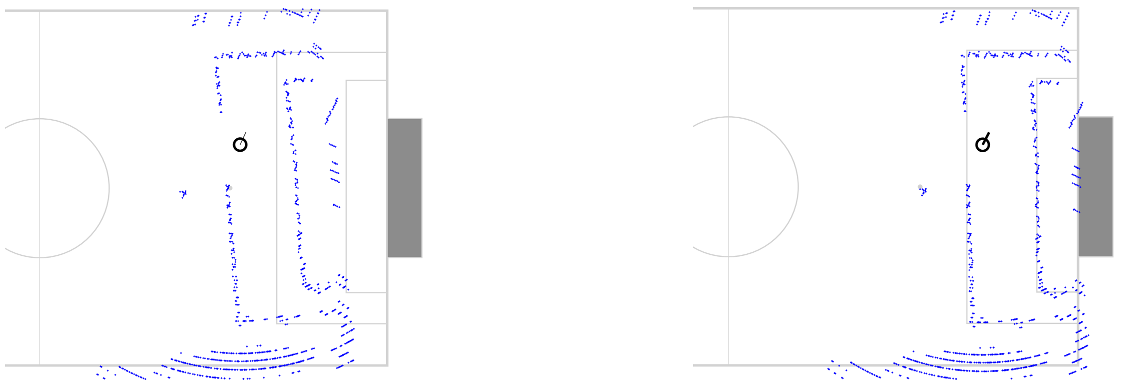

Fig. 3. Line points extracted from the omnidirectional camera system transformed to the preliminary pose estimation $\widehat{p}$ visualized as black circle (left). The same line points transformed to the final pose estimation p (right).

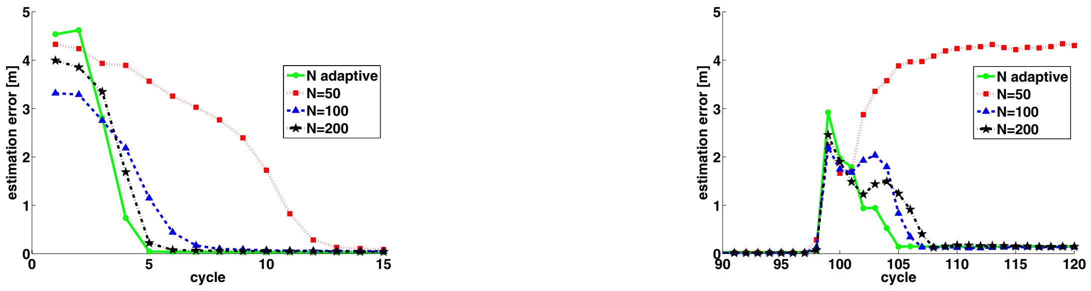

Fig. 4. Comparison of our algorithm to MCL with fixed numbers of samples.

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. 4 shows the estimation error of the algorithm with different numbers of samples. Independent of N the algorithm computes a very good pose estimation after at most 15 cycles. The number of samples in the adaptive method drops from $N = N_{\text{max}} = 200$ to $N = 1$ in 6 cycles. From cycle no. 20 to no. 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 algorithm immediately generates $N = N_{\text{max}}$ samples again, reacting to the high estimation error. Apart from the algorithm with $N = 50$ samples all methods generate a good pose estimation after at most 10 cycles again, 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 our algorithm finds a good pose estimation a few cycles earlier than the ones with a fixed number of samples $N = 100$ and $N = 200$ it performes much better than the others if the computation time is considered. Table II lists the mean computation time and the mean estimation error.

	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

TABLE II

RESULTS OF 196 CYCLES USING DIFFERENT NUMBERS OF PARTICLES N.

With the third experiment we show that the method is also able to correctly track the pose of a moving robot. As a ground truth for the experiment we placed a laser scanner at pose (-1.3,3.0,0.0), extracted the object data from the scan and recorded the position of the object. In order to get comparable position data the clock of the host computer running the laser scanner software was synchronized with the clock of the robot running the localization algorithm, and both programs stored a time stamp with their data. As the programs themselves were not synchronized, the laser measurements were interpolated to the time stamps of the localization cycles. The robot was then manually controlled around the field in two different speeds. 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 differential drive robot with constantly changing orientation in the first run and then the orientation was nearly fix while the direction of movement constantly changed in the second run. Fig. 5 (left) shows the results of the localization algorithm for the first run. In the beginning the samples are initialized randomly on the field and thus, the weighted mean over all samples results in a

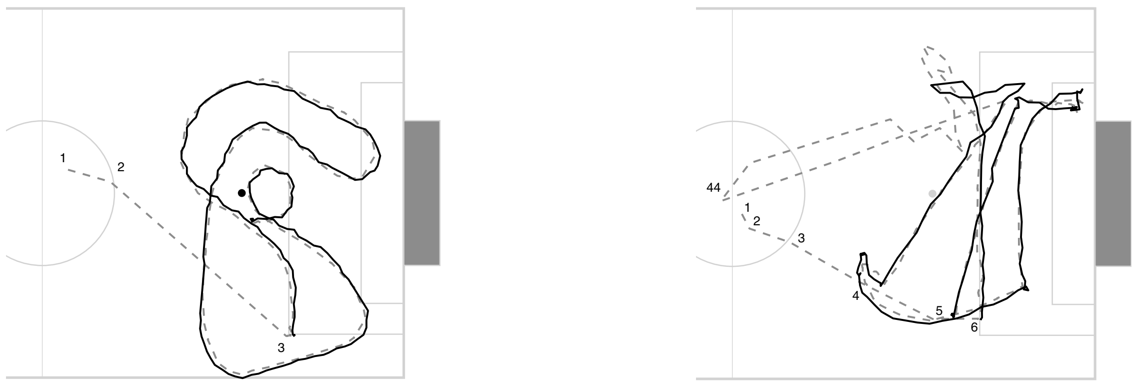

Fig. 5. This figure compares the position estimates of the proposed algorithm (dashed grey line) to the ground truth (solid black line) from a laser scanner. The robot was controlled at two different speeds of approx. 1 m/s (left) and 2 m/s (right). The algorithm needs up to 6 cycles to find the initial pose and then correctly tracks the robot around the field. Only with 2 m/s the algorithm fails to track the robot's position once, but relocalizes again only a few cycles later.

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, compared to the laser measurements, 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 (cf. rigth part of Fig. 5) the algorithm needed 6 cycles to correctly estimate the starting position. The estimated position then follows the real position along the penalty area until the robot changes its direction very quickly two times in a row. Here the algorithm looses the track of the robot and needs to reinitialize with 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 with only one sample. Nevertheless, this experiment shows the smooth transition between the local tracking mode and the global localization.

V. Conclusion and Future Work

This paper presented an efficient combination of global Monte-Carlo localization with an adaptive number of samples and local position tracking. The algorithm used a look-up table for a fast estimation of the samples' fitness and a local search for iterative improvement of the estimated pose. With this improvement it was possible to reduce the number of samples down 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.

The future work on this algorithm will include the application of the localization algorithm to other domains like office buildings. We expect that the combination of global localization and local optimization of the pose estimation is suitable for those environments, too.

References

[1] F. de Jong, J. Caarls, R. Bartelds, and P. Jonker. A Two-Tiered Approach to Self-Localization. In *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *LNCS*, pages 405–410. Springer, 2002.

[2] S. Enderle, M. Ritter, D. Fox, S. Sablatnög, G. Kraetzschmar, and G. Palm. Vision-based Localization in RoboCup Environments. In *RoboCup 2000: Robot Soccer World Cup IV*, volume 2019 of *LNCS*, pages 291–296. Springer, 2001.

[3] D. Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, University of Bonn, Germany, 1998.

[4] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the National Conference on Artificial Intelligence*, pages 343–349, 1999.

[5] J. Gutmann, T. Weigel, and B. Nebel. 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.

[6] P. Heinemann, T. Rückstieß, and A. Zell. Fast and Accurate Environment Modelling using Omnidirectional Vision. In *Dynamic Perception 2004*. Infix, 2004.

[7] L. Iocchi and D. Nardi. Self-Localization in the RoboCup Environment. In *RoboCup-99: Robot Soccer World Cup III*, volume 1856 of *LNCS*, pages 318–330. Springer, 2000.

[8] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The Robot World Cup Initiative. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM Press, 1997.

[9] C. Marques and P. Lima. A Localization Method for a Soccer Robot Using a Vision-Based Omni-Directional Sensor. In *Proceedings of EuRoboCup Workshop 2000*, 2000.

[10] E. Menegatti, A. Pretto, and E. Pagello. A New Omnidirectional Vision Sensor for Monte-Carlo Localization. In *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *LNCS*, pages 97–109. Springer, 2005.

[11] M. Plagge, R. Günther, J. Ihlenburg, D. Jung, and A. Zell. The Attempto RoboCup Robot Team. In *RoboCup-99: Robot Soccer World Cup III*, volume 1856 of *LNCS*, pages 424–433. Springer, 2000.

[12] T. Röfer and M. Jüngel. Vision-Based Fast and Reactive Monte-Carlo Localization. In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 856–861, 2003.

[13] T. Röfer and M. Jüngel. Fast and Robust Edge-Based Localization in the Sony Four-Legged Robot League. In *RoboCup-2003: Robot Soccer World Cup VII*, volume 3020 of *LNCS*, pages 262–273. Springer, 2004.

[14] H. Utz, A. Neubeck, G. Mayer, and G. Kraetzschmar. Improving Vision-Based Self-localization. In *RoboCup 2002: Robot Soccer World Cup VI*, volume 2752 of *LNCS*, pages 25–40. Springer, 2003.

[15] F. von Hundelshausen, M. Schreiber, F. Wiesel, A. Liers, and R. Rojas. MATRIX: A force field pattern matching method for mobile robots. Technical Report B-08-03, Free University of Berlin, 2003.