Vision-based Localization in RoboCup Environments

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. In this 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 [4, 5]. 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 [6], 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 unidirectional camera and matched to a known model of the RoboCup environment.

2 Monte Carlo Localization

Monte Carlo localization (MCL) [5] is an efficient implementation of the general Markov localization approach (see e.g. [4]). Here, the infinite probability distribution $Bel(l)$ expressing the robot´s belief in being at location l is represented by a set of N samples $S = \{s_1, \ldots, s_N\}$. Each sample $s_i = \langle l_i, p_i \rangle$ consists of a

P. Stone, T. Balch, and G. Kraetzschmar (Eds.): RoboCup 2000, LNAI 2019, pp. 291-296, 2001.

robot *location* l_i and *weight* p_i. As the weights are interpreted as probabilities, we assume $\sum_{i=1}^{N} p_i = 1$.

The algorithm for Monte Carlo localization is adopted from the general Markov localization framework. 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: As in the general Markov algorithm, a motion model $P(l|l', m)$ is used to update the probability distribution $Bel(l)$. 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 $P(l|l', m)$.

Observation Update and Weighted Resampling: Sensor inputs are used to update the robot's beliefs about its position. All samples are re-weighted by incorporating the sensor data o and applying the observation model $P(o|l')$. Given a sample $\langle l', p' \rangle$, the new weight p for this sample is given by

$$p = \alpha \, P(o|l') \, p' \tag{1}$$

where α is a normalization factor which ensures that all beliefs sum up to 1. These new weights for the samples in S' provide a probability distribution, 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.

3 Vision-Based Localization

In RoboCup where laser range finders are often not available and sonar data are too unreliable due to the highly dynamic environment, a natural candidate is 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. [1].

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 show how to adapt the MCL approach in order to overcome these problems.

3.1 Feature-Based Modeling

As described in Eq. 1, 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 $\tilde{o}$.

As it is not possible to efficiently estimate complete camera images and then compute image distances for hundreds of samples, we use a feature-based approach. After evaluating various feature detectors on our robots (see [3]), we decided to use the following features: 1) goal posts of blue and yellow goal, 2) corners, and 3) distances to field edges.

Feature Detection: In a first step the camera image is segmented in order to simplify the feature detection process. Based on the segmented image, we use different kinds of filters detecting a respective color discontinuity.

The *goal post detector* detects a vertical *white-blue* or a *white-yellow* transition for the blue or yellow goal post, respectively (see left image in Figure 1). The *corner detector* searches for vertical *green-white-green* transitions in the image (middle image in Figure 1). The *distance estimator* estimates the distance to the field edges based on detected horizontal *green-anything* transitions in the image. At the moment, we select four specific columns in the image for detecting the field edges (right image in Figure 1).

Fig. 1. Detection of post, corner and edge features.

Weight Update Let the sensor data o be a vector of n features $f_1 \dots f_n$. 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:

$$\begin{aligned} P(o|l') &= P(f_1 \dots f_n|l') \\ &= P(f_1|l') \dots P(f_n|l') \end{aligned} \tag{2}$$

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 2.

The application of these heuristics to the examples used in Figure 1 are illustrated in Figure 3. Probabilistic combination of evidence for several features

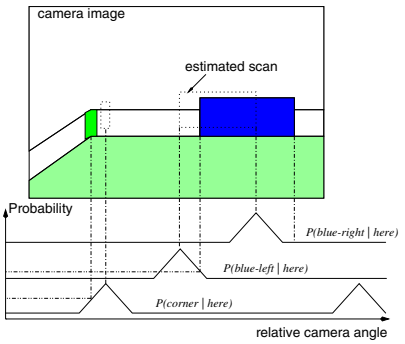

Fig. 2. Heuristics for estimating the probabilities $p(f_i|l)$ from the current camera view.

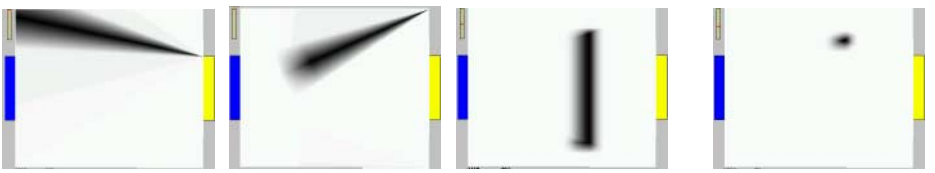

Fig. 3. Probability distributions $P(f_i|l)$ for all field positions l (with fixed orientation) given the detection of a particular feature f_i or all features, respectively. From left to right: goal posts, corners, distance measurements, all three combined.

yields significantly better results, as convincingly demonstrated by the rightmost image in Figure 3.

The figure illustrates various properties of our approach. 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. 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.

4 Experiments

The described method was implemented and evaluated on a Sparrow-99 robot, a custom-built soccer platform equipped with an uni-directional camera [2].

Experiment 1: Number of Features In this experiment, we show 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 4). In order to have an accurate reference path, we moved the robot by hand along a rectangular trajectory indicated by the dots.

The first image in Figure 4 displays the odometry data when moving four rounds and shows the drift error that occurs. The second image displays the corrected trajectory which does not drift away. The third image displays the first round corrected by the localization algorithm using only the goal posts as features. In the fourth image we can see a more accurate path found using all three feature types.

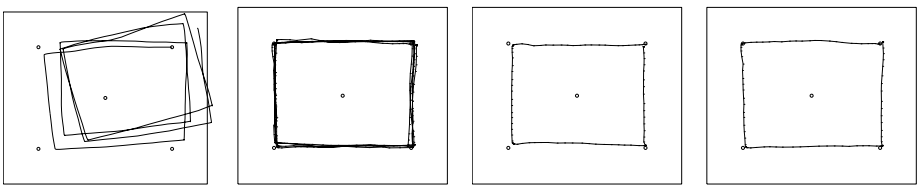

Fig. 4. A single round of the Sparrow-99 being pushed along a rectangular path in the RoboCup field.(Odometrie path, corrected path using goal post features, corrected path using all features).

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 5, 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 5 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.

5 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,

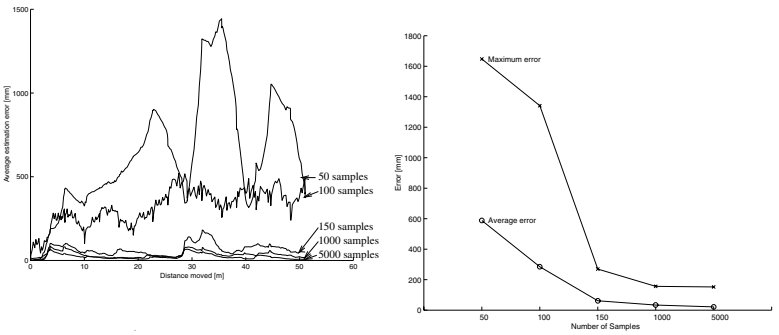

Fig. 5. Left: Average localization errors of the different sample numbers. Right: Average error and maximum error of the same five runs (50,100,150,1000 and 5000 samples used).

the robot could 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 well even under these restrictive conditions.

We could also show that even with a small number of detected features leading to sporadic observation updates, the localization results are usable. However, by increasing the number of visual features the accuracy enhances dramatically.

References

[1] Frank Dellaert, Wolfram Burgard, Dieter Fox, and Sebastian 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*, 1999.

[2] Stefan Enderle. The sparrow 99 robot. Technical report, University of Ulm, 1999. internal report.

[3] Stefan Enderle, Marcus Ritter, Dieter Fox, Stefan Sablatnög, Gerhard Kraetzschmar, and Günther Palm. Soccer-robot locatization using sporadic visual features. *Proceedings of the IAS-6 International Conference on Intelligent Autonomous Systems*, 2000.

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

[5] 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.

[6] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup a challenge problem for ai. *AI magazine*, 18(1):73–85, 1997.