Comparing Sensor Fusion Techniques for Ball Position Estimation

Alexander Ferrein[1], Lutz Hermanns[2], and Gerhard Lakemeyer[1]

[1] Knowledge-Based Systems Group
Computer Science Department
RWTH Aachen
Aachen, Germany
{ferrein, gerhard}@cs.rwth-aachen.de
[2] SMA Technologie AG
Niestetal, Germany
lutz.hermanns@sma.de

Abstract. In robotic soccer a good ball position estimate is essential for successful play. Given the uncertainties in the perception of each individual robot, merging the local perceptions of the robots into a global ball estimate often results in a more reliable estimate and helps to increase team performance. Robots can use the global ball position even if they themselves do not see the ball or they can use it to adjust their own perception faults. In this paper we report on our results of comparing state-of-the-art sensor fusion techniques like Kalman filters or the Monte Carlo approach in RoboCup's Middle-size league. We compare our results to previously published work from other Middle-size league teams and show how the quality of perceiving the ball position is increased.

1 Introduction

Estimating the ball position accurately and reliably is one of the central problems in robotic soccer (RoboCup, [18]), especially in the Middle-size league, where the robots often occlude the ball due to their size. Knowing where the ball is located is central for the cooperation and coordination task of the team. As the rules of the Middle-Size league allow for communication between the robots or to an external host global sensor fusion can be applied for calculating a global ball estimate. Merging the single estimates of the robots into a global ball position is one of the keys to robust team-play as the perception of a single robot might be wrong. With the knowledge of the team-mates the wrong estimate of a single robot can be adjusted. This leads to a better overall team performance.

In this paper we evaluate state-of-the-art sensor fusion techniques for merging the global ball position from the robot's local perceptions, filtering out wrong estimates and false positives. The methods comprise simple approaches based on averaging and more sophisticated ones like the Kalman filter, or the Monte Carlo approach. We tested the global sensor fusion with our team AllemaniACs during the World Cup 2003 in Padua, Italy, and 2004 in Lisbon, Portugal, and the

German Open 2004 in Paderborn observing a significant increase in the quality of the ball position estimate. Moreover, we use the global ball information to detect when a robot is dis-localized.

We show that the best methods yield better results than reported before in the literature for the RoboCup setting and that the one reported in [3] is outperformed by an even simpler one. The presented experiments were conducted in real game situations showing a significant increase in the availability and quality of ball position estimates.

The paper is organized as follows. In Section 2 we give a brief overview about the related work on sensor fusion in the robotic soccer domain. In Section 3 we describe our hardware platform and the software system. Section 4 describes the methods we use for merging the local robot perceptions to a global world model and show their results in Section 5. We conclude with Section 6.

2 Related Work

The soccer domain is an interesting domain for research on sensor fusion. Most of the work concentrates on merging perceptions of the ball to one consistent estimate. The methods commonly used are probabilistic method as Kalman filters [9] or Markov Localization [6]. Here, we will concentrate on the related work in the field of fusing ball estimates in the RoboCup domain. One can distinguish between the so-called *local sensor fusion* and *global sensor fusion*. In the former case the perceptions of several sensors on one robot are combined. The latter refers to merging the perceptions of different robots.

Dietl et al. [3] present a ball tracking algorithm, which combines a Kalman filter with Markov localization. They assign a new measurement to an existing track of observation by minimizing the sum of squared error distances. For predicting the ball position a Kalman filter is used. For this Kalman filter they use Markov localization as an observation filter. They report a mean error of 38 cm for a moving ball while the robots did not move.

Stroupe et al. [16] represent each ball estimate as a two-dimensional Gaussian in a canonical form. This allows to merge the single estimates of the robots simply by multiplying them. For predicting the ball position they use a Kalman filter approach.

Pinheiro and Lima [13] also represent sensor information about the ball as a Gaussian applying *Bayesian Sensor Fusion*. They assume that the last position is known and that the single estimates are close by each other.

The team *Mostly Harmless* [14] use local sensor fusion to integrate the perceptions from the different sensors their robots are equipped with. They use a Monte Carlo approach [2] to merge the data from the different sensors into a local world model. They also provide a merged global world model.

The *Milan RoboCup Team* use an *anchoring approach* [1]. The Sensor data are represented symbolically and are anchored with objects from the environment. For the sensor fusion of the symbolical sensor data they use fuzzy logics.

Several other teams participating at the world championships are using local sensor fusion.

3 The Platform

In this section we give a brief overview of the relevant parts of our hard and software system.

3.1 Hardware

The hardware platform is a self-development [20] with the aim of having robots which are competitive in RoboCup and can also be used in office domains for service robotics applications. The platform has a size of 39 cm × 39 cm × 40 cm (Fig. 1). For power supply we have two 12 V lead-gel accumulators with 15 Ah each on-board. The battery power lasts for approximately one hour at full charge. The robot has a differential drive, the motors have a total power of 2.4 kW. This power provides us with a top speed of 3 m/s and 1000 °/s by a total weight of approximately 60 kg.

Fig. 1. The hardware platform

On-board we have two Pentium III PC's at 933 MHz running Linux, one equipped with a frame-grabber for a Sony EVI-D100P camera mounted on a pan/tilt unit. Our other sensor is a 360° laser range finder from Sick Ibeo with a Gaussian error distribution and a deviation $\sigma \approx 3$ cm. It runs with a resolution of 1 degree at a frequency of 10 Hz. For communication a WLAN adapter using IEEE 802.11a is installed.

3.2 On-board Software

For our software architecture we are using a three layered architecture consisting of a low-level layer where sensory inputs and actuator outputs are controlled, a mid-level where modules like collision avoidance and localization are located and a high-level controller for making plans about future courses of actions. Important aspects for the process of merging local ball estimates are the *localization*, the *vision module* which provides the ball estimates, and the *world model* which communicates the local estimates to a global world model where the fusion process takes place.

For self-localization our software provides the *localize* module using the 360 ° laser range finder following a Monte Carlo approach. As for RoboCup it is very important to have continuity in the position updates we track the robot's position with odometry in between two position estimates from the Monte Carlo localization module. The localization uses KLD sampling and cluster-based sampling (e.g. cf. [4,11]). The position estimates provided by the localization have an accuracy of about ±2.5 cm, position losses are very rarely. For more information about the localization we confer to [15].

The ball, goals and flag-posts are localized with the *vision* module using color segmentation and knowledge about the objects' shape. Color segmentation is inexpensive, but the mapping between single objects and their colors or chrominances, if reduced to two dimensions, must be known in advance. For each shape detected object, we compute the chrominance histogram. The histograms are combined based on the Bayes Theorem to obtain a lookup table [7], which is used for color segmentation. To profit from the speed of color segmentation, shape detection is only applied to regions containing the interesting object's color. The circular ball is detected by a randomized hough transform. The goals and flag-posts are modeled as quadrilaterals, which are bordered by straight lines. Object detection takes about 50 ms on a Pentium III with 933 MHz. The entire vision system including the color segmentation and color recalibration step runs at 10 Hz.

The *world model* consists of information like the ball and player position as well as some other internal state information. We provide two kinds of world model, a local one which is constructed from the own perception and a global one which is the result of a world model fusion on an external computer.

For inter-process communication we use a blackboard communicating via shared memory between the two on-board computers and via UDP between the robots.

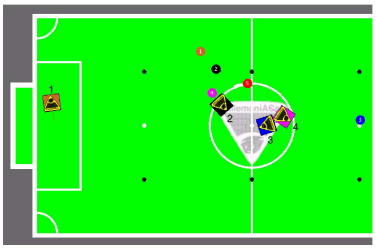
4 Sensor Fusion Techniques

In this section we give a brief overview of the sensor fusion techniques we use in this case study. We start with simple ones like mean methods and go over to more sophisticated ones like the Weight grid method, the Kalman filter, and the Monte Carlo method. Finally, we test a combination between the Weight grid and the Kalman filter. This method is similar to the one proposed in [3].

4.1 Arithmetic Mean

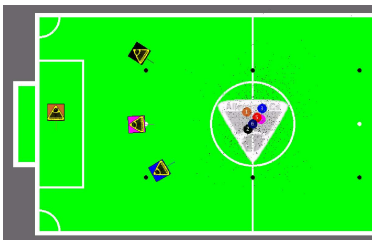
Each robot i seeing the ball contributes his local estimate $(lb_x, lb_y)^{(i)}$ about the ball position by communicating it to a central server. The global ball estimate (gb_x, gb_y) is calculated by averaging over the estimates from each robot, i.e. $gb_x = \frac{1}{n}\sum_{i=0}^{n} lb_x^{(i)}$, $gb_y = \frac{1}{n}\sum_{i=0}^{n} lb_y^{(i)}$.

Here, every local ball estimate has the same importance. The estimate of a robot which is far away from the ball should not be weighted as much as the

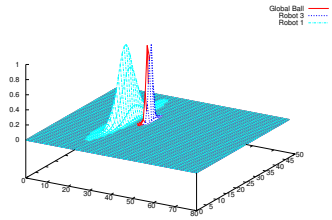


(a) Arithmetic mean

(b) Example situation for the weight grid fusion



(c) Monte Carlo ball localization



(d) The weight grid for the situation in Fig. 2(b)

Fig. 2. Fusion techniques

estimate of a robot being close to the ball. Therefore, in a variant we weight the estimates according to the distance from the robot to the ball and a time factor which denotes how long ago the robot has seen the ball for the last time:

$$w_i = \frac{1}{dist_i} \cdot {conf_b}^{(i)} \cdot {conf_p}^{(i)}. \tag{1}$$

$conf_b$ is the ball confidence provided by the vision system. The role of this confidence is to give some means of persistence to the ball estimate. If the ball is not seen in one frame it is not reasonable to assume that the ball disappeared from the previously detected position as the vision system has some detection error. The confidence is modeled by a rapidly decreasing function over the time, which is set to 1 if the ball is detected. This confidence helps stabilizing the local ball estimates. The other confidence $conf_p$ is provided by the localization system and represents the confidence that the robot is located at the given position. The weighted mean is then calculated as $gb_x = \frac{1}{\sum_{i=0}^{n} w_i} \sum_{i=0}^{n} w_i \cdot {lb_x}^{(i)}$ and $gb_y = \frac{1}{\sum_{i=0}^{n} w_i} \sum_{i=0}^{n} w_i \cdot {lb_y}^{(i)}$. An example of the weighted mean estimation is

depicted in Fig. 2(a). The local ball estimates are enumerated and marked with the respective robot's color. The merged estimate is numbered as estimate 5 (red ball).

4.2 Weight Grid

This method uses a grid to represent positions on the field. The representation is similar to an occupancy grid representation [12], but each cell can take weights greater than 1. For each ball estimate a 2-dimensional Gaussian distribution is calculated. The parameters for this Gaussian are taken from the *distance error* and the *pan error* which are provided by the vision system. Fig. 3 shows the parameters.

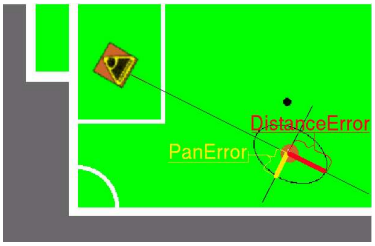


Fig. 3: Ball confidence

The cell update works as follows: each local ball estimate is weighted by the weight function given in the previous section (Eq. 1). These weights are then multiplied with the Gaussian distribution and the result is stored in the respective grid cells. Fig. 2(b) shows an example situation where the ball can be seen by the goal keeper and blue robot. For illustration purposes a camera picture from a ceiling camera is overlaid. The perception of the goal keeper (brown) is closest to the real ball position (dark blue). The ball is further seen by the blue robot. The merged ball position is depicted in red. In this case the distance weighting has negative effect to the global ball position as the wrong estimate of the blue robots has higher weights and drags the merged ball position into the wrong direction. The influence of the weighting function can also be seen in Fig 2(d). The distribution of the hypothesis of the goal keeper is wider (light blue). This resembles the distance weighting (Eq. 1) of the estimates. The weighted Gaussian of the global ball estimate is depicted in red.

4.3 Kalman

A Kalman filter [9] is used to estimate the state of a process variable $x \in \mathbb{R}^n$ in a dynamic system. As we want to estimate the position of the ball we have $x = (gb_x, gb_y)^T$. The basic equation describing the stochastic process is $x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$ with measurements $z_k = Hx_k + v_k$, where w_k and v_k represent the process and measurement noise, resp. They are assumed to be independent of each other and normally distributed, i.e. $p(w) \sim N(0, Q)$ and $p(v) \sim N(0, P)$.

The matrix A represents the motion model relating the old process state with the new one. In our case we do not integrate a motion model as the local ball measurements from all robots are from the same point in time. Instead we propagate the global ball position by the ball velocity each time the Kalman filter is called. Applying a motion model at this point would mean that the ball has moved during the integration of the respective ball measurements from the

robots. As the single measurements from the robots can be assumed to happen at the same time the movement of the ball can be taken into account only after the measurement update step from the Kalman filter. To integrate control inputs in the estimate the variable $B \cdot u$ is added to x_k. In our case $u = 0$. Therefore, the state equation results in

$$\begin{pmatrix} gb_{x,k} \\ gb_{y,k} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} gb_{x,k-1} \\ gb_{y,k-1} \end{pmatrix} + w_{k-1}$$

To integrate the sensor measurements z_k the matrix H is used denoting the observable components of x. In our case it is also the identity matrix as we can observe both coordinates of the ball. The process noise covariance matrix Q was empirically found as $Q = diag(3, 3)^T$, the measurement noise covariance matrix P is initially set to $P = diag(4, 4)^T$. For the first time when the filter is started we take the very first measurement from one robot as initial value for x_0. Applying the time update and measurements update equations (cf. eg. [10]) we estimate the global ball position from the robots' local estimates.

As another variant we use the *Kalman reset* filter. Here, the matrix P is reset to its initial value each time a global ball position is calculated, which in our case happens 10 times a second. This means that the actual measurement receives more attention.

4.4 Monte Carlo Localization

Monte Carlo Localization was introduced in [5] as an improvement of the Markov Localization [6]. Both techniques originally were developed for the self-localization of a robot.

The main idea of *Markov Localization* is to provide a probability distribution $Bel(l)$ over the space of possible positions. New sensor readings are used to re-calculate the distribution with the help of Bayes' rule. The distribution is represented in an occupancy grid.

The *Monte Carlo (Ball) Localization* represents the distribution $Bel(l)$ with a set of weighted samples instead of the grid which is used in the Markov localization. For the ball fusion we take samples that represent the ball position hypotheses. Therefore, a *sample* x_i consists of a possible ball position $l = \langle x, y \rangle$ and a weight w_i which is also called *importance factor*.

$$x_i = \langle \langle x, y \rangle, w_i \rangle$$

In Fig. 4 the Monte Carlo Algorithm is shown based on [19] using a set of samples X the ball velocity vel and a set of local ball estimates LB. In one iteration of the algorithm m samples from the sample set are drawn by chance according to their importance factor. A sample with a high weight is drawn more often than one with a lower weight. Additionally, some new samples are generated around new local ball estimates. Then, the algorithm works in two steps. First the motion model is applied to the drawn samples. This is done by promoting the samples according to the ball velocity and some noise. In the

```
AlgorithmMCL(X, vel, LB):
 X' = ∅
 for i = 0 to m do
      generate random x from X according to w_1, ..., w_m
      generate random x' ~ p(x'|vel, x)
      w' = p(LB|x')
      add⟨x', w'⟩ to X'
 endfor
 normalize the importance factors w' in X'
 return X'
```

Fig. 4. MCL

second step the samples are re-weighted with the new local ball estimates using the measurement error covariance matrix around the estimated ball position.

In practice it turned out that Monte Carlo is applicable with a set size of 1000 samples in our time constraints. With more than 1500 samples we are not able to fulfill our time constraints of 100 ms of computation time any more. In Fig. 2(c) the samples of the distribution approximation are depicted. The dark blue ball resembles the true position of the ball, the red one is the fusion result. The black dots show the samples.

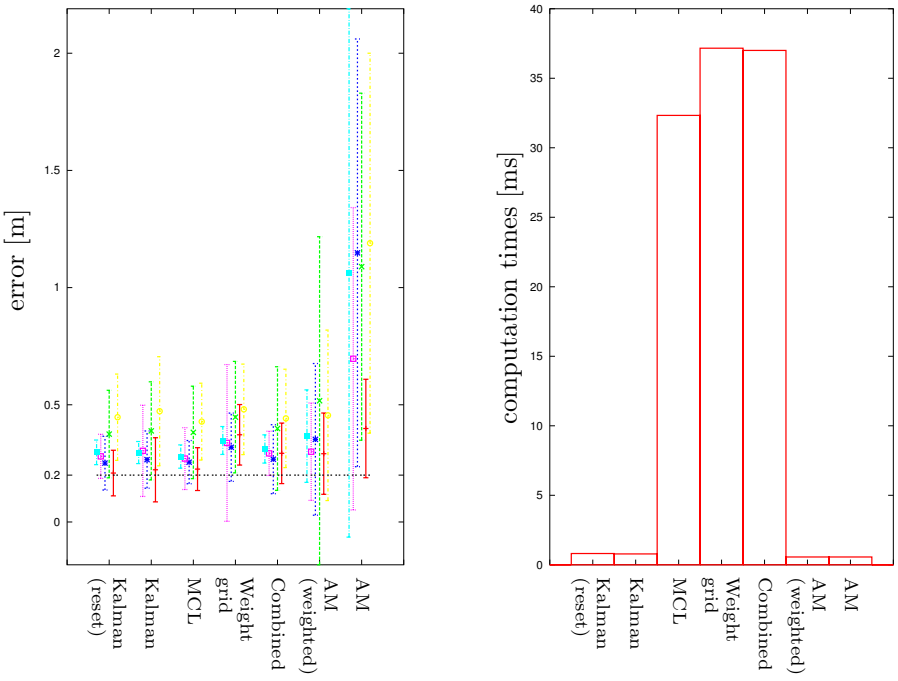
4.5 Combined Fusion

Inspired by the work of the CS Freiburg Team [3] we combined the weight grid technique (Sect. 4.2) with the Kalman filter (Sect. 4.3). All local ball estimates are used to calculate a *reference ball* by using our weight grid algorithm. Only those local balls are used as input for the Kalman filter that are in between a radius of 1 meter to the reference ball. If the resulting ball of the Kalman filter is too far away from the calculated reference ball the Kalman filter receives a reset.

5 Experiments and Results

To get significant and realistic results about the quality of the presented fusion techniques we tested them in real game situations. We made several test games at the Philips RoboCup team in Eindhoven. To get ground truth for the real ball position we used a ceiling camera. All relevant data were logged and can be replayed. The ground truth data were manually added to the log data. As this is rather elaborate, we do not have any ground truth data from the German Open or the RoboCup championships.

In the game against Philips, we picked 6 different sequences lasting between one and three minutes with significant action on the field. For example, in one



(a) The mean error for the different methods

(b) Computation times in ms

Fig. 5. Comparison of the methods

sequence the ball was blocked between two robots so that the other robots of the team did not see the ball at all. One sequence is a typical game start situation, in another one the Philips robot kicked very often (and very hard, as usual) resulting in a rapidly moving ball. During some of the sequences one or more robots were dis-localized reporting wrong ball estimates.

The evaluation results are shown in Fig. 5(a). The data represent the mean error in meters together with its deviations over each test run.[3] Not very surprisingly, the arithmetic-mean method yields the worst results, followed by the weighted arithmetic mean. The reason is that one outlier is enough to drag the merged position into the wrong direction. Even with weighting the estimates according to their distance to the ball this bias cannot be prevented. The grid-based method (combined and weight grid) are in the midfield wrt. to their accuracy.

[3] For the whole testing time we had 8.520 cycles where the fusion module calculated a global ball estimate.

The first three places are taken by the Kalman, Kalman with reset, and Monte Carlo. Depending on the game sequence one of the three scored first place. One can see that in the mean these method have an error of about 20 cm. This is better than the results reported on in [3]. They reported on a position error of about 38 cm. Moreover, they conducted their experiments in a static setting, where the robots did not move during the experiments.

With respect to the computation times one can state that the Kalman filter methods clearly outperform the other methods. With an average computation time of 0.8 ms and an error of about 20 cm the Kalman filter methods are accurate and fast. What surprised us most was that combining a Kalman filter with a weighted grid which is similar to the method proposed in [3] does not seem to pay off, as a simple Kalman performed better, both in accuracy and computation time.

In the next experiment we tested another typical RoboCup scenario: it happens very often in RoboCup that the referee picks up the ball, for instance after the ball passed the side line or was stuck between robots. Then, no robot can see the ball. The referee places the ball at some restart spot again. In this situation it is important to get stable ball estimates as soon as possible. In our second experiment (Fig 6), the robots took their kick-off positions and two helpers staying at the opposing restart points randomly placed a ball at one of these points.

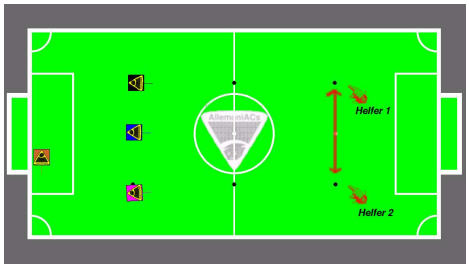


Fig. 6: "Ball stuck" experiment

In this experiment, again the Kalman reset method showed the best results. The Kalman reset had a mean error of 0.45 m, which is reasonable with respect to a minimum distance of 5 meters between the robots and the ball. Moreover, the computation time of 0.6 ms is a good result, especially compared to Monte Carlo which takes in the order of two magnitudes longer.

6 Conclusion

In the RoboCup domain it is very important to have a good estimate about the ball position. As the ball is perceived by only some robots most of the time a stable global ball estimates helps to increase the team performance. The result for RoboCup's Middle-size league is that the Kalman reset filter shows the best performance.

Fig. 7 gives an impression about the quality gain using a ball fusion technique. The lower curve shows the percentage with which all robots have seen the ball themselves, i.e. where they were able to use their local estimate. The upper one shows the availability of a global ball estimate. In over 80 % of the times there exists a ball which the robot can rely on.

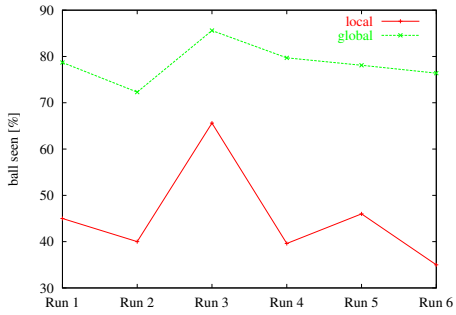


Fig. 7. Comparison between local and global ball perception.

Another nice effect of using a ball fusion technique is that one is able to detect when a robot is dis-localized. Comparing local and global estimates it is easy to notify when those values differ too much in order to detect a wrong localization position. Of course, it is crucial to find that value as false positives must be avoided. It turned out that using the Kalman reset method we detected all dis-localizations during the test runs having only one false positive. More details about that can be found in [8].

Another problem is to gather ground truth data for empirical evaluation. In our experiments we installed a ceiling camera and associated the real ball position with the logged data by hand. This is a very time consuming process. Recently, Stulp et al. [17] proposed a ceiling camera system to acquire ground truth data for the robots and the ball. With these data, we could evaluate the methods on a wider data basis. For the future we would like to conduct further experiments with real tournament data.

As the presented results are very specific to the RoboCup domain in general it turns out that one should try out several methods for the specific application domain in order to find the most appropriate method. Further, one can observe that applications of the Bayes filter (Kalman, Monte Carlo) provided the best estimates for the fusion task.

Acknowledgment

This work was supported by the German National Science Foundation (DFG) in the Priority Program 1125, *Cooperating Teams of Mobile Robots in Dynamic Environments* and a grant by the NRW Ministry of Education and Research (MSWF). We thank the anonymous reviewers for their comments.

References

1. A. Bonarini, M. Matteucci, and M. Restelli. Anchoring: do we need new solutions to an old problem or do we have old solutions for a new problem? In *Proc. AAAI*

Fall Symposium on Achoring Symbols to Sensor Data in Sungle and Multiple Robot Systems, 2001.
2. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, May 1999.
3. M. Dietl, J.-S. Gutmann, and B. Nebel. Cooperative sensing in dynamic environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS-2001)*, 2001.
4. D. Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.
5. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI/IAAI*, pages 343–349, 1999.
6. D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11, 1999.
7. C. Gönner, M. Rous, and K.-F. Kraiss. Robust color based picture segmentation for mobile robots. In *Autonome Mobile Systeme*, pages 64–74, Karlsruhe, Germany, Dec. 2003. Springer. (in German).
8. L. Hermanns. Fusing uncertain world information of cooperating robots into a global world model. Diploma thesis, Knowledge-based Systems Group, Computer Science V, RWTH Aachen, Aachen, Germany, (in German), 2004.
9. R. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, pages 35–45, Mar. 1960.
10. P. Maybeck. *Stochastic Models, Estimation and Control*, volume 1. Academic Press, 1979.
11. A. Milstein, J. N. Sànchez, and E. Williamson. Robust Global Localization Using Clustered Particle Filtering. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 581–586, 2002.
12. H. Moravec and A. Elfes. High resolution maps from wide angular sensors. In *Proc. of the IEEE International Conference On Robotics and Automation (ICRA)*, pages 116–121, 1985.
13. P. Pinheiro and P. Lima. Bayesian sensor fusion for cooperative object localization and world modeling. In *Proc. 8th Conference on Intelligent Autonomous Systems*, 2004.
14. G. Steinbauer, M. Faschinger, G. Fraser, A. Mühlenfeld, S. Richter, G. Wöber, and J. Wolf. Mostly harmless team description. In *Proc. RoboCup Symposium*, 2003.
15. A. Strack, A. Ferrein, and G. Lakemeyer. Laser-based localization with sparse landmarks. to appear in Proc. RoboCup Symposium 2005, 2005.
16. A. Stroupe, M. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proc. of 2001 IEEE Int. Conf on Robotics & Automation (ICRA-01)*, 2001.
17. F. Stulp, S. Gedikli, and M. Beetz. Evaluating multi-agent robotic systems using ground truth. In *In Proceedings of the Workshop on Methods and Technology for Empirical Evaluation of Multi-agent Systems and Multi-robot Teams (MTEE)*, 2004.
18. The RoboCup Federation. http://www.robcup.org, 2004.
19. S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2001.
20. J. Wunderlich. Technical description of the allemaniacs soccer robots. Technical report, LTI / KBSG, Aachen University, 2002.