Fast and Robust Edge-Based Localization in the Sony Four-Legged Robot League *

Thomas Röfer[1] and Matthias Jüngel[2]

[1] Bremer Institut für Sichere Systeme, Technologie-Zentrum Informatik, FB 3, Universität Bremen. E-Mail: `roefer@tzi.de`

[2] Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin. E-Mail: `juengel@informatik.hu-berlin.de`

Abstract. This paper presents a fast approach for edge-based self-localization in RoboCup. The vision system extracts edges between the field and field lines, borders, and goals following a grid-based approach without processing whole images. These edges are employed for the self-localization of the robot. Both image processing and self-localization work in real-time on a Sony Aibo, i. e. at the frame rate of the camera. The localization method was evaluated using a laser range sensor at the field border as a reference system.

1 Introduction

The Sony Four-Legged Robot League (SFRL) is one of the official leagues in RoboCup. Besides the use of four-legged robots, there are some other specialties in that league. The first one is that the robot platform is standardized, i. e. the Sony Aibo ERS-210 and ERS-210A (cf. Fig. 1a) are the only permitted systems, and they can only be used without any modification. Therefore, in some sense the SFRL can be seen as a *software league*, because it is neither possible nor required to construct robots. Another characteristic is that the robots are completely autonomous, i. e. there is no external computer beside the field (except from one running the so-called *game manager* for the referee) that can help the players in their calculations. The main sensor of the Sony Aibo is the camera located in its head. The head can be turned around three axes (tilt, pan, and roll), and the camera has a field of view of 58° by 48°. Thus, all teams in the league have to tackle the problem of *directed vision* (in contrast to *omni-vision* as often used in the middle-sized league or in the first approaches of the small-sized league to local vision systems). With 20 degrees of freedom, the color camera, and more than 30 further sensors, the movements and the sensor equipment of the robots are the most complex in RoboCup so far, and they have to be controlled by a single 200 MHz MIPS processor (400 MHz in the ERS-210A), i. e. all algorithms used, e. g. for image-processing or self-localization, have to be highly efficient to run in real-time.

* The Deutsche Forschungsgemeinschaft supports this work through the priority program "Cooperating teams of mobile robots in dynamic environments".

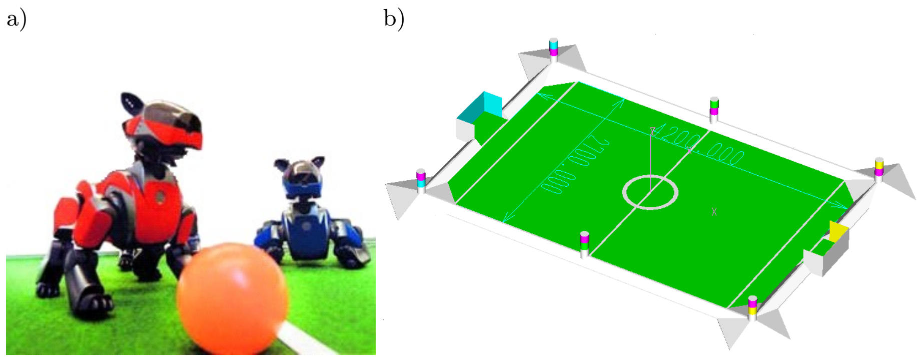

Fig. 1. a) Two Sony Aibo robots and a ball. b) The field used in the SFRL.

The soccer field in the SFRL has a size of approximately 5m×3m (cf. Fig. 1b). As the main sensor of the robot is a camera, all objects on the RoboCup field are color coded. There are two-colored flags for localization (pink and either yellow, green, or skyblue), the two goals are of different color (yellow and skyblue), the ball is orange (as in all RoboCup leagues), and the robots of the two teams wear tricots in different colors (red and blue). However, there are no flags on a real soccer field, and as it is the goal of the RoboCup initiative to compete with the human world champion in 2050, it seems to be a natural thing to develop techniques for self-localization that do not depend on artificial clues. In the SFRL, all teams have to participate in three technical challenges as part of the RoboCup championship. In 2003, self-localization without the six two-colored flags around the field is one of these challenges. This challenge can be seen as a preparation to remove the flags in the soccer games in 2004.

2 Grid-Based Line Detection

The localization method presented in this paper relies on the detection of edges between differently colored objects on the field: the edges between the skyblue goal and the field, the edges between the yellow goal and the field, the edges between the border and the field, and the edges between the field lines and the field (cf. Fig. 2a). The key idea of the method presented here is not to actually extract *lines* from the image, but *pixels on lines* instead. This approach is faster and more robust against misinterpretations, because lines are often partially hidden either by other robots or due to the limited opening angle of the camera.

A very common preprocessing step for vision-based object recognition is color segmentation using color tables, e. g. [1, 9]. Such methods directly map colors to color classes on a pixel by pixel basis, which has some crucial drawbacks. On the one hand, the color mapping has to be adapted when the lighting conditions change, on the other hand, the mapping results in a loss of information, because the membership of a pixel in a certain class is a yes/no decision, ignoring the

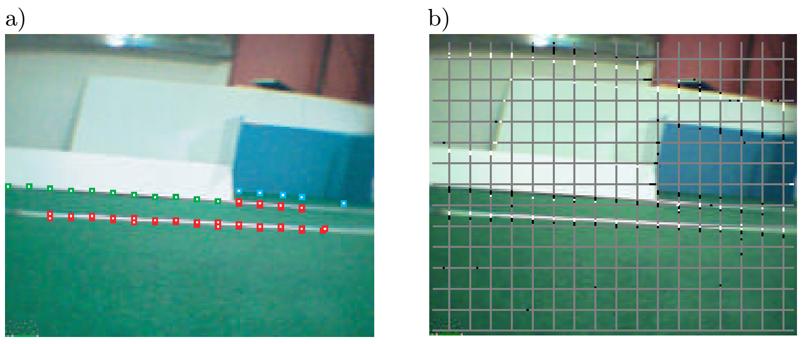

Fig. 2. Detection of lines. a) Three types of lines: field/goal, field/border, and field/line. b) The scan lines are scanned from top to bottom and from left to right. White pixels: increase in Y-channel, black pixels: decrease in Y-channel.

influences of the surrounding pixels. Some researchers try to overcome these limitations [5], but the solutions are too slow to work under real-time conditions on a robot such as the Aibo.

The key ideas of the image-processing method used in this paper are that speed can be achieved by avoiding processing all pixels of an image, and a certain independence of the lighting conditions can be reached by focusing on contrast patterns in the three different color channels. In case of the Aibo, these channels are Y, U, and V.

To find pixels on edges, in the image horizontal and vertical lines having a distance of ten pixels to each other are scanned from left to right and from top to bottom following the method described in [6] (cf. Fig. 2b). In contrast to this method color classification is only applied when a significant decrease in the Y-channel is recognized, because the field is darker then the adjacent surfaces of the field lines, the border, and the goals. If such a decrease in brightness has been detected, the colors above and below are this edge are checked for being green, white, skyblue, or yellow using a color table (cf. [7] for a solution to this problem without using color tables).

If the color above the decrease in the Y-channel is skyblue or yellow, the pixel lies on an edge between a goal and the field. The differentiation between a field line and the border is a bit more complicated. In most of the cases the border has a bigger size in the image than a field line. But a far distant border might be smaller than a very close field line. For that reason the pixel where the decrease in the Y-channel was found is assumed to lie on the ground. With the known height and rotation of the camera the distance to that point is calculated by projecting it to the ground plane. The distance leads to expected sizes of the border and the field line in the image. For the classification these sizes are compared to the distance between the increase and the decrease of the Y-channel in the image. The projection of the pixels on the field plane is also used to determine their relative position to the robot (cf. Fig. 3).

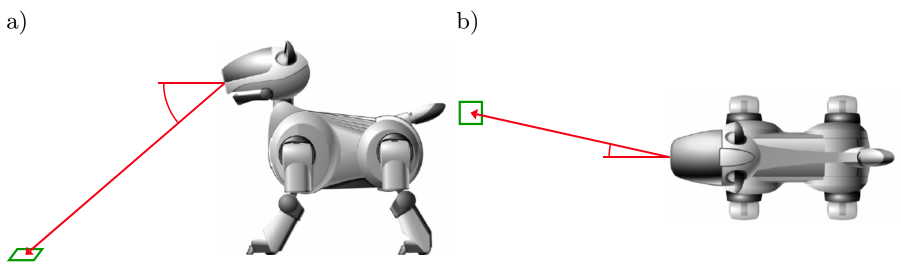

Fig. 3. The projection of edge points to the field plane. a) Vertically. b) Horizontally.

3 Self-Localization Based on Edge Points

An approach to self-localization is the so-called Monte-Carlo localization (MCL) by Fox *et al.* [3]. It is a probabilistic method, in which the current location of the robot is modeled as the density of a set of particles. Each particle can be seen as the hypothesis of the robot being located at that position. Therefore, such particles mainly consist of a robot pose (x, y, θ), i. e. a vector representing the robot's x/y-coordinates and its rotation θ.

In many implementations, MCL was used on robots equipped with distance sensors such as laser scanners or sonar sensors, e. g. in the original one [3]. Only in a few approaches, vision is used for self-localization [2, 11]. Self-localization in RoboCup is different, because the area the robots can be located at is relatively small, i. e. the field, but in that area the position of the robots has to be determined quite precisely to allow different robots of the same team to communicate about objects on the field, and to follow some location-based rules of the game. Odometry is very unreliable, because the robots walk, and they tend to push each other around. As the Aibo is equipped with a sensor with a narrow opening angle of 58°, only a few objects usable for self-localization can be seen at once, and sometimes misreadings are in the majority. The method presented here takes these circumstances into account.

3.1 Monte-Carlo Localization

A Markov-localization method requires both a *motion model* and an *observation model*. The motion model expresses the probability for certain actions to move the robot to certain relative positions. The observation model describes the probability for taking certain measurements at certain locations.

The localization approach works as follows: first, all particles are moved according to the motion model of the previous action of the robot. Then, the probabilities q_i are determined for all particles on the basis of the observation model for the current sensor readings. Based on these probabilities, the so-called *resampling* is performed, i. e. moving more particles to the locations of samples with a high probability. Afterwards, the average of the probability distribution is determined, representing the best estimation of the current robot pose. Finally, the process repeats from the beginning.

3.2 Motion Model

The motion model represents the effects of actions on the robot's pose. First of all, an odometry position is maintained that is derived from the motions performed (gaits, kicks, etc.). As this value is only a rough estimate, in addition a random error Δ_{error} is assumed that depends on the distance traveled and the rotation performed since the last self-localization. For each sample, the new pose is determined as $pose_{new} = pose_{old} + \Delta_{odometry} + \Delta_{error}$. Note that the operation $+$ involves coordinate transformations based on the rotational components of the poses.

3.3 Observation Model

The localization is based on the points on edges determined by the image-processing system (cf. Sect. 2). Each pixel has an edge type (field, border, yellow goal, or blue goal), and by projecting it on the field, a relative offset from the body center of the robot is determined. Note that the calculation of the offsets is prone to errors because the pose of the camera cannot be determined precisely. In fact, the farther away a point is, the less precise the distance can be determined. However, the precision of the direction to a certain point is not dependent on the distance of that point.

Information Provided by Edge Points. The four edge types provide very different information: *The field lines* are mostly oriented across the field. As lines only provide localization information perpendicular to their orientation, the field lines can only help the robot to find its position along the field. The field lines are seen less often than the border. *The border* is surrounding the field. Therefore it provides information in both Cartesian directions, but it is often quite far away from the robot. Therefore, the distance information is less precise than the one provided by the field lines. The border is seen from nearly any location on the field. *Goals* are the only means to determine the orientation on the field, because the field lines and the border are mirror symmetric. The goals are seen only rarely.

If the probability distribution for the pose of the robot had been modeled by a large set of particles, the fact that different edges provide different information and that they are seen in different frequency would not be a problem. However, to reach real-time performance on an Aibo robot, only a small set of samples can be employed to approximate the probability distribution. In such a small set, the samples sometimes behave more like individuals than as a part of joint distribution. To clarify this issue, let us assume the following situation: as the field is mirror symmetric, only the recognition of the goals can determine the correct orientation on the field. Many samples will be located at the actual

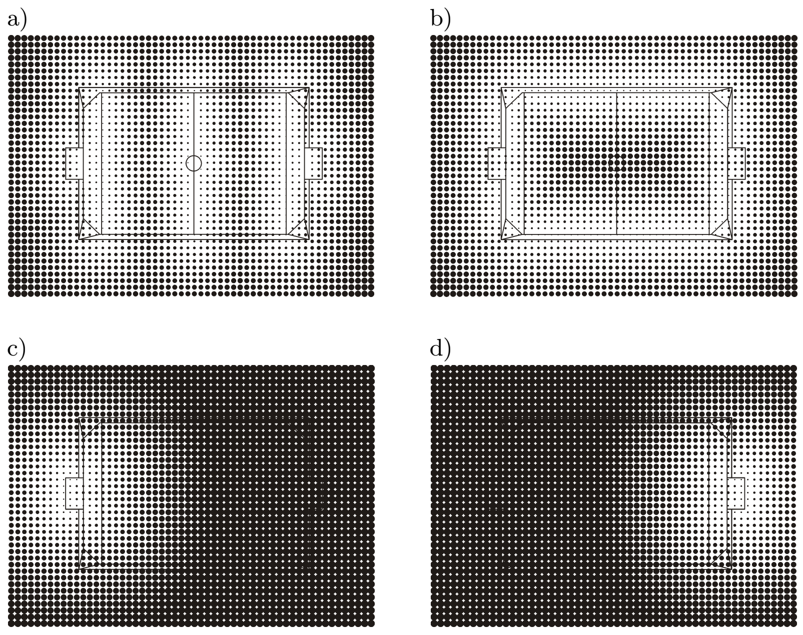

Fig. 4. Distances from edges. Distance is visualized as thickness of dots. a) Field lines. b) Border. c) One goal. d) The other goal.

location of the robot, but several others are placed at the mirror symmetric variant, because only the recognition of the goals can discriminate between the two possibilities. For a longer period of time, no goal is detected, but the border and the field lines are seen. Under these conditions, it is possible that the samples on the wrong side of the field better match the measurements of the border and the field lines than the correctly located ones, resulting in a higher probability for the wrong position. So the estimated pose of the robot will flip from one orientation alternative to the other without ever seeing a goal. This is not the desired behavior, and it would be quite risky in actual soccer games.

To avoid this problem, separate probabilities for field lines, borders, and goals are maintained for each particle.

Closest Model Points. The projections of the pixels are used to determine the three probabilities of each sample in the Monte-Carlo distribution. As the positions of the samples on the field are known, it can be determined for each measurement and each sample, where the measured points would be located on the field if the position of the sample was correct. For each of these measured points in field coordinates, it can be calculated, where the closest point on a real field line of the corresponding type is located. Then, the horizontal and vertical angles from the camera to this model point are determined. These two angles of the model point are compared to the two angles of the measured point. The smaller the deviations between the model point and the measured point from a hypothetic position are, the more probable the robot is really located at that position. Deviations in the vertical angle (i. e. distance) are judged less rigidly than deviations in the horizontal angle (i. e. direction).

Calculating the closest point on an edge in the field model for a small number of measured points is still an expensive operation if it has to be performed for, e. g., 100 samples. Therefore, the model points are pre-calculated for each edge type and stored in two-dimensional lookup tables with a resolution of 2.5 cm. That way, the closest point on an edge of the corresponding type can be determined by a simple table lookup. Figure 4 visualizes the distances of measured points to the closest model point for the four different edge types.

Probabilities. The observation model only takes the bearings on the edges into account that are actually seen, i. e., it is ignored whether the robot has *not* seen a certain edge that it should have seen according to its hypothetical position and the camera pose. Therefore, the probabilities of the particles are only calculated from the similarities of the measured angles to the expected angles. Each similarity s is determined from the measured angle ω_{seen} and the expected angle ω_{exp} for a certain pose by applying a sigmoid function to the difference of both angles weighted by a constant σ:

$$s(\omega_{seen}, \omega_{exp}, \sigma) = \mathrm{e}^{-\sigma(\omega_{seen} - \omega_{exp})^2} \tag{1}$$

If α_{seen} and α_{exp} are vertical angles and β_{seen} and β_{exp} are horizontal angles, the overall similarity of a sample for a certain edge type is calculated as:

$$p = s(\alpha_{seen}, \beta_{seen}, \alpha_{exp}, \beta_{exp}) = s(\alpha_{seen}, \alpha_{exp}, 4) \cdot s(\beta_{seen}, \beta_{exp}, 100) \tag{2}$$

Calculating the probability for all points on edges found and for all samples in the Monte-Carlo distribution would be a costly operation. Therefore, only a single point of each edge type (if detected) is selected per image by random. To achieve stability against misreadings, resulting either from image processing problems or from the bad synchronization between receiving an image and the corresponding joint angles of the head, the change of the probability of each sample for each edge type is limited to a certain maximum. Thus misreadings will not immediately affect the probability distribution. Instead, several readings are required to lower the probability, resulting in a higher stability of the distribution. However, if the position of the robot was changed externally, the measurements will constantly be inconsistent with the current distribution of the samples, and therefore the probabilities will fall rapidly, and resampling (cf. Sect. 3.4) will take place.

The filtered probability p' for a certain edge type is updated ($p'_{old} \to p'_{new}$) for each point of that type:

$$p'_{new} = \begin{cases} p'_{old} + 0.01 & \text{if } p > p'_{old} + 0.01 \\ p'_{old} - 0.005 & \text{if } p < p'_{old} - 0.005 \\ p & \text{otherwise.} \end{cases} \tag{3}$$

The probability q of a certain particle is the product of the three separate probabilities for edges of field lines, the border, and goals:

$$q = p'_{field\ lines} \cdot p'_{border} \cdot p'_{goals} \tag{4}$$

3.4 Resampling

In the resampling step, the samples are moved according to their probabilities. This is done in two steps: First, the samples are copied from the old distribution to a new distribution. Their frequency in the new distribution depends on the probability q of each sample, so more probable samples are copied more often than less probable ones, and improbable samples are removed. In a second step that is in fact part of the next motion update, the particles are moved locally according to their probability. The more probable a sample is, the less it is moved. This can be seen as a probabilistic random search for the best position, because the samples that are randomly moved closer to the real position of the robot will be rewarded by better probabilities during the next observation update steps, and they will therefore be more frequent in future distributions. The samples are moved according to the following equation:

$$pose_{new} = pose_{old} + \begin{pmatrix} \Delta_{trans}(1-q) \times rnd \\ \Delta_{trans}(1-q) \times rnd \\ \Delta_{rot}(1-q) \times rnd \end{pmatrix} \tag{5}$$

rnd returns random numbers in the range $[-1 \ldots 1]$. Typical values used for Δ_{trans} and Δ_{rot} are 20 cm and 30°.

3.5 Drawing from Observations

So far, the observation of edge points has only been used to determine the probability of the robot for being at a certain location. However, observations can also be used to generate candidate positions for the localization, i. e. to place samples at certain positions on the field. This approach follows the *sensor resetting* idea of Lenser and Veloso [8], and it can be seen as the small-scale version of the Mixture MCL by Thrun *et al.* [10]. As a single observation cannot uniquely determine the location of the robot, candidate positions are drawn from all locations from which a certain measurement could have been made. To realize this, the robot is equipped with a table for each edge type that contains a large number of poses on the field indexed by the distance to the edge of the corresponding type that would be measured from that location in forward direction. Thus for each measurement, a candidate position can be drawn in constant time from a set of locations that would all provide similar measurements. As all entries in the table only assume measurements in forward direction, the resulting poses have to be rotated to compensate for the direction of the actual measurement.

Such candidate positions are used to replace samples with a low probability. Whether a sample j is replaced or not is also drawn, based on the probability of that sample in relation to the average probability of all samples, i. e. if the following condition is satisfied:

$$\frac{rnd}{n}\sum_{i}^{n} q_i > q_j \tag{6}$$

In this case, rnd provides a random number between 0 and 1. If a sample is replaced, the new sample has probabilities p' that are a little bit below the average. Therefore, they have to be acknowledged by further measurements before they are seen as real candidates for the position of the robot.

3.6 Estimating the Pose of the Robot

The pose of the robot is calculated from the sample distribution in two steps: first, the largest cluster is determined, and then the current pose is calculated as the average of all samples belonging to that cluster. To calculate the largest cluster, all samples are assigned to a grid that discretizes the x-, y-, and θ-space into $10 \times 10 \times 10$ cells. Then, it is searched for the $2 \times 2 \times 2$ sub-cube that contains the maximum number of samples. All samples belonging to that sub-cube are used to estimate the current pose of the robot. Whereas the mean x- and y-components can directly be determined, averaging the angles is not straightforward, because of their circularity. Instead, the mean angle θ_{robot} is calculated as the orientation of the sum of all direction vectors:

$$\theta_{robot} = \operatorname{atan2}\left(\sum_i \sin\theta_i, \sum_i \cos\theta_i\right) \tag{7}$$

4 Experiments

To judge the performance of the localization approach, two different experiments were conducted. The first one measures the localization error when the robot is continuously moving. The second one evaluates the precision in reaching certain goal points.

4.1 Experimental Setup

To be able to evaluate the precision of an approach for self-localization, a reference method for localization is required. Gutmann and Fox [4] have analyzed different localization approaches using the Aibo by manually controlling the robot around using a joystick, and whenever it reached a position that was previously marked, they stored the position of that marker and the position as calculated by the robot in a log file. They also stored all perceptions of the robot, allowing them to test different localization approaches based on the same data.

The setup used for the experiments presented in this paper is a little bit different. To be able to continuously track the position of the robot, a laser

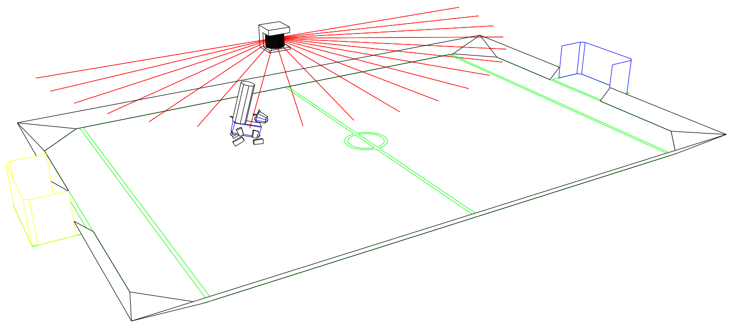

Fig. 5. The experimental setup. The laser scanner is fixed to the border of the field. The robot carries a vertical paper tube on its back that is measured by the laser sensor.

range finder was placed at the border of the field. Within its opening angle of 180°, it measured distances in a height of 35 cm, i. e. above the goals. The robot used for the experiment was equipped with a paper tube on its back that was high enough to be detected by the laser range finder (cf. Fig. 5). This way, the position of the robot could easily be determined by searching for an area that was significantly closer to the laser scanner than the neighboring areas. The shortest distance within that area plus the radius of the tube was used as distance to the robot. Together with the angle under which the robot was measured, the exact location of the robot was determined.

In both experiments, the robot was continuously turning its head from left to right and vice versa. The Monte-Carlo localization method used 100 samples.

4.2 Experiment 1

The goal of the first experiment was to judge the precision of the localization approach when the robot is continuously moving. To accomplish this, the robot was randomly moved around on the field with a maximum speed of 15 cm/s using a joystick. The positions of the robot as calculated by the robot itself and as measured by the laser scanner were stored in a file. The experiment took about 18 minutes, resulting in approximately 5300 measurements.

The result was an average error of 10.5 cm, i. e. less than 4% of the width of the soccer field and less than 2.2% of its length. 60% of the measurements had an error less than this average. Figure 6a shows the path traveled and the errors made. Please note that this outcome is similar to the results presented in [4], with the two exceptions that Gutmann and Fox used color marks for localization, and that they performed their experiments on a small 3m×2m field. In addition, they worked on a log file, allowing them to optimally adjust the parameters of their algorithms, e. g. the Monte-Carlo localization approach used needed only 30 samples.

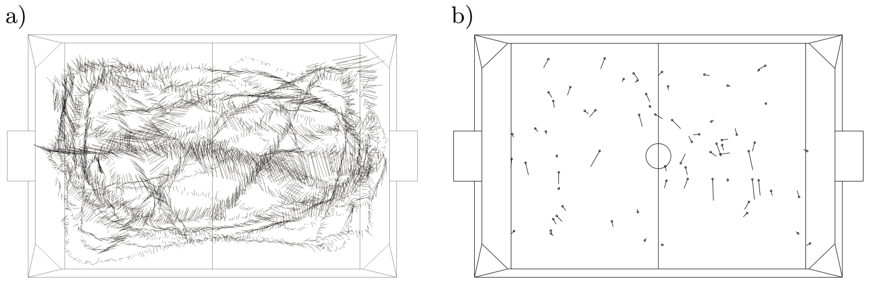

Fig. 6. Experimental results. Each line connects a position calculated by the robot with one determined by the laser scanner. a) First experiment. b) Second experiment.

4.3 Experiment 2

The goal of the second experiment was to evaluate the precision in reaching certain goal points. In this experiment, random goal positions were given to the robot. The system then performed the so-called *go-to-point* skill to reach the specified location. When the robot did not move anymore, the coordinates of the goal position, the position calculated by the robot, and the position measured by the laser scanner were stored in a file. In the experiment, 68 positions had to be reached.

There were two results: the average error between the goal position and the position reported by the laser scanner was 9.4 cm. 66% of the goals were reached with smaller deviations. However, the go-to-point skill does not reach the goal position precisely. It often stops one or two cm too early. Therefore, the average error between the position measured by the robot and the position measured by the laser sensor is smaller, namely 8.4 cm. 60% of the goals were even reached with a smaller error. Figure 6b shows the 68 goal positions and the positions reached by the robot.

5 Conclusions and Future Work

This paper presents an approach for edge-based self-localization in the SFRL. It is based on a vision system that extracts edges without processing whole images. The localization method is a variant of the well known Monte-Carlo localization. While using only a small number of samples, it increases the stability of the localization by maintaining separate probabilities for different edge types for each sample. These probabilities are only adapted slowly. This results in a fast, robust, and precise self-localization of the robot, and it can be seen as a milestone for the SFRL, because it shows that a self-localization without the color beacons is possible.

However, the results presented in this paper only show that edge-based localization is possible for a robot that is alone on the field. Further experiments have to show whether the localization method will also work during actual RoboCup games. While the recognition of the edge points is quite robust, the head cannot swing from left to right and back during actual soccer games, because the robot has to track the ball. Therefore, the situation is different, and it requires for a suitable control strategy for the head posture.

6 Acknowledgments

The authors would like to thank all members of the GermanTeam for providing the basis for their research, namely Martin Lötzsch, Uwe Düffert, and Max Risler.

References

1. J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, pages 2061–2066, 2000.
2. F. Dellaert, W. Burgard, D. Fox, , and S. Thrun. Using the condensation algorithm for robust, vision-based mobile robot localization. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.
3. D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.
4. D. Gutmann, J.-S. Fox. An experimental comparison of localization methods continued. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002. EPFL.
5. Robert Hanek, Thorsten Schmitt, Sebastian Buck, and Michael Beetz. Fast image-based object localization in natural scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002, Lausanne*, 2002.
6. M. Jamzad, B. Sadjad, V. Mirrokni, M. Kazemi, H. Chitsaz, A. Heydarnoori, M. Hajiaghai, and E. Chiniforooshan. A fast vision system for middle size robots in robocup. In *5th International Workshop on RoboCup 2001 (Robot World Cup Soccer Games and Conferences)*, number 2377 in Lecture Notes in Computer Science, pages 71–80. Springer, 2002.
7. M. Jüngel, J. Hoffmann, and M. Lötzsch. A real-time auto-adjusting vision system for robotic soccer. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence. Springer, 2004. to appear.
8. S. Lenser and M. Veloso. Sensor resetting localization for poorly modeled mobile robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.
9. F. K. H. Quek. An algorithm for the rapid computation of boundaries of run length encoded regions. *Pattern Recognition Journal*, 33:1637–1649, 2000.
10. S. Thrun, D. Fox, and W. Burgard. Monte carlo localization with mixture proposal distribution. In *Proc. of the National Conference on Artificial Intelligence*, pages 859–865. AAAI, 2000.
11. J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.