Fast, Accurate, and Robust Self-Localization in the RoboCup Environment*

Jens-Steffen Gutmann **Thilo Weigel** **Bernhard Nebel**

Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Am Flughafen 17, D-79110 Freiburg, Germany
{gutmann,weigel,nebel}@informatik.uni-freiburg.de

Abstract

Self-localization is important in almost all robotic tasks. For playing an aesthetic and effective game of robotic soccer, self-localization is a necessary prerequisite. When we designed our robotic soccer team for RoboCup'98, it turned out that all existing approaches did not meet our requirements of being fast, accurate, and robust. For this reason, we developed a new method, which is presented and analyzed in this paper. We additionally present experimental evidence that our method outperforms other methods in the RoboCup environment.

1 Introduction

Robotic soccer is an interesting scientific challenge [10] and an ideal domain for testing new ideas and demonstrating existing techniques. One of our main intentions in participating in last year's *RoboCup'98* [1] was to demonstrate the usefulness of self-localization techniques that we have developed [8].

Solving the self-localization problem—the problem of determining the position and orientation of the robot—is necessary for almost all tasks. In robotic soccer it seems even impossible to play an effective and aesthetic game if the soccer agents do not know where they are and how they are oriented. As a matter of fact, some of the problems displayed in the games of the middle size league at *RoboCup'97* [9] seemed to have to do with the fact that the soccer robots had the wrong idea about their positions, which led to erratic movements and a number of own goals.

The self-localization problem can be addressed using a wide range of sensors (e.g. odometry, sonars, vision, compasses, laser range finders, other sensors, or combinations thereof) and a wide range of methods. In the sequel we will only consider the combination of data from the odometry and from laser range finders (LRF), since the latter provide accurate and reliable data, which can be interpreted with much less computational effort than, say, data from a vision system.

Self-localization can be based on recognizing known *landmarks* or on *dense sensor matching*. In the first approach, features are extracted from the sensor inputs and matched with the features of the landmarks in order to determine the locations of the landmarks. However, in the RoboCup environment, there are only few natural landmarks that are always visible to the sensors and for this reason we did not consider this approach. In the second approach, all sensor inputs are matched against the expected sensor inputs for a given model. Two competing methods for dense sensor matching are grid-based *Markov localization* [3; 2] and *Kalman filtering using scan matching* [5; 8]. As it has been demonstrated, Markov localization is more *robust*, because it always generates some position hypotheses and because it can recover from catastrophic failures. However, self-localization using Kalman filtering based on scan matching is more *accurate* [6], since it does not rely on grids.

For robotic soccer, we need *robustness*, *accuracy*, and *efficiency*, whereby the latter property means that we want to estimate the position and orientation in a few milliseconds. Unfortunately, none of the approaches described above satisfies all three requirements. For this reason, we designed a new scan-matching approach that extracts features from the raw sensor inputs, namely, straight lines, that are matched against an *a priori* model. Using the scan match, which can be computed efficiently, the new position estimation is then derived by combining it with the odometry reading using Kalman filtering.

2 Scan Matching

Scan matching is the process of translating and rotating a range scan (obtained from a range device such as a laser range finder) in such a way that a maximum overlap between sensor readings and an *a priori* map emerges. Most of the scan matching methods presume an initial

*This work has been partially supported by *Deutsche Forschungsgemeinschaft* (DFG) as part of the graduate school on *Human and Machine Intelligence*, by *Medien- und Filmgesellschaft Baden-Württemberg mbH* (MFG), and by *SICK AG*, who donated a set of new generation laser range finders.

pose estimation that must be close to the true pose in order to limit the search space.

The robot pose and its update from scan matching are modeled as single Gaussian distributions. This has the advantage that robot poses can be calculated with high precision, and that an efficient method for computing the update step can be used, namely, Kalman filtering.

The extended Kalman filter method has the following form. The probability of a robot pose is modelled as a Gaussian distribution $l(t) \sim N(\mu_l, \Sigma_l)$, where $\mu_l = (x, y, \alpha)^T$ is the mean value and Σ_l its 3×3 covariance matrix.

On robot motion $a \sim N((\delta, \theta)^T, \Sigma_a)$ where the robot moves forward a certain distance δ and then rotates by θ, the pose is updated according to:

$$\begin{aligned} \mu_l &:= E(F(l,a)) = \begin{pmatrix} x + \delta\cos(\alpha) \\ y + \delta\sin(\alpha) \\ \alpha + \theta \end{pmatrix} \\ \Sigma_l &:= \nabla F_l \Sigma_l \nabla F_l^T + \nabla F_a \Sigma_a \nabla F_a^T \end{aligned}$$

Here E denotes the expected value of the function F and ∇F_l and ∇F_a are its Jabobians with respect to l and a.

From scan matching a pose update $s \sim N(\mu_s, \Sigma_s)$ is obtained and the robot pose is updated using standard Kalman filter equations [13]:

$$\begin{aligned} \mu_l &:= (\Sigma_l^{-1} + \Sigma_s^{-1})^{-1} \cdot (\Sigma_l'^{-1}\mu_l + \Sigma_s^{-1}\mu_s) \\ \Sigma_l &:= (\Sigma_l^{-1} + \Sigma_s^{-1})^{-1} \end{aligned}$$

The success of the Kalman filter depends heavily on the ability of scan matching to correct the robot pose. There are a number of methods for matching scans:

Cox [5] matches sensor readings with the line segments of a hand-crafted CAD map of the environment. He assigns scan points to line segments based on closest neighborhood and then searches for a translation and rotation that minimizes the total squared distance between scan points and their target lines.

Weiss et. al. [16] use histograms for matching a pair of scans. They first compute a so-called angle histogram for determining the rotation of the two scans and then use x and y histograms for computing the translation. Although this method seems to be well suited for the RoboCup environment it is computationally expensive and the precision of the algorithm depends on the discretization size of the histograms.

Lu and Milios [12] match pairs of scans by assigning points in one scan to points in the other scan. For finding a corresponding scan point two heuristics called *closest-point-rule* and *matching-range-rule* are applied and a combination is used for computing the rotation and translation of the two scans. This IDC algorithm (*iterative dual correspondence*) is well suited for any type of environment including non-polygonal ones.

Gutmann and Schlegel [8] use a combination of the Cox matching approach and the IDC method for combining the efficiency and robustness of the line matching method with the universal capabilities of the IDC algorithm. They call their algorithm the *combined scan matcher* (CSM).

Unfortunately all those matching algorithms possess a high computational complexity, e.g. $O(n^2)$ where n are the number of scan points, and their robustness is limited due to the small search space.

Therefore we developed a new algorithm LineMatch that makes use of the simple polygonal structure of the RoboCup environment and trades off generality for speed and the ability to globally localize the robot on the soccer field.

3 The LineMatch Algorithm

The LineMatch algorithm extracts line segments from a scan and matches them with an *a priori* map of line segments similar to the methods of [15; 4]. We expected that this algorithm has better run-time performance and is more robust than the other scan matchers while retaining the same accuracy as the other matchers. In how far these expectations are realistic will be shown in Section 4.

In order to guarantee that extracted lines really correspond to field-border lines, only scan lines significantly longer than the extent of soccer robots are considered. The following algorithm shows how a matching between model lines and scan lines is computed by recursively trying all pairings between scan lines and model lines:

Algorithm 1 LineMatch*(M, S, P)*
Input: model lines M, scan lines S, pairs P
Output: set of positions hypotheses H

```
if |P| = |S| then
  H := P
else
  H := ∅
  s := SelectScanline(S, P)
  for all m ∈ M do
    if VerifyMatch(M, S, P ∪ {(m, s)}) then
      H := H ∪ {LineMatch(M, S, P ∪ {(m, s)})}
return H
```

SelectScanline selects the next scan line that should be matched and *VerifyMatch* verifies that the new (m, s) pairing is compatible with the set of pairings P already accepted by computing a common rotation and translation. The algorithm returns position hypotheses in the form of sets of pairs which can be easily transformed into possible locations where the scan could have been taken. For the RoboCup field the algorithm is capable of determining the global position of the robot modulo the symmetry of the field. This means that we get two position hypotheses if three field borders are visible (see Figure 1) and four hypotheses if two borders are visible.

This scan matching method is similar to the methods described by Castellanos *et al.* [4] and Shaffer *et al.* [15]. In contrast to these approaches, however, we only verify that the *global constraints* concerning translation and rotation as well as the length restrictions of scan lines are satisfied. This is sufficient for determining the position hypothesis and more efficient. Further, we do not need any initial estimation of the pose, which means

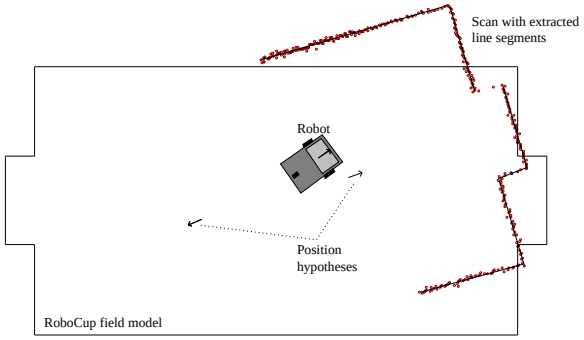

Fig. 1. The LINEMATCH algorithm returns two hypotheses for the robot position.

that even if the robot has an extreme error in its position estimation, it may still be able to recover from that.

After matching a range scan, the most plausible position is used in the Kalman filter step for updating the robot position. We use the position information from odometry to determine the most plausible position based on a combination of closest neighborhood and similarity in heading.

For initializing the self-localization system the robot is placed at any position in the RoboCup field but roughly oriented towards the opponent goal and the mean and error covariance of the robot position are set to:

$$\mu_l := (0,0,0)^T$$
$$\Sigma_l := \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix}$$

This ensures global self-localization on the first scan match.

While it turns out that the implemented algorithm is extremely fast in the RoboCup environment (see Section 4.2), one may wonder how well it scales with the size of the set M. A first rough analysis suggests that the worst-case runtime of the algorithm is $O(|M|^{|S|})$, because the depth of the recursion is $|S|$ and in each recursive call of LINEMATCH $|M|$ different pairings are tried.

As it turns out, however, it is possible to come up with a much better run-time estimation. After the second level of recursion, when two pairings have been made, all degrees of freedom for rotation and translation have been removed (*SelectScanline* is implemented in such a way that it chooses non-parallel lines in the first two levels of recursion). This means that on deeper levels of the recursion only one pairing can be consistent, which leads to invoking another recursive call of LINEMATCH. This means that we may get $|M|^2$ possible pairings on the first two levels of recursion which are verified by further recursive calls trying $|M||S|$ different pairings. Finally, since *VerifyMatch* needs $O(|S|)$ time, we get an overall bound of $O(|M|^3|S|^2)$. In the general case, one has to live with the cubic upper bound. Nevertheless, for realistic environments where not all walls are simultaneously visible—such as is the case in office environments—preprocessing can be used to guarantee runtime almost linear in $|M|$. Such a preprocessing phase would store for each line all other lines that are simultaneously visible. Using such a data structure, the amount of lines that must be tested can be dramatically reduced and assuming a constant upper bound of simultaneously visible walls, we would get a linear complexity of the algorithm.

4 Comparison with other Scan Matchers

In order to show the advantages of the LINEMATCH algorithm we compared the Cox, CSM and LINEMATCH techniques with each other. We did not include the IDC and histogram matching methods as the properties of these algorithms are covered by the CSM algorithm [8].

Since the CSM algorithm needs a set of reference scans as its *a priori* map, we collected a small set of scans, corrected the accumulated odometry error by applying the registration method from [11], and used them as reference scans. This approach has proven to be a successful and easy way for enabling mobile robot navigation in an indoor environment without modifying the environment or creating hand-crafted maps [7].

For comparing the different methods we recorded real data with one of our mobile robotic soccer players. Each of our soccer robots is a *Pioneer I* mobile robot equipped with a *SICK* laser range finder, a *Cognachrome* vision system for ball tracking, a *Libretto 70CT* laptop with wireless ethernet connection and a custom kicking device. The laser range finder covers a 180° field of view with an angular resolution of 1° and a range resolution of $5cm$.

In order to record data of a realistic game scenario we ran the soccer robot in our RoboCup environment with several stationary and moving obstacles. From these data we computed the average run-time of the different algorithms and added different kinds of noise to the data for determining the accuracy and robustness of the methods.

Similar work has been reported by Shaffer *et al.* [14], who compared two scan matching methods that are similar to the Cox and LINEMATCH algorithm in this paper. However, they used only single scan matches for their experiments whereas in our experiments all data recorded during a whole robot run is taken into account. Also they only ran their algorithms in an almost static environment whereas we recorded our data in a realistic dynamic scenario with many stationary and moving obstacles that can block the robot's sensors. Therefore the results presented in this paper should give a better picture of how good the methods actually are in a dynamic environment like RoboCup.

4.1 Noise Models

There are several kinds of noise typically observed when robots operate in real-world environments. On one hand

there is a typical Gaussian noise in the odometry and proximity sensors coming from the inherent inaccuracy of the sensors. On the other hand there are non-Gaussian errors arising from robot colliding with obstacles, e.g. other robot players, or from interference with the sensors.

In this paper, odometry errors coming from wheel-slippage, uneven floors, or different payloads are characterized according to the following three parameters (see left part of Figure 2).

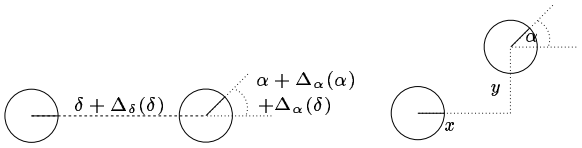

Fig. 2. Effect of adding noise $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta) \rangle$ (left) and bump noise $\langle x, y, \alpha \rangle$ (right) to the odometry.

Range noise: the error $\Delta_\delta(\delta)$ in range when the robot moves a certain distance δ.

Rotation noise: the error $\Delta_\alpha(\alpha) + \Delta_\alpha(\delta)$ in rotation when the robot turns a certain angle α or moves a certain distance δ.

There is another source of less frequent but much larger odometry error coming from situations in which the robot collides with obstacles. These abrupt errors can be characterized by the following parameters (see right part of Figure 2).

Error of the odometry: The error x, y, and α is added to the odometry information.

Frequency: Probability that a bump occurs if the robot travels one meter. Throughout the experiments described below, this probability was set to 0.2 per meter travelled.

4.2 Run-Time Performance

For computing the run-time performance of the scan matching techniques we measured the average time a method needed for computing the pose update before it is fused with the odometry estimate. In order to receive measurements that show the performance under real game conditions we setup a realistic game scenario in our RoboCup environment with stationary and moving objects (see Figure 3) and used our soccer robot as a right defender where it moved over the entire field a couple of times. In this run the robot moved a total distance of approximately 41 meters, turned about a total of 11000 degrees (about 30 revolutions) and collected over 3200 scans.

Figure 4 shows run-time results performed on the robots on-board computer, a Pentium 120 MHz laptop running the Linux operating system. As expected the LineMatch algorithm outperforms the other competing techniques. It is 8 times faster than the Cox algorithm and about 20 times faster than the CSM method. The

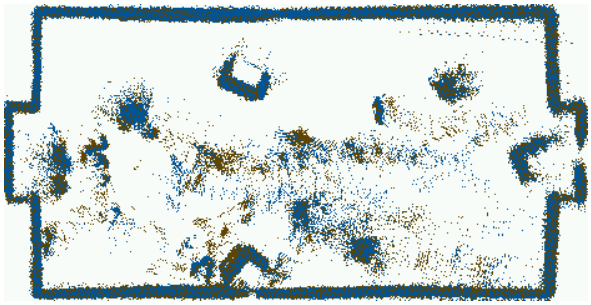

Fig. 3. Experimental setup: several boxes were placed in the RoboCup field to give a realistic game scenario. Noisy sensor readings are caused by moving obstacles.

very low average run-time of only $2ms$ per scan match allows the processing of all incoming range finder data in real time.

Cox	CSM	LineMatch
$16ms$	$39ms$	$2ms$

Fig. 4. Run-time results on a Pentium 120MHz laptop.

4.3 Performance in a Game Scenario

For showing the accuracy and robustness of the LineMatch algorithm we used the data collected in the above run and added different kinds of noise to the odometry information. In order to measure the accuracy of the position estimates generated by the different matching methods, a set of reference positions are needed. To ease the determination of the reference positions we ran the Cox method with the recorded data and used this output as the set of reference positions.

For each set of noise values, 26 runs with different seed values for initializing a random noise generator were performed. Figure 5 shows the trajectory measured by the robots wheel encoders and a typical

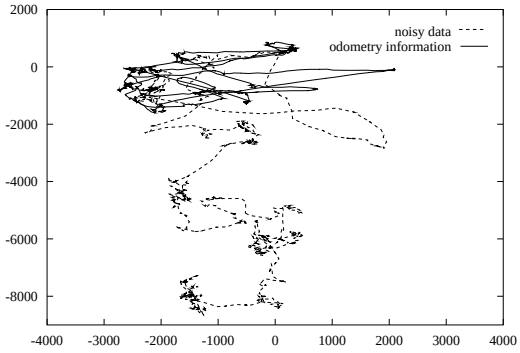

Fig. 5. Trajectory measured by the robot and typical trajectory obtained by adding large Gaussian noise with standard deviations $\langle 400, 100, 40 \rangle$ to these data.

trajectory when adding the maximum Gaussian noise

$\langle 400, 100, 40\rangle$. The values correspond to the standard deviation of the Gaussian noise $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta)\rangle$ with the units $\sqrt{mm^2/m}$, $\sqrt{deg^2/360°}$, and $\sqrt{deg^2/m}$.

For each scan matching method we computed the number of times the robot position was lost and the distance and heading error to the reference pose in case the position was not lost. We used a threshold of $0.5m$ for the distance and 30° for the heading error for determining whether or not the position of the robot was lost.

Figure 6 shows the average distance and Figure 7 the average heading error to the reference positions for five different levels of Gaussian noise. The value triples on the x-axis correspond to the standard deviation of the Gaussian noise $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta)\rangle$. In these and all following figures the error bars indicate the 95% confidence interval of the average mean.

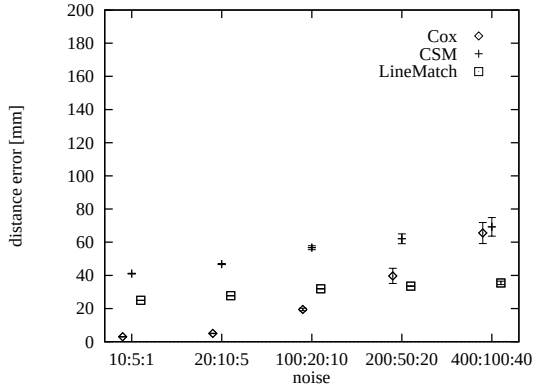

Fig. 6. Distance error to reference positions in typical game scenario for different levels of Gaussian noise.

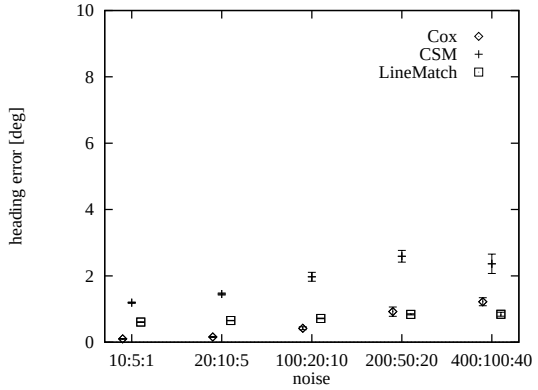

Fig. 7. Heading error to reference positions in typical game scenario for different levels of Gaussian noise.

From both figures it can be seen that all three methods have a similar accuracy usually better than $5cm$ and 2°. Only the Cox method has a significant higher accuracy than the others when only little Gaussian noise is present but this is due to the fact that the reference positions also have been generated by the Cox method.

However, the LineMatch method is much more robust than the other matching algorithms. Figure 8 shows the number of times where the robot position was lost for the same levels of Gaussian noise as in the previous figures. Here the LineMatch algorithm shows a very

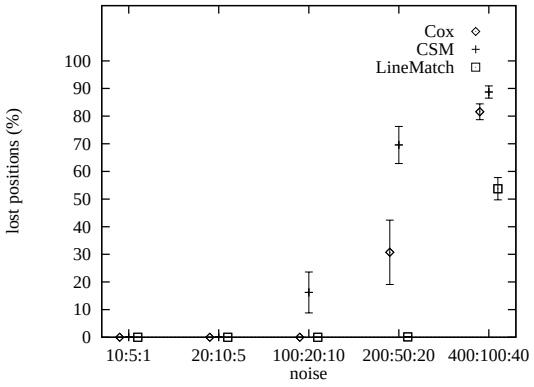

Fig. 8. Number of times where position error was above $0.5m$ or above 30° in typical game scenario for different levels of Gaussian noise.

good performance and keeps the robot localized even under high odometry noise. Only for the maximum level of noise, LineMatch also starts losing the position. We believe that the higher robustness of LineMatch is due to the larger search space it uses for finding matches.

In the same manner, we investigated how the methods compare given simulated bump noise. For accuracy the results were similar to the case of Gaussian noise. All three methods had a similar accuracy for the distance and heading error than in the Gaussian case. Figure 9 shows the average number of positions where the robot was lost when bump noise was added to the odometry information. The triples at the x-axis correspond to the

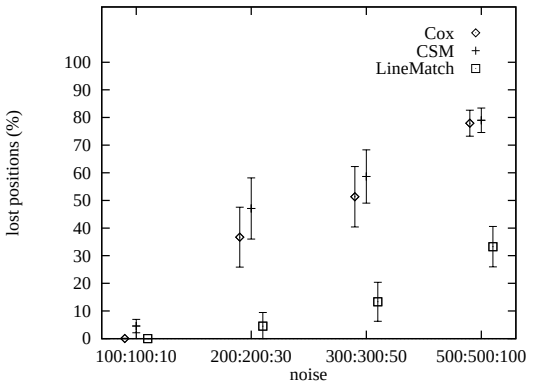

Fig. 9. Number of times where position error was above $0.5m$ or above 30° in typical game scenario for different levels of bump noise.

bump noise values $\langle x, y, \alpha\rangle$ used in this experiment. The scale of these values is mm for x and y, and degrees for

α. In addition to these bumps occurring with probability 0.2 per meter, we applied a small Gaussian odometry error using the parameters $\langle 100, 5, 2 \rangle$. As can be seen in Figure 9 all scan matching approaches have problems when bump noise is present. This is due to the fact that the Gaussian distribution assumption when fusing the observations with odometry in the Kalman filter does not model bump noise well. However the LINEMATCH method shows less failures than the other methods and is thus again more robust than the other ones.

In a final set of experiments, which can not be covered in this paper due to lack of space, we compared the scan matching methods in "confusing game scenarios" where a long wall was placed inside the RoboCup field. We expected that under these conditions the LINEMATCH algorithm gets irritated since the long wall is not filtered out in its preprocessing step and thus LINEMATCH produces wrong matches or relies on dead-reckoning only for the position estimation. Luckily the LINEMATCH algorithm did not suffer too much from these conditions. We suspect that this is due to the fact that there are a lot of situations where the irritating wall is not present in the range scans.

5 Conclusion and Future Work

In this paper we presented a new method for matching range scans to an *a priori* model of line segments which is well suited for localizing a mobile robot in a polygonal-shaped, dynamic environment like RoboCup. Experimental results confirm that the new method is much faster and much more robust than other existing scan matchers while retaining the accuracy of the competing methods.

The proposed method has been developed as one of the key components of the *CS Freiburg* robotic soccer team and has been proven to be fast, reliable, precise and robust. It never failed in any official or in-official game and led the team to its success at RoboCup'98 where *CS Freiburg* won the competition in the middle size league [1].

Although the method has been utilized for RoboCup so far only, it is an obvious step to use it in other polygonal-shaped environments, e.g. as a localization method in our navigation system for office environments [7]. Therefore we will extend the algorithm in various ways, e.g. to allow for partial matches where not all lines of a range scan are matched to model lines and to explore several ways to optimize the algorithm in order to deal with larger environments.

Finally we are going to explore the problem of cooperative self-localization in the RoboCup environment for allowing the reorientation of disoriented group members.

References

[1] M. Asada and H. Kitano, editors. *RoboCup-98: Robot Soccer World Cup II*. LNAI. Springer-Verlag, Berlin, 1999. To appear.

[2] W. Burgard, A. Derr, D. Fox, and A. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *Proc. IROS'98*, 1998.

[3] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. AAAI'96*, 896–901, 1996.

[4] J. A. Castellanos, J. D. Tardós, and J. Neira. Constraint-based mobile robot localization. In *International Workshop on Advanced Robotics and Intelligent Machines*. University of Salford, Apr. 1996.

[5] I. J. Cox. Blanche—an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. Robotics and Automation*, 7(2):193–204, 1991.

[6] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An experimental comparison of localization methods. In *Proc. IROS'98*. IEEE/RSJ, 1998.

[7] J.-S. Gutmann and B. Nebel. Navigation mobiler Roboter mit Laserscans. In *Autonome Mobile Systeme (AMS'97)*, 36–47. Springer-Verlag, 1997.

[8] J.-S. Gutmann and C. Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proc. 1st Euromicro Workshop on Advanced Mobile Robots*, 61–67. IEEE, 1996.

[9] H. Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1998.

[10] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. RoboCup: A challenge problem for AI. *The AI Magazine*, 18(1):73–85, 1997.

[11] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[12] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *J. Intelligent and Robotic Systems*, 18:249–275, 1997.

[13] P. S. Maybeck. The Kalman filter: An introduction to concepts. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*. Springer-Verlag, 1990.

[14] G. Shaffer, J. Gonzalez, and A. Stentz. Comparison of two range-based estimators for a mobile robot. In *SPIE Conf. on Mobile Robots VII*, 661–667, 1992.

[15] G. Shaffer et al. Position estimator for underground mine equipment. In *IEEE Trans. Industry Applications*, volume 28, September 1992.

[16] G. Weiss and E. Puttkamer. A map based on laserscans without geometric interpretation. In U. Rembold et al., editor, *Intelligent Autonomous Systems*, 403–407. IOS Press, 1995.