Line Based Robot Localization under Natural Light Conditions

Artur Merke[1] and **Stefan Welker**[2] and **Martin Riedmiller**[3]

Abstract. In this paper we present a framework for robot selflocalization in the robocup middle size league. This framework comprises an algorithm for robust selflocalization and a set of benchmarks which can be used offline to test other algorithms and to compare their outcomes with our results. The algorithm is part of our competition team Brainstormers-Tribots which won the Robocup German Open 2004. This is a multi agent real time environment, therefore our algorithm is prepared to work with 30 frames per second, leaving enough time for other tasks like robot control or path planning. Our approach uses a particle filter method relying on features found in the image. The features are points on field lines. They can be recognized reliably under natural light conditions, so the is no longer a need for a well defined and constant light source. Also color coded landmarks or goals are not required for a stable selflocalization. We present results for different runs on our benchmark suite, which is an outdoor soccer field with the size of 16x10m. This field size is bigger then in current competitions and anticipates the trend of using larger fields in future competitions.

1 Introduction

Since 1996, when the first Robocup competition took part, there was a steady pursuit of making the Robocup environment more realistic and less artificial. Certainly one can argue that there were not enough changes in this direction, as the games are still conducted under well defined artificial floodlight. Also different color coded landmarks are used for self-localization on the field. The use of such color coded landmarks strongly relies on color classifiers, which are very sensitive to external light conditions. So to get a system which works under natural light condition one has to extract more shape oriented features from the images.

In this paper we present a method which relies on easily extractable shape information, which can be robustly recognized under different light conditions. Our method uses particle filtering and is a significant extension of the method used in our Brainstormers-Tribots team in 2003. The old method worked well and our team scored 5 wins, 1 draw and conceived 2 defeats (scoring 26:8 goals altogether) in the world championships in Padova 2003. But there were also foreseeable limitations. The old method relied on color coded poles and goals. For example distant poles were easily overlooked (they can be few pixel large due to the geometry of the mirror) or other colored objects could be mistaken for poles or goals.

In our new method we only detect points on lines along rays radially arranged around the center of the omni-directional camera image. As such points can be recognized without highly tuned color classification (see section 4 for more details) we were able to conduct different benchmarks under natural light conditions. Also such features are insensitive to varying surroundings, so for example the robot cannot get misled by different colored objects in the audience. Another advantage is that we can now self localize on larger fields, as we don't rely on specific distant and therefore small features. Using particle filter methods enables us to estimate the position of the robot very exactly on a 16x10 meters large outdoor field (see figure 1 in section 2). For example driving and turning the robot (using omni-directional drive) for 30 seconds across the field with a speed of 1.5 m/s, the self-localization deviates only approximately 20 cm on average from the reference path (with maximal deviation of 50 cm). See section 5 for further results.

The second most important aspect of this paper is the reproducibility of the presented results. We compiled a set of 25 runs of our robot on an 16x10 meters large outdoor field. Each run consists of all image information (e.g. 30 fps), the gathered odometry data, and externally measured reference robot positions during the whole run. These reference positions were measured with a laser scanner positioned outside the field, and can be used to evaluate the accuracy of the used algorithms.

To our knowledge it is the first benchmark in the Robocup middle size league for self localization. We hope this will encourage other researchers (also such which haven't yet participated in Robocup, but are active in the machine vision field) to compare their algorithms against our benchmark. The setting of our benchmark excels current rules of the middle size league (larger field, natural light). We hope that this is a chance to test current algorithms for coming requirements, and that this also will accelerate the process of making the conditions in the middle size league more realistic.

2 Environment

In this section we describe the setting of our experiments. All experiments for this paper were conducted on an outdoor field with the length of 16 meters and the width of 10 meters, see figure 1. We have chosen a field covered with tartan, be-

[1] Universität Dortmund, Germany, artur.merke@udo.edu
[2] Universität Dortmund, Germany, stefan.welker@udo.edu
[3] Universität Osnabrück, Germany, martin.riedmiller@uos.de

Figure 1. Outdoor field, 16 meters long, 10 meters wide

cause on surfaces like asphalt or concrete the omni-directional wheels of our robot do not have enough grip and also fret extremely. Originally we looked for a field with the dimensions 16x12 meters but could not find a tartan field in such size without disturbing lines. As the fields in Lisboa in 2004 will have the dimensions of 12x8 meters we are still much ahead of the current Robocup requirements.

It was very important to us to use an outdoor field, so that we could demonstrate self-localization under natural light conditions. As it was quite difficult to find a large enough tartan field without disturbing lines it was even more difficult to find one covered with green tartan. Here we decided for a compromise, weighting the outdoor conditions and large size more then a specific surface color. Our field is therefore colored dark red with slight color intensity variations (the tartan is quite old and soiled). Meanwhile we consider the different surface color as an additional challenge, actually real soccer is also played on different fields not always lawn covered, but also covered with red ash (at least in the lower soccer leagues).

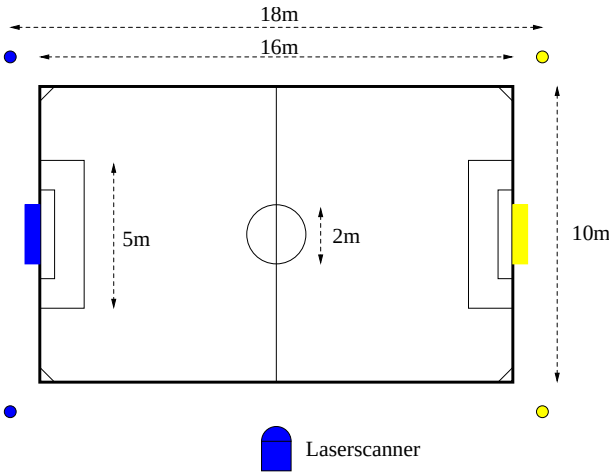


Figure 2. Field dimensions

The dimensions of the field and positions of lines, goals and poles conform to the current Robocup rules for the year 2004 [2], see figure 2. The poles and goals are not needed in our algorithm, but we positioned them on the field such that groups with other approaches could also use our benchmarks. The outer lines are 12 cm thick, all inner lines are 6 cm thick.

Outside the field we stationary positioned a laser range scanner. This enables us to scan the robot path for each run. Afterwards the scanned positions (each position has an unique time stamp) of the robot can be used to verify the obtained algorithmic results.

Figure 3. Tribot, omni-directional drive and camera

For all runs we used our standard competition robot - the Tribot. See figure 3 for a picture of the Tribot. It has an omni-directional drive and can for example drive towards the ball and rotate at the same time (see [1] for more details). During a run the robot records camera images and odometry data. In the collected odometry data we abstract from this particular drive, and only collect the velocity in direction x and y and the rotation velocity ϕ of the robot. So using the benchmark one is not confined to this particular robot drive.

The images recorded by the robot are omni-directional, because the camera captures the reflections of the field produced by a hyperbolical mirror (see [1] for more details). Due to a calibration process one can compute for every pixel in the image the corresponding real world distance. This matches the reality as long the recognized objects stand on the ground. The accuracy of the distance estimation decreases significantly with the distance from the robot. In figure 4 one can see two separate images from the camera, where one can also clearly recognize the different light conditions.

3 Benchmarks

At the moment our benchmark suite consists of 25 different robot runs. Currently each run is at least 11 and at most 35 seconds long. Using a frame rate of 30 frames per seconds and storing the images uncompressed in VGA resolution it requires between 200 and 600 megabyte disk space per run.

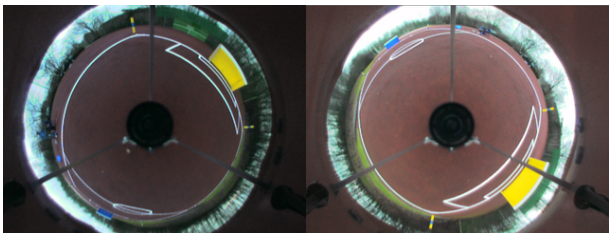


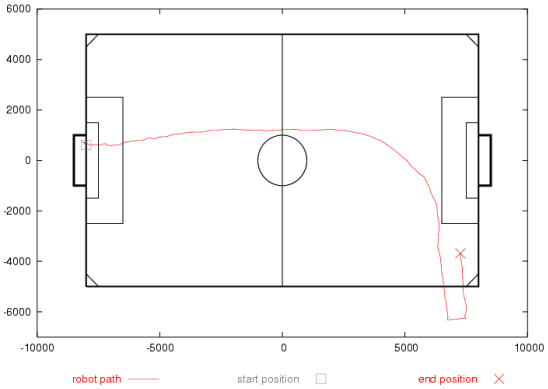
Figure 4. Two camera images showing different light conditions.

In figure 5 we depicted four exemplary robot runs, see [8] for a complete list. For example in figure 5 (a) the robot starts in front of the blue (=left) goal, drives across the whole field almost crossing the right penalty area, exits the field near the right bottom pole and enters it again driving toward the yellow goal. The sequence is 23 seconds long, consisting of 350 raw images (approx. 200 megabyte) taken with the frame rate of 15 frames per second. During the run approximately 4 laser measurements per second were recorded, resulting altogether in approx. 100 reference positions. These externally measured reference positions can be used to measure the quality of the deployed algorithms. As another example the run in figure 5 (d) is 35 seconds long, consists of 1050 raw images (because of the doubled frame rate) with approximately 160 reference positions.

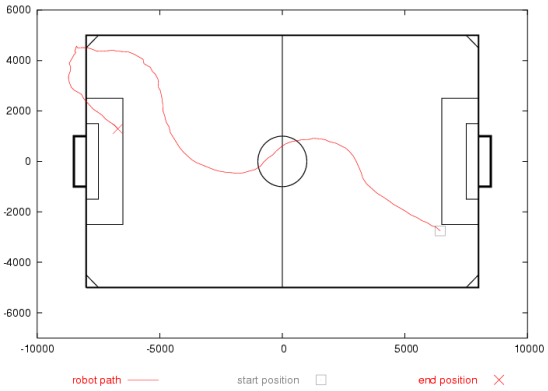
The collection of benchmarks is supposed to grow in the future. The current set of 25 benchmarks can be found at [8]. At this URL we also provide source code for reading and showing the raw images and related data. We hope that this will encourage other teams to use our benchmark suite and maybe also to contribute in extending the existing data base.

4 Algorithm

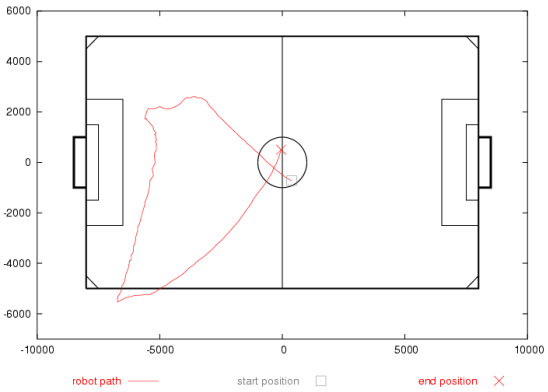
For localizing our robot in the Robocup environment we essentially rely on a camera as the primary sensor. A camera image provides very significant input data, a human can easily estimate the position of a robot by looking at the image. On the other hand a camera image can contain a lot of useless, even obstructive data such as image noise, light artifacts , color shift, brightness or camera shutter issues. It can be a difficult task to find an algorithm that recognizes meaningful features in an image, which can be used for localization. Most of the time a trade-off has to be made between speed and reliability. In the past most approaches were based on recognizing the color coded landmarks in the Robocup environment. Due to lightning conditions it can be hard to classify pixels safely to different color classes such as blue, yellow, green, orange or black. Therefore it is more robust to rely on shape oriented features. We decided to use the white field lines for localization, which can be recognized under varying light conditions. As our incremental algorithm makes do with even a small number of such features, lines occluded by obstacles and overlooked distant lines do not present a problem for our approach.



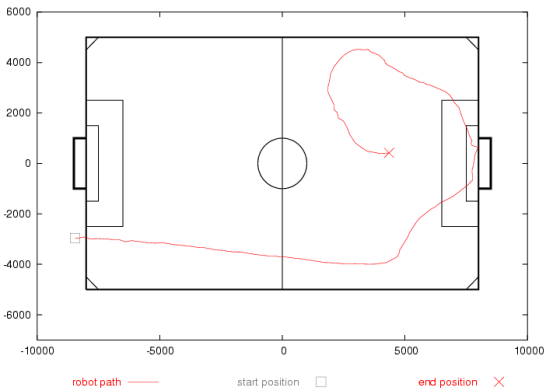
(a) speed 1.5 m/s, 15 fps



(b) speed 1.5 m/s, 15 fps



(c) speed 1.1 m/s, 30 fps



(d) speed 1.5 m/s, 30 fps

Figure 5. Example of different recorded robot paths.

4.1 Vision Architecture

With our vision system we present a fast detection method for pixels in the image that belong to the field lines. Our algorithm does not recognize the location and direction of the lines and corners in the image. Also color coded landmarks like poles and goals are not necessary. We do not use these additional features because our self-localization algorithm uses particle filtering which is a probabilistic method and the points on white lines are sufficient to get good and robust localization results.

To gather samples of points on lines in the image, we scan the image along several scan lines. These lines are radially arranged around the center of the omni-directional camera image. See figure 6 for an arrangement of these scan lines.

To recognize a line crossing along a scan line we search for significant variations in the color values. The variations are measured using an euclidean distance function in the YUV color space. By applying a threshold to these distances, we detect possible color transitions. Two consecutive transitions are recognized as a line transition if they are in close real world proximity to each other and the color before and after the transition show only a small color distance. This process gathers all kinds of line transitions in the image. To sort out transitions that do not belong to field lines an additional color validation is conducted. The deployed color classifier does not need to be non-ambiguous. The color classes may overlap and can therefore be tolerant enough cope with changes in light conditions during the classifying process.

The recognition of line transitions along the scan lines has the additional advantage of using only small amounts of computational resources. The image does not have to be segmented as a whole. This allows to run the system with 30 frames per second at a resolution of 640 x 480 pixels.

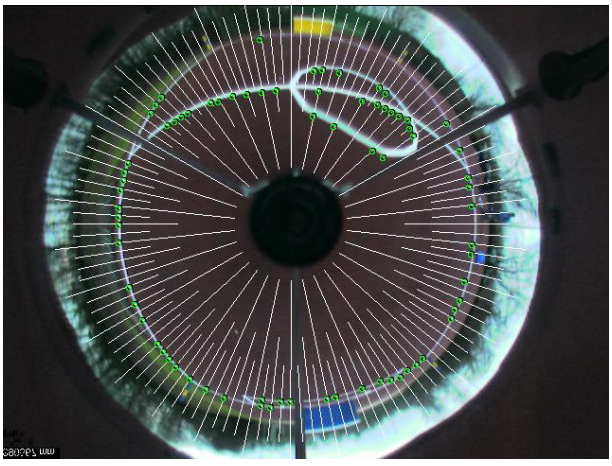


Figure 6. Analysis of an image with scan lines and recognized line transitions

Each line transition is a possible sensor measurement. To make it meaningful, we need its distance and angle to the center of the robot. To get the position of a pixel in the image in real world coordinates, we calibrate a distance mapping $D(r, \varphi)$ into every direction φ of the robot. The process of calibration is partially automated. A distance calibration environment has to be set up before the process. It consists of several color coded markings that are located at defined distances from the robot. While turning slowly, the robot recognizes these markers to gather data about the real distances of pixels in the image. With this data a complete distance mapping can be calculated for every pixel of the image. This mapping is only meaningful for objects that are located on the ground.

4.2 Self-Localization

In order to localize a robot correctly an appropriate estimate of the robot position $x_t \in R^n$ at time t has to be found. In our setting the position consists of the coordinates of the robot on the field and its relative orientation ($n = 3$). We utilize a sequential Monte Carlo method to generate the posterior probability distribution $\pi_{t|t}$ of the robot state x_t with regard to the prior distribution $\pi_{t-1|t-1}$. The former estimate $\pi_{t-1|t-1}$ is incrementally updated using new odometry data a_t and sensor values y_t. This is done in two stages.

First a *prediction* step is conducted using the odometry:

$$\pi_{t|t-1}(\cdot) = \int_{R^{n_x}} \pi_{t-1|t-1}(dx_{t-1}) K(\cdot|x_{t-1}, a_t) \qquad (1)$$

where $K(\cdot|x_{t-1}, a_t)$ denotes the Markov transition kernel for *action* a_t.

Afterwards an *update* step is performed using the current sensor values:

$$\pi_{t|t}(\cdot) = [\int_{R^{n_x}} g(y_t|x_t)\pi_{t|t-1}(dx_t)]^{-1} * g(y_t|x_t)\pi_{t|t-1}(\cdot) \quad (2)$$

where $g(y_t|x_t)$ is the conditional probability density of the observed sensor values with respect to the estimated position. See [4] for more details.

In particle filtering the real probability distribution $\pi_{t|t}$ is represented by a discrete probability measure using a set of N (currently about 200) weighted particles. The steps from equation 1 and 2 imply the following procedure of sequential importance sampling and resampling steps. This process consists of

1. Predicting new positions for particles while incorporating action information i.e. odometry
2. Updating the particle probability weights by estimating sensors input probability
3. Normalizing the probability weights of the particles
4. Resampling from the particle distribution to get the posterior distribution

In the following we will elaborate on the particularities of the steps 1, 2 and 4 in our approach.

Step 1. To predict the particles position by action we add the odometry reading a_t that consists of $(x_{odo}, y_{odo}, \varphi_{odo})$ for each particle. To represent the uncertainty of the odometry reading, gaussian noise is added to the particle location, proportional to the length of the odometry.

Step 2. We estimate the probability of obtaining the captured camera image at the location of every particle. This is done using an approximation which only relies on the detected

line transitions. For this approximation a product of all single transition probabilities is a reasonable estimate. To compute the probability of a single line transition, the location of the transition is mapped to a point p in the global coordinate system using the orientation and position of the considered particle. The minimal distance of this point p to the existing lines determines the probability value for the transition.

As this has to be done for every line transition and every particle, it can be very resource consuming process. Therefore we use a precalculated two dimensional look-up table of the field that provides the distance of every location on the field to the lines in $O(1)$.

Additionally the probability value for the transition depends on the distance of the point p to the position of the particle. This is because the distance measurement error of the vision system increases significantly for distant objects.

Step 4. To make the algorithm work the particles have to be resampled. Resampling statistically multiplies or discards particles at each time step to adaptively concentrate particles on regions of high posterior probability. This process consists of drawing N new Particles from the existing ones according to the particle weights using a multinomial distribution. In general this requires $O(n \log n)$ but can be done in $O(n)$ according to [5].

With these steps we can achieve robust incremental knowledge of the robot position by determining the average of the particle positions and headings.

5 Results

In this section we present the results for the sequences introduced in section 3. In this test we assume that the initial position of the robot is known, but our algorithm also solves the global localization (kidnapped robot) problem.

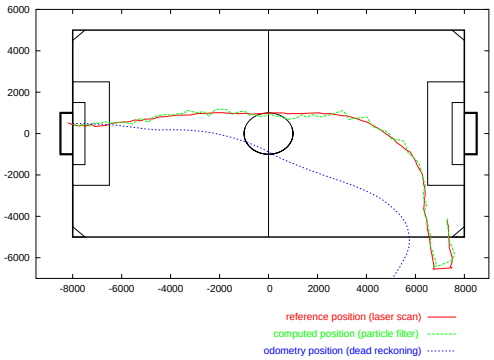


Figure 7. Sequence a, 15 frames/sec, 23 seconds

Sequence (a) (see Figure 7) shows the robot starting at the blue goal, driving across the whole field with a speed of 1.5 meters per second. The run is 23 seconds long and was captured at 15 frames per second. At the end of the run it leaves the field turns around and enters the field again. During this run the average deviation of the particle filter position to the laser scan position was only 18.1 cm, the maximum absolute error was 59.7 cm. In comparison to the large field size we only deviate by 1.8% of the field width and 1.1% with respect to the field length. This high quality of the result can be also seen in Figure 7 as the computed path lies very close to the reference position path. Also the deviation of the self-localization is uniformly accurate across the whole field, not only in regions close to lines. In contrast to the self-localization by particle filter the dead reckoning position obtained by the odometry measurements deviates very quickly from the real path and cannot be relied on for self-localization purposes.

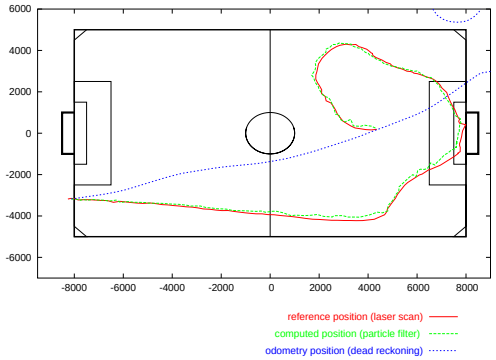


Figure 8. Sequence b, 30 frames/sec, 35 seconds

Sequence (d) (see Figure 8) shows a run with 30 frames per second and 35 seconds length. The robot starts at the side of the blue goal, drives across the field into the yellow goal region and back into the field while turning. Again the average error is only 17.4 cm and the maximum deviation 51.6 cm. This test run performed a little bit better which may be because of the increased frame rate. This shows that even when only processing half of the available image data the localization is still sufficiently precise. In Table 1 we summarized all results

Table 1. Evaluation for test runs in figure 5

sequence	length	avg. error	max. error
run (a)	23 sec	0.152 m	0.359 m
run (b)	23 sec	0.390 m	0.847 m
run (c)	35 sec	0.177 m	0.500 m
run (d)	35 sec	0.205 m	0.502 m

for the benchmarks presented in section 3. In all test runs the particle filter shows similar small deviations from the real path.

5.1 Results under reduced view range

The above results were obtained on a field without obstacles. Our algorithm also does work with partly occluded lines,

which happens in real world applications, where obstacles can cover significant parts of the image. We could prove this in the Robocup German Open 2004, where our team won the competition (winning all its 8 games, scoring 44:3 goals). The robots didn't delocalize during the matches, although the lines were covered by 7 other robots (3 teammates and 4 opponents) and a human referee.

As our current benchmark suite does not yet include sequences with obstacles (such sequences will be included in near future), we simulate an occluded view range using artificial black areas in the images. To this end we use different bitmap masks which reduce the original image information. A mask with four black areas can seen in figure 10. This mask constantly occludes two thirds of the image.

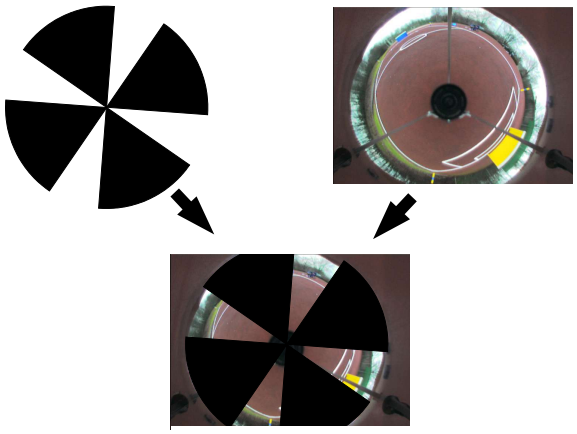


Figure 9. Reduction of the view range

As our algorithm also makes do with few lines, the results are only slightly worse than in the case without occlusions. This can be seen in table 2.

Table 2. Evaluation for test runs in figure 5 with occluded view range using mask shown in figure 9

sequence	length	avg. error	max. error
run (a)	23 sec	0.161 m	0.354 m
run (b)	23 sec	0.393 m	0.877 m
run (c)	35 sec	0.195 m	0.466 m
run (d)	35 sec	0.231 m	0.537 m

5.2 Simulating a directed view range

An occlusion mask as used in the above section, can also be used to simulate the view range of a directed camera. This can be seen in figure 10.

Also the results for this case are very good concerning the fact that only the lines and no other landmarks were used. This is remarkable because due to the big size of the field

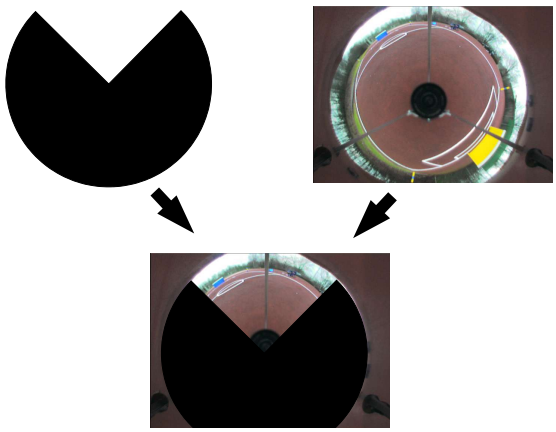


Figure 10. Simulation of directed view range

there are times where no lines at all are detected. But this short periods are compensated by the odometry. In figure 11 one can see such a period, where the robot leaves the left penalty area.

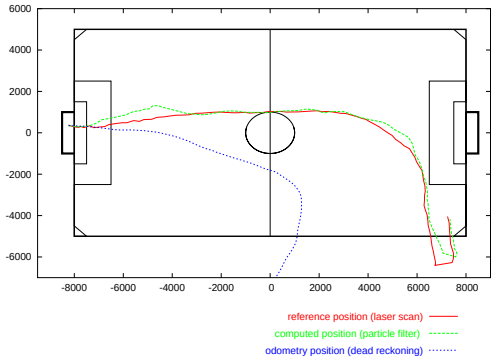


Figure 11. Sequence a, using directed view range

In table 3 we summarized the results for the directed view range case.

Table 3. Evaluation for test runs in figure 5 using directed view range from figure 10

sequence	length	avg. error	max. error
run (a)	23 sec	0.308 m	0.775 m
run (b)	23 sec	0.856 m	0.1764 m
run (c)	35 sec	0.274 m	0.687 m
run (d)	35 sec	0.352 m	0.837 m

6 Conclusions

In this paper we presented a framework for self-localization in the Robocup middle size league. This framework consists of

- a benchmark suite for testing self-localization algorithms and
- a newly developed algorithm for self-localization.

Using these freely available and reproducible benchmarks we presented the results for our algorithm. The obtained results show that our algorithm is very well suited for self-localization under natural light conditions. This is achieved without relying on color coded features like goals or poles and does work a 16x10 meters large outdoor field. Beside the presented benchmark results the algorithm was deployed in our competition team Brainstormers-Tribots, which won the Robocup German Open 2004.

The reason for the good performance of our algorithm lies in the combination of sequential Monte Carlo methods (particle filter) and the robust extraction of line features from the image data.

Both mentioned parts of the presented work are innovative. Benchmarks are extensively used in many fields of machine learning, but to our best knowledge our benchmark is the first considering self-localization of autonomous robots in the Robocup environment. The presented benchmarks are considered as a starting point and will be extended in the future (hopefully for non Robocup specific environments as well).

With respect to our algorithm and its outdoor deployment there are similar but distinct approaches in the literature. In [6] natural light conditions are considered, but only for color classification, no resulting self-localization performance tests were presented. In our work we do not heavily rely on color classification, as the main features are obtained from strong thresholds in color values which appear on line crossings. Color classification is also used, but just for validation of the obtained line crossings and can therefore be more fuzzy.

The work in [11] presents a robust self-localization algorithm for the middle size league. It relies on the detection of more complex line features in an homogeneously colored field. Our algorithm makes do with less structured features which we consider as one of the reasons of its robustness. Also the use of a particle filter distinguishes our work from [11]. It would be interesting to test the algorithm from [11] under the conditions of our benchmark suite.

Maybe the algorithm presented in [9] is most similar to ours. The extraction of line features is different, as we for example do not rely on separate detection of a horizon line. Also the deployment of the particle filter and the computation of sensor probabilities differ partially from our approach. The results presented in [9] were obtained on a quite small field (due to the Sony legged league limitations) and under artificial light conditions, therefore it would be interesting to see it's performance in our framework.

There are other approaches in the literature, see for example [10]. The methods used therein can also be distinguished from the work presented by us, but no qualitative comparisons were possible until now. We hope that our benchmark suite will be helpful in making such qualitative comparisons in the future. Also by using our challenging extensions of the current Robocup environment (natural light, large field), we hope to accelerate the progress in the Robocup environment.

The work presented in this paper would not be possible without the foundations created by the Brainstormers-Tribots team [1] in 2003. We would also like to thank the CoPS team from Stuttgart [3] for their support with the laser range scanner. Finally we thank our local sport facilities for providing the environment for our experiments.

REFERENCES

[1] M. Arbatzat, S. Freitag, M. Fricke, R. Hafner, C. Heermann, K. Hegelig, A. Krause, J. Krüger, M. Lauer, M. Lewandowski, A. Merke, H. Müller, M. Riedmiller, J. Schanko, M. Schulte-Hobein, M. Theile, S. Welker, and D. Withopf, 'Creating a robot soccer team from scratch: the brainstormers-tribots', in *RoboCup-2003 - Proceedings of the International Symposium*, (2003).

[2] M. Asada, T. Balch, A. Bonarini, A. Bredenfeld, S. Gutmann, G. Kraetzschmar, P. Lima, E. Menegatti, T. Nakamura, E. Pagello, F. Ribeiro, T. Schmitt, W. Shen, H. Sprong, S. Suzuki, and Y. Takahashi. Middle size robot league rules and regulations for 2004, http://www.tcsi.de/ROBOCUP/_DOCUMENTS/MSL/msl-rules-2004.pdf.

[3] T. Buchheim, G. Kindermann, R. Lafrenz, H. Rajaie, M. Schanz, F. Schreiber, and P. Levi, 'Team description - cops stuttgart', in *RoboCup-2003 - Proceedings of the International Symposium*, (2003).

[4] D. Crisan and A. Doucet, 'A survey of convergence results on particle filtering methods for practitioners', in *IEEE Transactions on Signal Processing*, (2002).

[5] Arnaud Doucet. On sequential monte carlo sampling methods for bayesian filtering, 1998.

[6] G. Mayer, G. K. Kraetzschmar, and H. Utz, 'Playing robot soccer under natural light: A case study', in *7th International Workshop on RoboCup 2003*, Lecture Notes in Artificial Intelligence. Springer, (2004).

[7] A. Merke and M. Riedmiller, 'Karlsruhe brainstormers - a reinforcement learning way to robotic soccer ii', in *RoboCup-2001: Robot Soccer World Cup V, LNCS*, eds., A. Birk, S. Coradeschi, and S. Tadokoro, 322–327, Springer, (2001).

[8] A. Merke, S. Welker, and M. Riedmiller. Benchmark suite for self-localization, http://lrb.cs.uni-dortmund.de/~merke/robocup/sloc.

[9] Th. Röfer and M. Jüngel, 'Fast and robust edge-based localization in the sony four-legged robot league.', in *7th International Workshop on RoboCup 2003*, Lecture Notes in Artificial Intelligence. Springer, (2004).

[10] E. Schulenburg, T. Weigel, and A. Kleiner, 'Self-localization in dynamic environments base on laser and vision data', in *International Conference on Intelligent Robots and Systems (IROS)*, volume 18, pp. 998–1004, (2003).

[11] F. v. Hundelshausen and R. Rojas, 'Tracking regions', in *7th International Workshop on RoboCup 2003*, Lecture Notes in Artificial Intelligence. Springer, (2004).