Calculating the Perfect Match: an Efficient and Accurate Approach for Robot Self-Localization

Martin Lauer, Sascha Lange, and Martin Riedmiller

University of Osnabrück, Institute of Cognitive Science and
Institute of Computer Science, 49069 Osnabrück, Germany
{martin.lauer,sascha.lange,martin.riedmiller}@uos.de

Abstract. The paper develops a new approach for robot self-localization in the Robocup Midsize league. The approach is based on modeling the quality of an estimate using an error term and numerically minimizing it. Furthermore, we derive the reliability of the estimate analyzing the error function and apply the derived uncertainty value to a sensor integration process. The approach is characterized by high precision, robustness and computational efficiency.

1 Introduction

Autonomous robots need to know their position and heading to be able to solve a given task like driving to a certain position. Especially in the Robocup Midsize league reliable position estimates are essential for higher level behavior like path planning, strategy and multi-agent coordination. Since autonomous robots cannot refer to global sensors which are fixed with respect to a global coordinate system but all sensors are on-board they need a procedure of self-localization, i.e. an algorithm to calculate their position and heading.

In this paper, we focus on a camera-based self-localization approach for the Robocup Midsize league. Three main difficulties have to be faced: (a) the self-localization process must be robust. The soccer field is not encircled by a board that allows to distinguish objects inside and outside the field like spectators. This may lead to misinterpretations of the visual information.

(b) Position estimates must be accurate: images of standard camera systems exhibit a poor resolution of objects located more than a few meters away. Hence, distance estimates are very noisy. Furthermore, the dynamics of the game with large accelerations and collisions between robots leads to vibrations that further affect the quality of self-localization (see fig. 1).

(c) The self-localization approach needs to be computationally efficient since the robot control program must satisfy strict real time conditions: our goal is to reduce the computation time to less than 15 milliseconds.

Mainly three approaches [10] have been used so far to solve the self-localization task: (a) the use of colored landmarks combined with geometrical calculation, e.g. [4], (b) the detection of white field markings combined with a Hough-transform, e.g. [6], and (c) the detection of landmarks or field markings combined with a sequential importance sampling [3] approach like *Particle filtering* [2].

All of these approaches have some merits but none of them solves all objectives of self-localization satisfactorily: approaches using landmarks are easily mislead by colored objects outside the field. Secondly, the large size of the field ($8 \times 16m$) and the small number of landmarks restricts the use of landmarks.

Using the Hough-transform needs the calculation of a three dimensional accumulator array. Hence, a lot of calculation is done for positions of no interest and any increase in precision implicates a heavy increase of computation time.

Monte Carlo approaches like Particle filtering also spend a lot of time in evaluating positions of no interest since they follow a blind search paradigm. In experiments we made with a Particle filtering approach of our Robocup team [7] we observed that approximately 98% of the examined positions did not contribute to the final position estimate since positions are evaluated even if neighboring places have already been evaluated as poor estimates.

We therefore want to propose a new algorithm for robot self-localization that overcomes the problems stated before and that fulfills all requirements: robustness, accuracy and efficiency. It is based on guided update steps modeling the localization problem as an error minimization task and using an efficient numerical minimizer. Additionally, we derive a measure of reliability of the calculated position analyzing the structure of the error function so that we can apply a stochastic sensor fusion process that increases the accuracy of the estimate.

An extension of the algorithm described in a further section also allows to solve the global localization problem, i.e. to find a robot's position without any prior knowledge. Finally, we compare the new approach with an existing implementation of Particle filtering for self-localization.

We assume the robots being equipped with an omnidirectional color camera on top that perpetually takes pictures from the field area around the robot. We further assume the case of omnidirectional driving capabilities although the calculations can also be done for a differential drive in simplified form.

2 Interpreting the Pictures from the Camera

2.1 Image Preprocessing

The pictures from the camera (fig. 1) are preprocessed using a detector of line points based on an efficient search along pre-defined radial scanlines. A description of this approach can be found in [7]. The result of preprocessing is a list of positions relative to the robot position and robot heading where scanlines intersect white field markings that have been observed in the image. In the following, we will call these points *detected line points* or, simply, *line points*.

An example of such a list is given by the gray circles in figure 4 (left). The detected line points are not linked, i.e. the list does not preserve the neighborhood relationship of line points that belong to the same line.

2.2 Matching Visible Information with Position Estimates

To find the position and heading of the robot with respect to the information we get from image preprocessing we define an error function that describes the

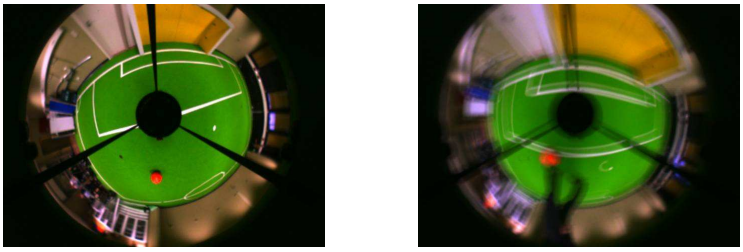

Fig. 1. Pictures taken by the omnidirectional camera. Left: a neat picture taken when the robot was not moving. Right: a blurred picture affected by vibrations when the robot was moving. All field markings are blurred and some of them occur twice.

fitness of a certain estimate. The idea is, that the detected line points and the known field markings match as best as possible if we assume the true robot position and robot heading. Hence, maximizing the fitness (=minimizing the error) yields the best estimate.

Let $(\boldsymbol{p}, \phi)$ be a pair of a possible robot position $\boldsymbol{p} = (p_x, p_y)$ and heading ϕ in a global coordinate system. The list of detected line points is given relative to the robot's pose as vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n$ (see fig. 2). Its position in world coordinates therefore is given by $\boldsymbol{p} + \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \boldsymbol{s_i}$. Minimizing the error between detected line points and true field markings means to solve:

$$\underset{\boldsymbol{p},\phi}{\mathit{minimize}} \quad E := \sum_{i=1}^{n} err(d(\boldsymbol{p} + \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \boldsymbol{s_i})) \tag{1}$$

The mapping $d(\cdot)$ gives the distance from a certain point on the field to the closest field marking. It is continuous and piecewise differentiable and can be calculated from the knowledge of the field markings that are defined in the Robocup rules.

err is an error function that punishes deviations between detected line points and the model lines. The squared error function $e \mapsto \frac{1}{2}e^2$ which is standard for many applications is not appropriate for the given task since it is not robust with

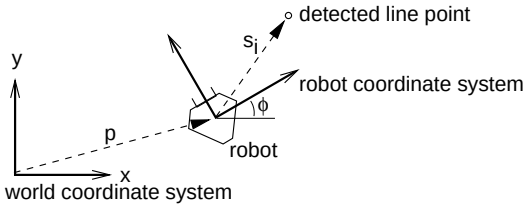

Fig. 2. Sketch of the fixed world coordinate system, the robot relative coordinate system and a vector s_i pointing to a detected line point.

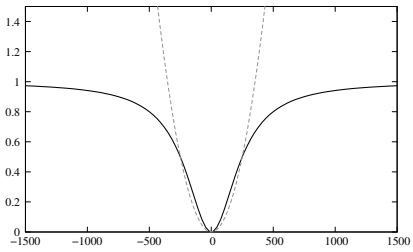

Fig. 3. Comparison of the squared error function (dashed line) and the more robust M-estimator $e \mapsto 1 - \frac{c^2}{c^2+e^2}$ (solid line)

respect to outliers [9]. Due to image noise and imperfect image preprocessing we are faced with a substantial amount of erroneously detected line points that would distort the estimate. Instead, we use the error function $e \mapsto 1 - \frac{c^2}{c^2+e^2}$ with parameter $c \approx 250$, see fig. 3. This error function is very similar to the squared error function for errors $e \leq c$ and is bounded above by a constant for larger errors, thus the influence of outliers onto the estimate is bounded.

Figure 4 (right) shows the error function for a certain set of detected line points. Obviously, the error function exhibits a large number of local minima. Due to the non-linearity of the minimization problem (1) we cannot analytically calculate its solution but we need a numerical minimizer.

Since d is almost everywhere differentiable we can build its gradient almost everywhere and interpolate the gradient at the non-differentiable places. Hence, we can use gradient descent to solve (1). Due to quick convergence and high robustness we use 10 iterations of *RPROP* to solve the minimization task [8]. *RPROP* was originally developed as learning rule for multi layer perceptrons but it can also be used to solve other types of unconstrained optimization problems.

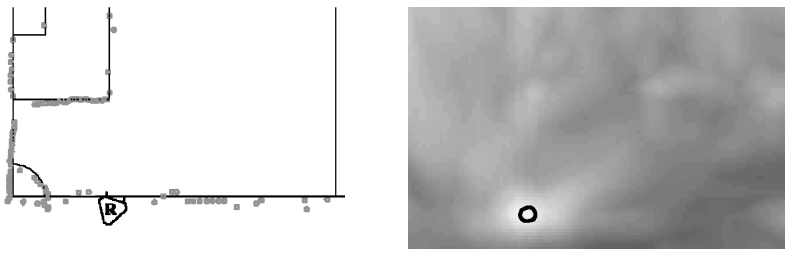

Fig. 4. Left: The set of line points (gray circles) and the field markings (solid lines) for an optimal position estimate. The estimated robot position and heading is indicated by the symbol "R". Right: A graylevel plot of the error function for the same clipping as in the left-hand figure. The 3D-error function was projected onto the two-dimensional field assuming optimal heading of the robot. Dark areas indicate positions with large error, bright areas positions with small error. The black circle indicates the optimal position estimate. The error function exhibits a distinctive global minimum.

Using the idea of error minimization we can realize the draft version of a robot localization algorithm:

1. start with a known position estimate
2. calculate the movement of the robot since the latest update of the position estimate and add it to the latest estimate
3. optimize the position applying the error minimization approach
4. repeat 2. and 3. every time a new camera image is received

2.3 Dealing with the Aperture Problem

Figure 4 shows a situation with a distinctive global minimum, i.e. the optimization task is well-posed and all parameters can be estimated reliably. Unfortunately, situations occur in which the optimization task is ill-posed due to a small number of line points or a poor structure of the line points. Such a situation is depicted in figure 5: the robot is located next to the touch-line of the field and all line points refer to the touch-line. Hence, the distance to the touch-line, i.e. the y-coordinate, can be estimated very reliably while the x-coordinate remains vague. The error function is characterized by a long valley of similar small values.

To tackle the aperture problem we have to recognize three possible situations: (a) the error function exhibits a distinctive global minimum. Hence, we can estimate $\boldsymbol{p}$ and ϕ reliably. (b) The error function is completely flat due to a small number of line points. Thus, we cannot estimate any parameter. (c) The error function exhibits a valley structure around the minimum. Here, we can estimate parameters robustly which refer to a coordinate axis orthogonal to the valley but we cannot estimate parameters that refer to a coordinate axis parallel to the valley. E. g. in fig. 5 we can estimate p_y and ϕ but not p_x.

To determine the structure of the error function around the minimum we propose the analysis of the second order derivatives of the error function: the value of $\frac{\partial^2 E}{(\partial p_x)^2}$ is small in the case of a valley parallel to the x-axis and in a

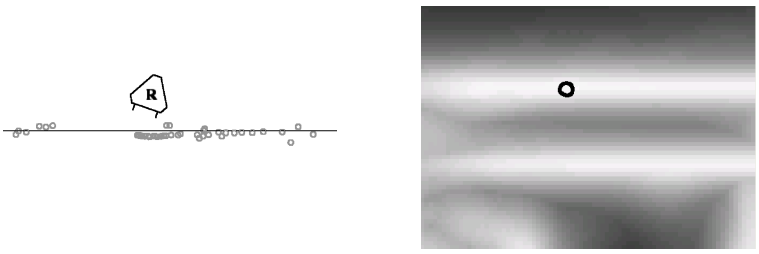

Fig. 5. Aperture problem: The left hand figure shows the mapping of the line points onto the known field markings while the right hand figure shows the error function in the same way as in figure 4. The error function shows a valley of small error values. Hence, the position estimate is very reliable with respect to the y-coordinate but unreliable with respect to the x-coordinate.

completely flat case while it is large if E shows a distinctive minimum with respect to the x-axis. Analogously we can analyze $\frac{\partial^2 E}{(\partial p_y)^2}$ and $\frac{\partial^2 E}{(\partial \phi)^2}$.

From a practical point of view the calculation of the second order derivatives simplifies: The function d is build out of the line markings on the field. Most of them are lines parallel to the coordinate axis, only the corner arcs and the center circle are no straight geometrical objects. Hence, the function d is piecewise linear in most parts of the field and its second order derivatives are zero in these areas. Using this simplification we get:

$$\frac{\partial^2 E}{(\partial p_x)^2} \approx \sum_{i=1}^{n} err''(s_i) \cdot (\frac{\partial d(s_i)}{\partial x})^2 \tag{2}$$

$$\frac{\partial^2 E}{(\partial p_y)^2} \approx \sum_{i=1}^{n} err''(s_i) \cdot (\frac{\partial d(s_i)}{\partial y})^2 \tag{3}$$

$$\begin{aligned}\frac{\partial^2 E}{(\partial \phi)^2} \approx \sum_{i=1}^{n} \Big(& err''(s_i) \cdot \big(\frac{\partial d(s_i)}{\partial x}(-\sin\phi \ -\cos\phi)\boldsymbol{s}_i + \frac{\partial d(s_i)}{\partial y}(\cos\phi \ -\sin\phi)\boldsymbol{s}_i\big)^2 \\ & + err' \cdot \big(\frac{\partial d(s_i)}{\partial x}(-\cos\phi \ \sin\phi)\boldsymbol{s}_i + \frac{\partial d(s_i)}{\partial y}(-\sin\phi \ -\cos\phi)\boldsymbol{s}_i\big)\Big)\end{aligned} \tag{4}$$

where err' and err'' denote the first and second order derivative of err.

Unfortunately, the error function $e \mapsto 1 - \frac{c^2}{c^2+e^2}$ used in (1) is not completely positive definite and therefore the curvature criterion may be mislead, i.e. the second order partial derivative may be small also the minimum is distinctive. However, this problem is caused only from outlying observations since the error function is positive definite in the interval $(-\frac{c}{\sqrt{3}}, \frac{c}{\sqrt{3}})$. To avoid this problem we adopt the following artifice: in (2)–(4) we replace the original error function by the squared error function $e \mapsto \frac{1}{2}(\frac{e}{c})^2$ and ignore outlying observations. Hence, E becomes positive definite and the curvature criterion yields sound results.

3 Tracking and Smoothing

The approach described so far estimates the robot position and heading that matches optimally to the information extracted from the camera image. Due to vibrations of the robot, especially in the case of high velocity or due to collisions, this position is affected by a reasonable amount of noise and inaccuracy (see fig. 1 (right)). The dotted line in figure 6 (left) shows an example of a trajectory build out of the positions calculated only from the image information. Obviously, the noise of the self-localization process is severe.

To reduce the noise we propose to evaluate the temporal dependency of positions estimated from subsequent images. Since subsequent positions of the robot are neighbored and linked using some transition depending on the robot velocity we can use a stochastic weighted averaging approach that is in fact a simplified application of the Kalman filter (see e.g. [5]).

We thereto enclose all estimates with variances that model the degree of uncertainty. We don't use covariances to simplify the modeling. Let denote $(\boldsymbol{r}_t, \psi_t)$

the estimate of the robot's position and heading at time t and $\sigma^2_{r_x,t}$, $\sigma^2_{r_y,t}$, $\sigma^2_{\psi,t}$ the respective variances. After a robot movement the position estimate and variances are updated using the motion model of an omnidirectional robot with velocity $\boldsymbol{v}$ and rotational velocity[1] ω:

$$\hat{\psi}_{t+\tau} = \psi_t + \omega \cdot \tau \tag{5}$$

$$\hat{\boldsymbol{r}}_{t+\tau} = \begin{cases} \boldsymbol{r}_t + \boldsymbol{v} \cdot \tau & \text{if } \omega = 0 \\ \boldsymbol{r}_t + \frac{1}{\omega} R_{\psi_t} \begin{pmatrix} \sin(\omega\tau) & \cos(\omega\tau)-1 \\ 1-\cos(\omega\tau) & \sin(\omega\tau) \end{pmatrix} R_{-\psi_t} \boldsymbol{v} & \text{if } \omega \neq 0 \end{cases} \tag{6}$$

with R_ψ denoting the rotation matrix by the angle ψ. The velocity and rotational velocity is measured by odometers. The update of the variances takes into account the inaccuracy of the movement:

$$\sigma^2_{\hat{\psi},t+\tau} = \sigma^2_{\psi,t} + \alpha(\hat{\psi}_{t+\tau} - \psi_t)^2 \tag{7}$$

$$\sigma^2_{\hat{r}_x,t+\tau} = \sigma^2_{r_x,t} + \alpha(\hat{r}_{x,t+\tau} - r_{x,t})^2 \tag{8}$$

$$\sigma^2_{\hat{r}_y,t+\tau} = \sigma^2_{r_y,t} + \alpha(\hat{r}_{y,t+\tau} - r_{y,t})^2 \tag{9}$$

The parameter $\alpha > 0$ controls the assumed accuracy of the movement.

In (8) and (9) we ignore the non-linear dependency between rotational and translational movements of a robot. Ignoring it keeps the statistical modeling efficiently tractable while considering the dependency would require time-consuming statistical techniques like e.g. sequential importance sampling. As long as the frequency of updates remains high, the additional error made by the assumption of independence remains small.

After receiving an image from the camera and calculating the optimal estimate with respect to the image information $(\boldsymbol{p}, \phi)$ we are able to calculate a smoothed position estimate combining $(\boldsymbol{p}, \phi)$ and $(\boldsymbol{r}, \psi)$. Therefore we introduce variances for $(\boldsymbol{p}, \phi)$ that model the uncertainty of the image-based estimator.

The reliability of the image-based estimator is influenced by several aspects: the precision of the optical system, mechanical vibrations, camera calibration errors, the accuracy of image preprocessing and the structure of detected line points. While we can only make crude assumptions about the accuracy of the former aspects we need to model the latter aspect, i.e. the structure of line points, carefully to avoid erroneous estimates.

In section 2.3 we discussed the aperture problem recognizing that the estimate may be reliable in some parameters and unreliable in others. This means, the assumption of uncertainty is different for each of the parameters p_x, p_y and ϕ. We therefore propose to use the curvature criterion to individually determine the variance of each parameter: a small second order partial derivative should be related to a large variance while a large second order partial derivative should be related to a small variance. We use a heuristic function to map second order partial derivatives onto variances. It was determined from a set of experiments by visual inspection.

[1] $\boldsymbol{v}$ and ω refer to the global coordinate system and describe the movement at time t

The sensor fusion step consists of averaging two independent Gaussian distributions. Denoting with σ_ϕ^2 the variance of ϕ we get:

$$\psi_{t+\tau} = \frac{\sigma_\phi^2 \hat{\psi}_{t+\tau} + \sigma_{\hat{\psi},t+\tau}^2 \phi}{\sigma_\phi^2 + \sigma_{\hat{\psi},t+\tau}^2} \tag{10}$$

$$\sigma_{\psi,t+\tau}^2 = \frac{\sigma_\phi^2 \cdot \sigma_{\hat{\psi},t+\tau}^2}{\sigma_\phi^2 + \sigma_{\hat{\psi},t+\tau}^2} \tag{11}$$

The sensor fusion steps for $\boldsymbol{r}_{t+\tau}$ can be calculated analogously.

Using the filtered estimates helps to improve both, robustness and precision. A single misleading camera image does not lead any more to a loss of track since the filter does not allow to jump to a completely different position which would be possible using the simple algorithm shown in section 2.2.

Additionally, the aperture problem is tackled appropriately: even if the image-based estimate is unreliable with respect to some coordinate axis sensor fusion leads to reliable estimates for all parameters. Moreover, the precision is increased by the sensor fusion due to its implicit smoothing. Hence, erroneous image information do not have a strong impact on the final estimate.

4 Solving the Global Localization Problem

Section 3 discussed the problem of tracking a robot's position starting with a known initial position, i.e. to look for the locally optimal estimate. To solve the global localization problem means to find the global minimum of the error E from (1) even if no initial estimate is available. Certainly, it is not possible to solve the global minimum search under hard real time constraint every cycle.

We therefore propose to apply the tracking approach several times in parallel with random initial positions. The estimates converge very quickly to the next local minima of E. Hence, we can compare the different estimates and choose the best one as our main estimate while the sub-optimal estimates remain under inspection as possible alternatives. By repeated random reinitialization of the alternative estimates we successively scan the whole parameter space.

To avoid premature switches between main estimate and an alternative due to random effects we introduced a discounted scoring system: the best estimate in a cycle gets one point while the others don't get any point. By comparing the discounted sums of points we can evaluate which of the estimates is the overall best one over a longer period of time. Switching the main estimate to an alternative happens only if the score of the alternative becomes greater than the score of the current main estimate.

5 Experiments and Comparison

5.1 Accuracy

All experiments explained here were made on the robots of the *Brainstormers Tribots* team. The algorithms were implemented in C++ under Linux and ran

on JVC sub-notebooks with 1GHz Pentium processor. We used a test field which had the dimensions of the fields used in the Robocup 2004 competition.

We made experiments using the joystick to control the robot. In all cases the self-localization approach worked fine. Figure 6 (left) shows a trajectory of self-localization positions that were calculated while the robot drove on a curved trajectory of $6m$ length. The trajectory based on the Kalman filtered positions is very smooth. In contrast, the trajectory build out of the image-based estimates without Kalman filtering exhibits deviations up to $43cm$ and erratic outliers orthogonal to the direction of movement.

In figure 6 (right) we repeated the same experiment with a Particle filtering based self-localization [7]. Both approaches worked on exactly the same input so that they can be compared directly. Obviously, the Particle filter exhibits large deviations from the curved trajectory that are even worse than the deviations of the image-based trajectory in figure 6 (left). These examples show the high precision of the new approach that clearly outperforms the hitherto used Particle filter. In further experiments these results have been confirmed.

5.2 Computational efficiency

We also measured the computation time and compared it to the Particle filter with 200 and 500 particles. The average computation time is given in table 1. The Particle filter needed four times (with 200 particles) and ten times (with 500 particles) as much computation time as the error minimizing algorithm. In the given framework with hard real-time constraints this saving of time allowed us to increase the number of camera images analyzed per second to 30 which was far not possible with the Particle filtering approach.

Figure 7 shows the cumulative distribution function of the time needed by the error minimizing self-localization algorithm. The computation time linearly depends on the number of line points. It varied between none and 300 per cycle. The maximal computation time was $11ms$ while the average was $4.2ms$. One

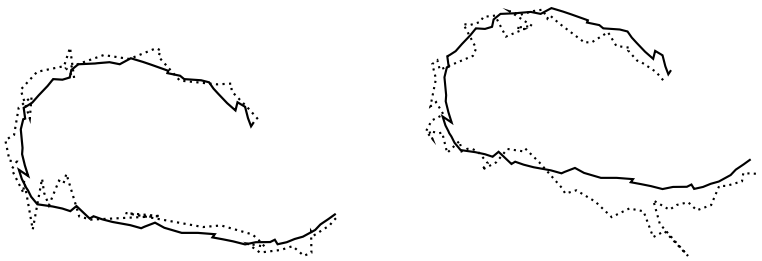

Fig. 6. Left: example of a robot driving with $2\frac{m}{s}$ on a curved trajectory of $6m$ length. The dotted line shows the trajectory of positions which are evaluated optimal considering the camera image. The solid line shows the smoothed trajectory using the Kalman filter. Right: comparison of the error minimizing approach (solid line) with the Particle filter with 500 particles (dotted line) on the same trajectory.

Table 1. Computation time of different methods for self-localization in milliseconds

Approach	Average computation time per cycle
Particle filter with 500 particles	48.3
Particle filter with 200 particles	17.9
Error minimizing self-localization	4.2

possibility of restricting the maximal computation time is therefore to restrict the number of line points used. E.g. restricting the number of line points to 100 (50) yields a maximal computation time of $6ms$ ($4ms$) without reducing the accuracy of estimates considerably.

5.3 Global Localization

In a third experimental setup we measured the time that was needed to globally localize the robot. Thereto we repeatedly activated a random reset of self-localization and measured the time needed to find the robot's position again.

The global localization approach used one main estimate and three alternative estimates which were repeatedly reinitialized after between $100ms$ and $2000ms$, depending on their quality. Although in every iteration only four positions were evaluated the self-localization found the global optimum in most cases quickly. On average, it took 2.9 seconds. The time needed for global localization heavily depended on the number of line points and their structure: it was far easier to localize in the penalty area in front of a goal with lots of horizontal and vertical field markings than next to the touch line where only a few line points of only a single field marking could been detected.

6 Related Work

There are two different approaches that are closely related to the error minimizing approach: the two-step approach of Cox [1] and the so-called *MATRIX*-

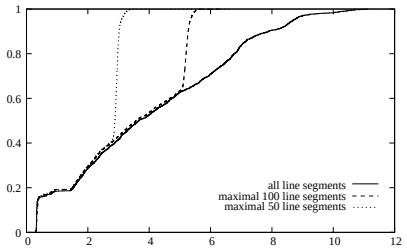

Fig. 7. Cumulative distribution function of the computation time per cycle of self localization in milliseconds (x-axis). The solid line refers to the case of unlimited number of line points while the dashed (dotted) line shows the case of a maximum of 100 (50) line points per cycle.

Table 2. Comparison of three approaches for robot self-localization

	Cox	MATRIX	Error minimizer
sensory system	range finder/walls	omnidirectional camera/field markers	omnidirectional camera/field markers
principle	2-step	force-field	gradient descend
error function	squared error	≈squared error	M-estimator
deals with outliers	remove in advance	weighted observations	M-estimator
optimizer	analytically/2-step	ad hoc	RPROP
variance estimation	analytically, only for straight lines	none	analyzing the Hessian
sensor integration	fusion of Gaussians	none	fusion of Gaussians
global localization	none	exhaustive search at beginning	randomized parallel search
experiments	$\frac{1}{40}\frac{m}{s}$ robot velocity, frame rate of $\frac{1}{8}Hz$	none	$3\frac{m}{s}$ robot velocity, frame rate of $30Hz$

approach [11]. The work of Cox uses a range finder to detect walls instead of field markers. Self-localization is based on assigning every observed wall point to the closest true wall and minimizing the squared error. Since this approach does not consider curved walls solving the optimization problem can be done analytically. Experiments are presented only for very slowly moving robots.

The *MATRIX*-approach models the task using an artificial force field which resembles a gradient vector field for the squared error measure. This approach does not consider the aperture problem. Experimental results are missing.

Comparing the error minimizing approach with both alternatives (see tab. 2) shows that the new approach completes its alternatives: in contrast to *MATRIX* it tackles the aperture problem and allows sensor integration while in contrast to Cox' approach it can even deal with noisier sensors like optical systems and with higher robot velocities which cause slippage and imprecise odometer signals.

7 Summary

We proposed a new approach to efficiently solve the robot self-localization problem in the Robocup Midsize league. The approach is based on an efficient numerical approach to find the locally best match between the camera image and the model of the field. Additionally, a stochastic sensor fusion step similar to the Kalman filter is used to link the position estimates calculated from subsequent images and to smooth the trajectory.

This approach enables a low-noise tracking of a robot's position. Experiments comparing the new approach with an existing Particle filtering approach point out the immense noise reduction. Position estimates become more reliable and more precise. Hence, they are much better suited for further calculations like path planning. Thus, using the error minimizing self-localization enables the development of higher level capabilities of robot control and team play.

While an increase in accuracy implicates an increase of computation time using methods like Particle filtering and Hough transform the new approach is very efficient. In experiments we could show that it needs only a tenth of the computation time of a Particle filter. By restricting the number of line points it was possible to restrict the maximal computation time to 4 milliseconds. Hence, the new approach can be used even under hard real time constraints.

Although the basic modeling is not guaranteed to find the overall optimal position we proposed an extension that also solves the global localization problem. We could show by experiments that the robot found its position on average in only 2.9 seconds. Hence, even if the tracking approach is mislead by heavily erroneous sensor information the robot is able to quickly find its position again.

The error minimizing algorithm for self-localization that is presented in this paper is characterized by three properties: high accuracy, robustness and efficiency. It outperforms Particle filtering in all of these aspects. It therefore is a step towards a completely autonomous and robust soccer playing robot.

Acknowledgments. This work was supported by the German Research Foundation (DFG) SPP 1125.

References

1. Ingemar J. Cox. Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.
2. Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte Carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, 1999.
3. Arnaud Doucet. On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR.310, University of Cambridge, 1998.
4. Hikari Fujii, Yusuke Ohde, Masayuki Kato, Fumitaka Otsuka, Naoka Sema, Marie Kawashima, Eisuke Sugiyama, Shuichi Niizuma, and Kazuo Yosida. EIGEN team description. In *Robocup-2004*, 2004.
5. Arthur Gelb, editor. *Applied Optimal Estimation*. Cambridge, 1974.
6. Luca Iocchi and Daniele Nardi. Self-localization in the robocup environment. In *Proceedings of the 3rd Robocup Workshop*, pages 116–120, 1999.
7. Artur Merke, Stefan Welker, and Martin Riedmiller. Line based robot localization under natural light conditions. In *ECAI 2004 Workshop on Agents in Dynamic and Real Time Environments*, 2004.
8. Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591, 1993.
9. Werner Stahel. Robust alternatives to least squares. Technical Report 78, ETHZ Seminar für Statistik, 1996.
10. Hans Utz, Alexander Neubeck, Gerd Mayer, and Gerhard Kraetzmar. Improving vision-based self-localization. In *Robocup-2001*, 2001.
11. Felix von Hundelshausen, Michael Schreiber, Fabian Wiesel, Achim Liers, and Raúl Rojas. MATRIX: A force field pattern matching method for mobile robots. Technical Report B-09-03, FU Berlin, 2003.