Soccer-Robot Localization Using Sporadic Visual Features

Stefan Enderle[1], Marcus Ritter[1], Dieter Fox[2], Stefan Sablatnög[1], Gerhard Kraetzschmar[1], Günther Palm[1]

[1]Dept. of Neural Information Processing
University of Ulm
D-89069 Ulm, Germany

[2]Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract. Knowing its position in an environment is an essential capability for any useful mobile robot. Monte-Carlo Localization (MCL) has become a popular framework for solving the self-localization problem in mobile robots. The known methods exploit sensor data obtained from laser range finders or sonar rings to estimate robot positions and are quite reliable and robust against noise. An open question is whether comparable localization performance can be achieved using only camera images, especially if the camera images are used both for localization and object recognition. Both problems can be easier solved if localization is based on features obtained by preprocessing images. In our image-based MCL approach, we combine visual distance features and visual landmark features, which have different characteristics. Distance features can always be computed, but have value-dependent and dramatically increasing margins for noise. Landmark features give good directional information, but are detected only sporadically. In our paper, we discuss the problems arising from these characteristics and show experimentally that MCL nevertheless works very well under these conditions.

1 Introduction

In the recent past, Monte-Carlo localization has become a very popular framework for solving the self-localization problem in mobile robots [1, 2]. This method is very reliable and robust against noise, especially if the robots are equipped with laser range finders or sonar sensors. In some environments, however, for example in the popular RoboCup domain [3], providing a laser scanner for each robot is difficult or impossible and sonar data is extremely noisy due to the highly dynamic environment. Thus, enhancing the existing localization methods such that they can use other sensory channels, like uni- or omni-directional vision systems, is a state-of-the-art problem in RoboCup. In this work, we present a vision-based MCL approach using visual features which are extracted from the robot's single camera and matched to a known model of the RoboCup environment.

2 Markov Localization

In Markov lokalisation [1], the position of a robot is estimated by computing the probability distribution over all possible positions in the environment. Let $l = \langle x, y, \theta \rangle$ denote a position in the state space. Then, $Bel(l)$ expresses the robot's belief in being at location l.

The algorithm starts with an initial distribution $Bel(l)$ depending on the initial knowledge of the robot's position. As the robot operates, two kinds of update steps are iteratively applied to incrementally refine $Bel(l)$:

Belief Projection across Robot Motion: This step handles robot motion and projects the robot's belief about its position across the executed actions. A *motion model* $P(l|l', m)$ is used to predict the likelihood of the robot being in position l assuming that it executed a motion command m and was previously in position l'. Here, it is assumed that the new position depends only on the previous position and the movement (Markov property). Using the motion model, the distribution of the robot's position belief $Bel(l)$ can be updated according to the commonly used formula for Markov chains [4]:

$$Bel(l) = \int P(l|l', m)\ Bel(l')\ dl' \tag{1}$$

Integration of Sensor Input: The data obtained from the robot's sensors are used to update the belief $Bel(l)$. For this step, a *observation model* $P(o|l')$ must be provided which models the likelihood of an observation o given the robot is at position l'. $Bel(l)$ is then updated by applying Bayes' rule as follows:

$$Bel(l) = \alpha\ P(o|l')\ Bel(l') \tag{2}$$

where α is a normalization factor ensuring that $Bel(l)$ integrates to 1.

The Markov localization method provides a mathematical framework for solving the localization problem. Unlike methods based on Kalman filtering [5], it is easy to use multimodal distributions. However, implementing Markov localization on a mobile robot in a tractable and efficient way is a non-trivial task.

3 Monte Carlo Localization

The Monte Carlo localization (MCL) approach [2] solves the implementation problem by representing the infinite probability distribution $Bel(l)$ by a set of N samples $S = \{s_i | i = 1 \ldots N\}$. Each sample $s_i = \langle l_i, p_i \rangle$ consists of a robot *location* l_i and *weight* p_i. The weight corresponds to the likelihood of l_i being the robots correct position, i.e. $p_i \approx Bel(l_i)$ [6]. Furthermore, as the weights are interpreted as probabilities, we assume $\sum_{i=1}^{N} p_i = 1$.

Initially, a set of samples reflecting initial knowledge about the robot's position is generated. During robot operation, the following two kinds of update steps are iteratively executed:

Sample Projection across Robot Motion: In MCL, a new sample set S is generated from a previous set S' by applying the motion model as follows: For each sample $\langle l', p' \rangle \in S'$ a new sample $\langle l, p' \rangle$ is added to S, where l is randomly drawn from the density $P(l|l', m)$. The motion model takes into account robot properties like drift and translational and rotational errors.

Belief Update and Weighted Resampling: According to Equation 2, all samples are re-weighted by incorporating the sensor data o and applying the observation model $P(o|l')$. Most commonly, sensors such as laser range finders or sonars, which yield distance data, are used. In this case, ideal sensor readings can be computed a priori, if a map of the environment is given. An observation model is then obtained by noisifying the ideal sensor readings, often simply using Gaussian noise distributions. Given a sample $\langle l', p' \rangle$, the new weight p for this sample is given by

$$p = \alpha \, P(o|l') \, p' \tag{3}$$

where α is again a normalization factor which ensures that all beliefs sum up to 1. These new weights for the samples in S' provide a probability distribution over S', which is then used to construct a new sample set S. This is done by randomly drawing samples from S' using the distribution given by the weights, i.e. for any sample $s_i = \langle l_i, p_i \rangle \in S'$,

$$Prob(s_i \in S) \approx p_i \tag{4}$$

The relationship is approximate only, because after each update step we add a small number of uniformly distributed samples, which ensure that the robot can re-localize in cases where it lost track of its position, i.e. where the sample set contains no sample close to the correct robot position.

4 Vision-Based Localization

There are mainly two cases where MCL based on distance sensor readings cannot be applied: (i) If distance sensors like laser range finders or sonars are not available. (ii) If the readings obtained by these sensors are too unreliable, e.g. in a highly dynamic environment. In these cases, other sensory information must be used for localization. A natural candidate is information from the visual channel, because many robots include cameras as standard equipment. An example for using visual information for MCL has been provided by Dellaert et al. [7].

An interesting and open question is whether the MCL approach still works when the number of observations is significantly reduced and when particular observations can be made only intermittently. In the following, we describe a situation where these problems arise and show how to adopt the MCL approach accordingly.

4.1 The RoboCup Domain

In RoboCup, two teams of robots play soccer against each other, where each team has three field players and one goalie. The field is about 9m long and 5m wide and has two goals which are marked blue and yellow for visual recognition. The field has a green floor and is surrounded with a white wall. It has four corners each of which is marked with two green vertical bars of 5cm width.

4.2 Feature-Based Modeling

As described in Eq. 3, the sensor update mechanism needs a sensor model $P(o|l)$ which describes how probable a sensor reading o is at a given robot location l. This probability is often computed by estimating the sensor reading $\tilde{o}$ at location l and determine some distance $dist(o, \tilde{o})$ between the given measurement o and the estimation $\tilde{o}$.

If sensor readings o_i are camera images two problems arise: (i) Estimating complete images $\tilde{o_i}$ for each sample's location is computationally very expensive. (ii) Finding a distance measure between images yielding a sensor model $P(o|l)$ is quite difficult.

To overcome these problems, we use a feature-based approach that allows to perform both the estimation step and the distance computation on a much lower dimension. The selection of the visual features is guided by several criteria: (i) uniqueness of the feature in the environment, (ii) computational effort needed to detect the feature in images, and (iii) reliability of the feature detection mechanism. After evaluating various feature detectors on our robots and taking the above criteria into consideration, we decided to use the following visual features: goal posts, corners, distances to field edges. Note, that the corner detector yields ambiguous landmarks, because corners are indistinguishable from each other, while the four goal post features are unique landmarks.

In order to incorporate sensor information on a feature-based level, two steps are needed: (i) The features have to be extracted from the current camera image and (ii) the samples' weights have to be adapted according to some model $P(o|l)$. The used model is described in the following sections.

Feature Detection: In a first step the camera image is segmented in order to simplify the feature detection process [8]. For example, in Figure 1 the original camera image as well as the segmented image are shown. Based on the segmented image, we use different kinds of feature detectors. All of them apply a filter on the segmented image for detecting a respective color discontinuity.

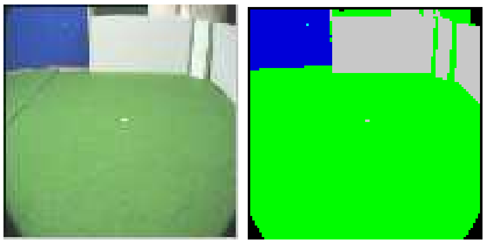

Figure 1: Original camera image (left) and segmented image (right).

The *goal post detector* detects a horizontal *white-blue* or a *white-yellow* transition for the blue or yellow goal post, respectively (see left image in Figure 2). The *corner detector* searches for a horizontal *green-white-green* transition in the image (middle image in Figure 2). The *distance estimator* estimates the distance to the field edges based on detected vertical *green-white* transitions in the image. At the moment, we select four specific columns in the image for detecting the field edges (right image in Figure 2). Note, that the correct distance to the blue goal is also detected.

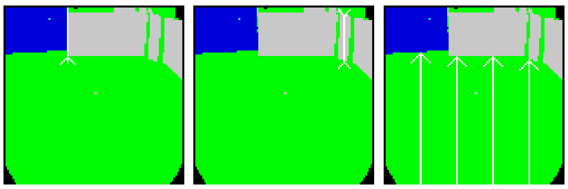

Figure 2: Detection of post, corner and edge features.

Weight Update As described in Eq. 3, the sample set must be re-weighted using a sensor model $P(o|l)$. Let the sensor data o be a vector of n features $f_1 \ldots f_n$. If we assume that the detection of features depends solely on the robot's position and does not depend on the detectability of other features, then the features are independent and we can conclude:

$$P(o|l') = P(f_1 \dots f_n|l') \quad (5)$$
$$= P(f_1|l') \dots P(f_n|l') \quad (6)$$

Note that n may vary, i.e. it is possible to deal with varying numbers of features.

The sensor model $P(f_i|l)$ describes how likely it is to detect a particular feature f_i given a robot location l. In our implementation, this sensor model is computed by comparing the horizontal position of the detected feature with an estimated position as determined by a geometrical world model. The distance between the feature positions is mapped to a probablility estimate by applying a heuristic function as illustrated in Figure 3.

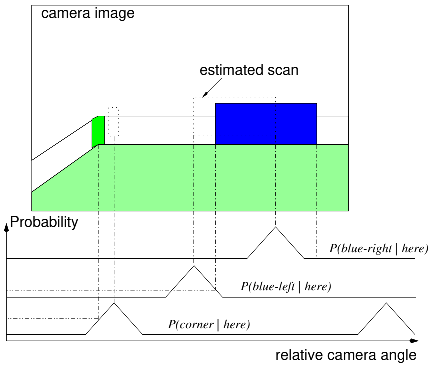

Figure 3: Heuristics for estimating the probabilities $p(f_i|l)$ from the current camera view.

The application of these heuristics to the examples used in Figure 2 are illustrated in Figure 4. Probabilistic combination of evidence for several features yields significantly better results, as convincingly demonstrated by the examples in Figure 5.

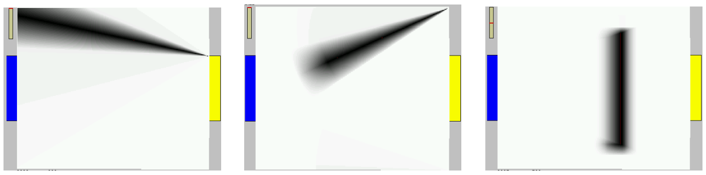

Figure 4: Probability distributions $P(f_i|l)$ for all field positions l (with fixed orientation) given the detection of a particular feature f_i (from left to right: goal posts, corners and distance measurements).

The figure illustrates various properties of our approach. First, the heuristic function for each of the four unique goal post features has only a single peak. The shape of the function causes all samples with comparatively high angular error to be drastically down-valued and successively being sorted out. Thus, detecting a single goal post will reshape the distribution of the sample set such that mostly locations that make it likely to see a goal post in a certain direction will survive in the sample set. Furthermore, if two goal posts are detected at the same time, the reshaping effect will be overlayed

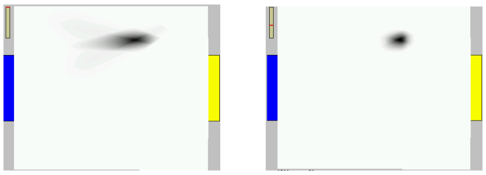

Figure 5: Resulting probability distributions after combining corner and goal features (left image) and all features (right image).

and drastically focus the sample set. Secondly, there is only a single heuristic function that captures all of the ambiguous corner features. Each actually detected corner is successively given a probability estimate. If the corner detector misses corners that we expected to see, this does not do any harm. If the detector returns more corners than actually expected, the behavior depends on the particular situation: detecting an extra corner close to where we actually expected one, has a small negative influence on the weight of this sample, while detecting a corner where none was expected at all has a much stronger negative effect. Note, that the robustness of the corner detector can be modelled by the heuristic function: Having a *robust* detector means assigning *very low* probabilities to situations where the distance between estimated and detected feature position is *large*, while for a *noisy* detector the heuristic function would assign *higher* probabilities in such situations. The effect of detecting too many features or detecting them at wrong positions is accordingly different.

5 Experiments

The described method was implemented and evaluated on a Sparrow-99 robot, a custom-built soccer platform equipped with an uni-directional camera (see Figure 6).

Figure 6: The Sparrow-99 robot.

Experiment 1: Number of Features

This experiment shows the usability of the described approach. It is shown that the robot can localize robustly and that the accuracy can be improved by adding more features.

A Sparrow-99 robot was placed in one corner of the field (the right top circle of Figure 7). In order to have an accurate reference path, we moved the robot by hand along a rectangular trajectory indicated by the dots.

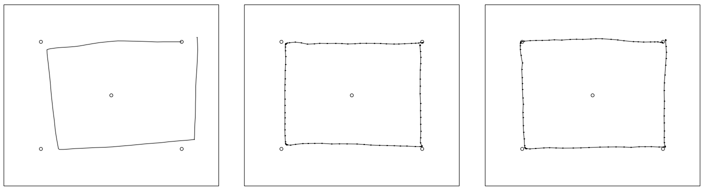

Figure 7: A single round of the Sparrow-99 being pushed along a rectangular path in the RoboCup field.(Odometrie path, corrected path using goal post features, corrected path using all features).

The first image in Figure 7 shows the trajectory given by the odometry when executing a single round. Note the displacement at the end of the tour. The second image displays the corrected path found by the localization algorithm using only the goal posts as features. In the third image we can see a highly accurate path found by the localization algorithm which used all possible features.

Figure 8 shows the drift that occurs when moving multiple rounds. The second image again displays the corrected trajectory which does not drift away.

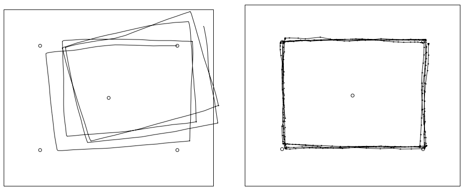

Figure 8: Four rounds of the Sparrow-99 being pushed along a rectangular path in the RoboCup field.(Odometrie path drifts away (left), the corrected path is stable).

Experiment 2: Number of Samples

Implementing sample-based localization, it is important to have an idea of how many samples you need. Obviously, a small number of samples is preferred, since the computational effort increases with the number of samples. On the other side, an appropriate number of samples is needed in order to achieve the desired accuracy. In this experiment, we moved four rounds exactly as in the previous experiment. This time, we used five different numbers of samples: 50, 100, 150, 1000, 5000

In the left image of Figure 9, the average localization errors of the different sample numbers are shown. One can see that the error decreases when more samples are used. On the other hand, the difference in accuracy does not increase very much from 150 samples over 1000 to 5000.

The right image of Figure 9 shows both the average error and the maximum error of the same five runs (50,100,150,1000 and 5000 samples). Again, you can see that above 150 samples, the accuracy hardly increases.

6 Conclusions

In this paper, we used the Monte-Carlo approach for vision-based localization of a soccer robot on the RoboCup soccer field. Unlike many previous applications, the robot could

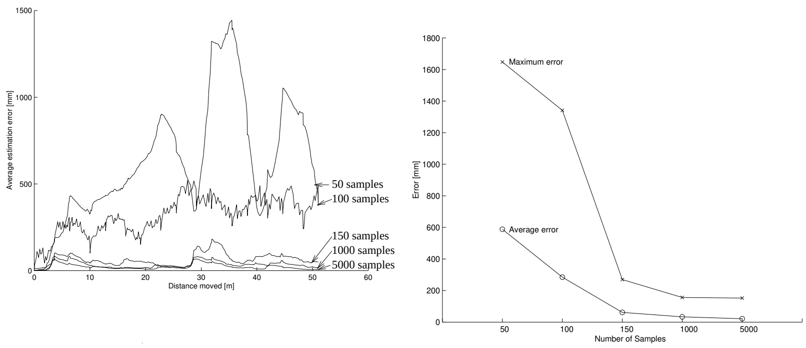

Figure 9: Left: Average localization errors of the different sample numbers. Right: Average error and maximum error of the same five runs (50,100,150,1000 and 5000 samples used).

not use distance sensors like laser scanners or sonars. Also, special camera setups were not available. Instead, the onboard camera was used for localization purposes in addition to object recognition tasks. As a consequence, sensor input to update the robot's belief about its position was low-dimensional and sporadic. Nevertheless, the experimental evaluation demonstrated that Monte Carlo localization works quite well even under these restrictive conditions.

We could also show that even with a small number of detected features which leads to sporadic vision updates, the localization results can be usable. However, by increasing the number of visual features the accuracy enhances dramatically.

References

[1] Dieter Fox, *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*, Ph.D. thesis, University of Bonn, Bonn, Germany, December 1998.

[2] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *In Proc. of the National Conference on Artificial Intelligence*. AAAI, 1999.

[3] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara, "Robocup a challenge problem for ai," *AI magazine*, vol. 18, no. 1, pp. 73–85, 1997.

[4] K. Chung, *Markov chains with stationary transition probabilitites*, Springer, 1960.

[5] Peter S. Maybeck, *The Kalman Filter: An Introduction to Concepts*, pp. 194 – 204, Springer, 1990.

[6] A.F.M. Smith and A.E. Gelfand, "Bayesian statistics without tears: A sampling-resampling perspective," *American statistican*, vol. 46, no. 2, pp. 84–88, 1992.

[7] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Proceedings of CVPR-99*, 1999.

[8] Alexander Neubeck, Marcus Ritter, Dominik Maschke, Pierre Bayerl, Stefan Enderle, Stefan Sablatnög, Gerhard Kraetzschmar, and Günther Palm, "Visual feature detection for robot localization and object recognition in RoboCup," 2000, technical report, to appear.