Particle-Filter-Based Self-Localization Using Landmarks and Directed Lines*

Thomas Röfer[1], Tim Laue[1], and Dirk Thomas[2]

[1] Center for Computing Technology (TZI), FB 3, Universität Bremen
roefer@tzi.de, timlaue@tzi.de
[2] Fachgebiet Simulation und Systemoptimierung, Fachbereich Informatik, Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany
dthomas@rbg.informatik.tu-darmstadt.de

Abstract. The paper presents the self-localization approach used by the World Champion in the Sony Four-Legged Robot League 2004. The method is based on a particle filter that makes use of different features from the environment (beacons, goals, field lines, field wall) that provide different kinds of localization information and that are recognized with different frequencies. In addition, it is discussed how the vision system acquires these features, especially, how the orientation of field lines is determined at low computational costs.

1 Introduction

The Sony Four-Legged Robot League (SFRL) is one of the official leagues in RoboCup, in which a standardized robot platform is used, namely the Sony Aibo. The robots act completely autonomously. The main sensor of the Sony Aibo is the camera located in its head. The head can be turned around three axes (2× tilt, 1× pan), and the camera has a field of view of approximately 57° by 42°. The soccer field in the SFRL has a size of approximately 5m×3m. As the main sensor of the robot is a camera, all objects on the RoboCup field are color coded. There are two-colored beacons for localization (pink and either yellow or skyblue), the two goals are of different color (yellow and skyblue), the field is green, and the field lines as well as the field wall are white.

During actual RoboCup games, the beacons and goals are rarely perceived, especially by the robot that is handling the ball. Therefore it is advantageous if also the field lines and the field wall can be employed for localization. However, different features in the environment are recognized with different frequency and they provide different kinds of information usable for localization. Lines only provide localization information perpendicular to their orientation. The *field lines* are mostly oriented across the field, but the side lines of the penalty area also provide important information, especially for the goalie. The field lines are seen less often than the *field wall* that is surrounding the field. Therefore the

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

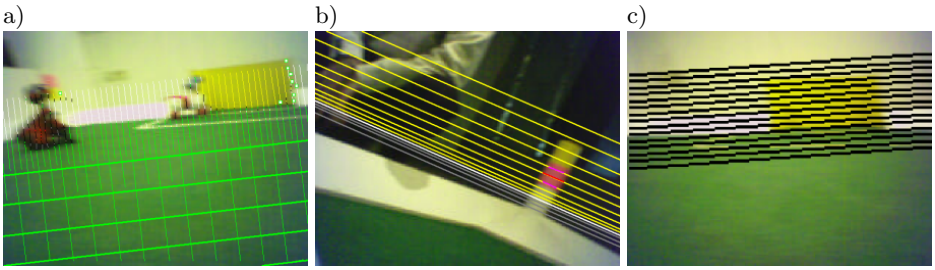

Fig. 1. Different scanlines and grids. a) The main grid which is used to detect objects on the field. b) The grid lines for beacon detection. c) The grid lines for goal detection.

latter 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 field wall is seen from nearly any location on the field. *Goals* and *beacons* are the only means to determine the orientation on the field, because the field lines and the field wall are mirror symmetric. Goals and beacons are seen only rarely. It turned out that the vision system is reliably able to determine the orientation of field lines, while the orientation of the edge between field wall and field is not as stable. Therefore, it is distinguished between field lines along the field and field lines across the field. This especially improves the localization of the goalie, because it sees both types of lines surrounding the penalty area.

2 Acquiring Localization Information

The vision system of the GermanTeam processes images of a resolution of 208×160 pixels, but looking only at a grid of less pixels. The grid is aligned to the so-called horizon, i. e. the plane that is in parallel to the field plane, but on the height of the camera. The idea is that for feature extraction, a high resolution is only needed for small or far away objects. In addition to being smaller, such objects are also closer to the horizon. Thus only regions near the horizon need to be scanned at a relative high resolution, while the rest of the image can be scanning using a wider spaced grid.

Each grid line is scanned pixel by pixel from top to bottom and from left to right respectively. During the scan each pixel is classified by color. A characteristic series of colors or a pattern of colors is an indication of an object of interest, e. g., a sequence of some orange pixels is an indication of a ball, a sequence of some pink pixels is an indication of a beacon, an (interrupted) sequence of sky-blue or yellow pixels followed by a green pixel is an indication of a goal, a sequence of white to green or green to white is an indication of an edge between the field and the border or a field line, and a sequence of red or blue pixels is an indication of a player. All this scanning is done using a kind of state machine; mostly counting the number of pixels of a certain color class and the number

of pixels since a certain color class was detected last. That way, beginning and end of certain object types can still be determined although some pixels of the wrong class are detected in between.

To speed up the object detection and to decrease the number of false positives, essentially three different grids are used. The main grid covers the area around and below the horizon. It is used to search for all objects which are situated on the field, i. e. the ball, obstacles, other robots, field borders, field lines, and the lower borders of the goals (cf. Fig. 1a). A set of grid lines parallel to and in most parts over the horizon is used to detect the pink elements of the beacons (cf. Fig. 1b and Fig. 2). The goal detection is also based on horizontal grid lines (cf. Fig. 1c).

As a first step towards a more color table independent classification, points on edges are only searched at pixels with a big difference of the Y channel of the adjacent pixels. An increase in the Y channel followed by a decrease is an indication of an edge. If the color above the decrease in the Y channel is sky-blue or yellow, the pixel lies on an edge between a goal and the field. The detection of points on field lines and borders is still based on the change of the segmented color from green to white or the other way round.

The differentiation between a field line and the border is a bit more complicated. In most 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. Therefore the pixel, where the segmented color changes back from white to green after a green-to-white change before, is assumed to lie on the ground. With the known height and rotation of the camera, the distance to that point is calculated. The distance leads to expected sizes of the field line in the image. For the classification, these sizes are compared to the distance between the green-to-white and the white-to-green change in the image to determine if the point belongs to a field line or a border. The projection of the pixels on the field plane is also used to determine their relative position to the robot.

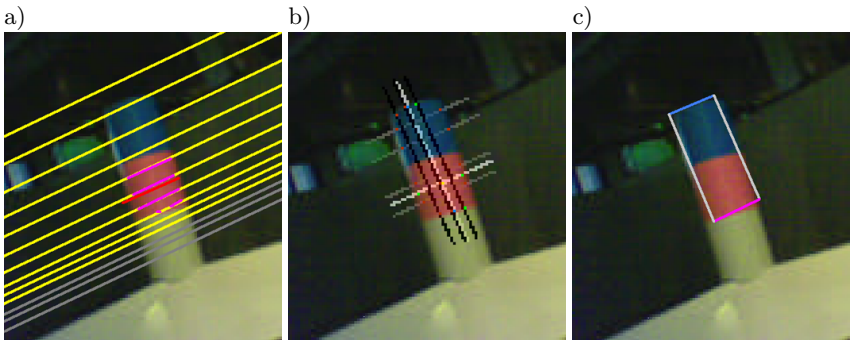

Fig. 2. Three steps in beacon detection: a) Scanlines searching for pink runs. The pink segments are the detected pink runs, the red segment is the result of clustering. b) The specialist detects the edges of the beacon. c) The generated percept.

In addition, for every point classified as being on the edge of a field line or the field wall, the gradient of the Y channel is computed (cf. Fig. 3a,b). This gradient is based on the values of the Y channel of the edge point and three neighboring pixels, using a Roberts operator ([3]):

$$\begin{aligned} s &= I\left[x+1\ y+1\right] - I\left[x\ y\right] \\ t &= I\left[x+1\ y\right] - I\left[x\ y+1\right] \\ |\nabla I| &= \sqrt{s^2+t^2} \\ \angle\nabla I &= \arctan\left(\tfrac{s}{t}\right) - \tfrac{\pi}{4} \end{aligned} \tag{1}$$

where $|\nabla I|$ is the magnitude and $\angle\nabla I$ is the direction of the edge. $\angle\nabla I$ is afterwards projected to the field plane, resulting in a direction in field coordinates.

3 Self-Localization

A Markov-localization method employing the so-called Monte-Carlo approach [1] is used to determine the position of the robot. It is a probabilistic approach, 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 this posture. Therefore, such particles mainly consist of a robot pose, i. e. a vector $pose = (\mathrm{x},\mathrm{y},\theta)$ representing the robot's x/y-coordinates in millimeters and its rotation θ in radians. A Markov-localization method requires both an observation model and a motion model. The observation model describes the probability for taking certain measurements at certain locations. The motion model expresses the probability for certain actions to move the robot to certain relative postures. The one used is described in [2].

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 for all particles are determined on the basis of the observation model for the current sensor readings, i. e. bearings on landmarks calculated from the

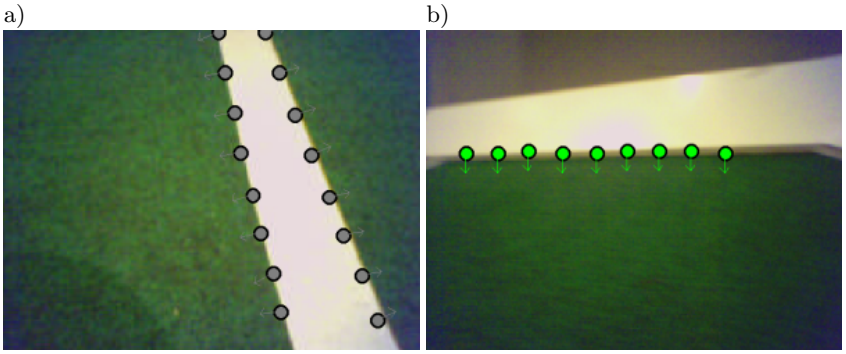

Fig. 3. Points on the edges, including computed gradients. a) On a line. b) On a field wall.

actual camera image. 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. Since the general approach has already been described in [2], this paper focuses on how to combine the perceptions of beacons, goals, and (directed) edge point in a way that results in a stable self-localization.

Observation Model. The observation model relates real sensor measurements to measurements as they would be taken if the robot were at a certain location. Instead of using the distances and directions to the landmarks in the environment, i. e. the beacons and the goals, this localization approach only uses the directions to the vertical edges of the landmarks. However, although the points on edges determined by the image processor are represented in a metric fashion, they are also converted back to angular bearings. The advantage of using landmark edges for orientation is that one can still use the visible edge of a landmark that is partially hidden by the image border. Therefore, more points of reference can be used per image, which can potentially improve the self-localization.

The utilized percepts are bearings on the edges of beacons and goals, and points on edges between the field and the field lines, the field wall, and the goals. These have to be related to the assumed bearings from hypothetical postures. As has been pointed out in [2], the different percepts contain different kinds of localization information and are seen with different frequencies. Therefore, it is required to represent separate probabilities for beacons and goals, horizontal field lines, vertical field lines, field walls, and goal edges for each particle.

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.

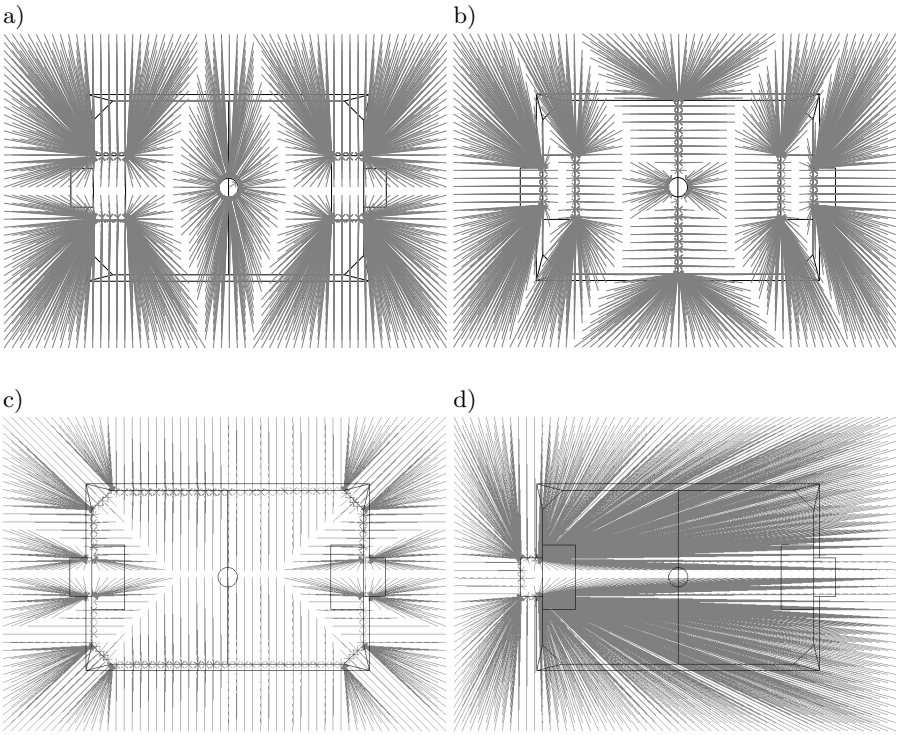

Fig. 4. Mapping of positions to closest edges. a) Field lines along the field. b) Field lines across the field. c) Field wall. c) A goal.

Probabilities for Beacons and Goals. The observation model only takes into account the bearings on the edges that are actually seen, i. e., it is ignored if the robot has *not* seen a certain edge that it should have seen according to its hypothetical posture 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 $\omega_{measured}$ and the expected angle $\omega_{expected}$ for a certain pose by applying a sigmoid function to the difference of both angles:

$$s(\omega_{measured}, \omega_{expected}) = \begin{cases} \mathrm{e}^{-50d^2} & \text{if } d < 1 \\ \mathrm{e}^{-50(2-d)^2} & \text{otherwise} \end{cases} \quad \text{where } d = \frac{|\omega_{measured} - \omega_{expected}|}{\pi} \tag{2}$$

The probability $q_{landmarks}$ of a certain particle is the product of these similarities:

$$q_{landmarks} = \prod_{\omega_{measured}} s(\omega_{measured}, \omega_{expected}) \tag{3}$$

Probabilities for Edge Points. The probabilities of the particles are 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:

$$s(\omega_{seen}, \omega_{exp}, \sigma) = \mathrm{e}^{-\sigma(\omega_{seen}-\omega_{exp})^2} \tag{4}$$

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:

$$q_{edge\ type} = s(\alpha_{seen}, \beta_{seen}, \alpha_{exp}, \beta_{exp}) = s(\alpha_{seen}, \alpha_{exp}, 10{-}9\frac{|v|}{200}) \cdot s(\beta_{seen}, \beta_{exp}, 100) \tag{5}$$

For the similarity of the vertical angles, the probability depends on the robot's speed v (in mm/s), because the faster the robot walks, the more its head shakes, and the less precise the measured angles are.

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 three points of each edge type (if detected) are 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 will take place.

The filtered probability q' for a certain type is updated ($q'_{old} \rightarrow q'_{new}$) for each measurement of that type:

$$q'_{new} = \begin{cases} q'_{old} + \Delta_{up} & \text{if } q > q'_{old} + \Delta_{up} \\ q'_{old} - \Delta_{down} & \text{if } q < q'_{old} - \Delta_{down} \\ q & \text{otherwise.} \end{cases} \tag{6}$$

For landmarks, $(\Delta_{up}, \Delta_{down})$ is $(0.1, 0.05)$, for edge points, it is $(0.01, 0.005)$

Overall Probability. The probability p of a certain particle is the product of the three separate probabilities for bearings on landmarks, edges of field lines along and across the field, the field wall, and goals:

$$p = q'_{landmarks} \cdot q'_{longitudinal\ lines} \cdot q'_{latitudal\ lines} \cdot q'_{field\ walls} \cdot q'_{goals} \tag{7}$$

4 Results

[2] presented quantitative results on the precision of the localization approach on an empty field using only lines and goals. The recent improvements clearly target to achieving a good localization during actual RoboCup games, i. e. in situations in which the main focus is on perceiving the ball and localization information is recognized rather rarely. Therefore, the games of the GermanTeam performed in Lisbon (videos at `http://www.tzi.de/4legged`) represent a good evaluation of how well the localization system works, because many parts of the behavior description of the GermanTeam rely on correct localization, e. g. which kick is selected at which position, and the placement of the defensive players, especially the goal keeper. At RoboCup 2003, it was also demonstrated that the GermanTeam can play soccer without the beacons.

5 Conclusions

This paper presents how beacons, goals, as well as points on edges between the field and field lines or field walls are determined, namely features that are required to localize on a RoboCup field. It is also shown how the edge points are augmented with the direction of the edge using a computationally cheap operation. All these features are used by a particle filter to determine the position of the robot. Here, separate probabilities for different features are used per particle, because the features provide different information about the position of the robot, and they are recognized with different frequencies.

Acknowledgments

The authors thank all members of the GermanTeam for providing the basis for this research.

References

1. 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.
2. T. Röfer and M. Jüngel. Fast and robust edge-based localization in the sony four-legged robot league. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, number 3020 in Lecture Notes in Artificial Intelligence, pages 262–273, Padova, Italy, 2004. Springer.
3. L.G. Roberts. Machine perception of three dimensional solids. *Optical and Electro-Optical Information Processing*, pages 159–197, 1968.