Efficient and Reliable Sensor Models for Humanoid Soccer Robot Self-Localization

Tim Laue [+1], Thijs Jeffry de Haas [#2], Armin Burchardt [#3], Colin Graf [#4],
Thomas Röfer [+5], Alexander Härtl [#6], Andrik Rieskamp [#7]

[+]*DFKI Bremen, Safe and Secure Cognitive Systems*
Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
[1]Tim.Laue@dfki.de, [5]Thomas.Roefer@dfki.de

[#]*Fachbereich 3 - Mathematik und Informatik, Universität Bremen,*
Postfach 330 440, 28334 Bremen, Germany
[2]jeffry@informatik.uni-bremen.de, [3]armin@informatik.uni-bremen.de,
[4]cgraf@informatik.uni-bremen.de, [6]allli@informatik.uni-bremen.de,
[7]rieskamp@informatik.uni-bremen.de

***Abstract*— Although the precise structure of the color-coded environment as well as different well-proven state estimation algorithms are known, self-localization in a humanoid soccer robot scenario remains a challenging task. Different problems arise, e. g., from an inaccurate proprioception, the sparsity of unique features, or the perception of false positives. In this paper, we present approaches for reliable and efficient feature extraction together with the features' incorporation into a robust state estimation process.**

I. Introduction

Self-localization in the RoboCup soccer domain might appear easy on first sight due to the clearly structured and color-coded environment. But in fact, the participating robots have quite limited sensorial and computational resources for this task. Their field of view is limited and the walking humanoid body structure provides additional noise to the sensor data. In addition, the scenario becomes more challenging over the years. Having started on small fields with additional beacons for localization and borders to prevent the robots perceiving false positives in the audience or on the floor, the field becomes larger – and thus significant features might be more far away –, the surrounding less specified, and unique features become reduced, e. g., in the Standard Platform League only colored goal posts remained.

The contribution of this paper is to provide a summary of approaches for processing the sensorial input in such a domain to obtain robust and precise world state estimates. This includes vision algorithms as well as a precise robot calibration and a robust probabilistic model. A short state of the art in the RoboCup domain is given in the respective sections.

All approaches described in this paper have been developed using the hardware and software of the team B-Human that has participated in the Humanoid Kid-Size League [1] and later on in the Standard Platform League [2]. All experiments have been conducted using the Nao robot platform [3].

This paper is structured as follows: Section II describes approaches that are necessary for obtaining precise perspective information based on proprioceptive data. In the following Sect. III, the currently used vision approach is described. Section IV describes the integration of the perceived features into a state estimation process. Finally, experiments and their results are shown in Sect. V.

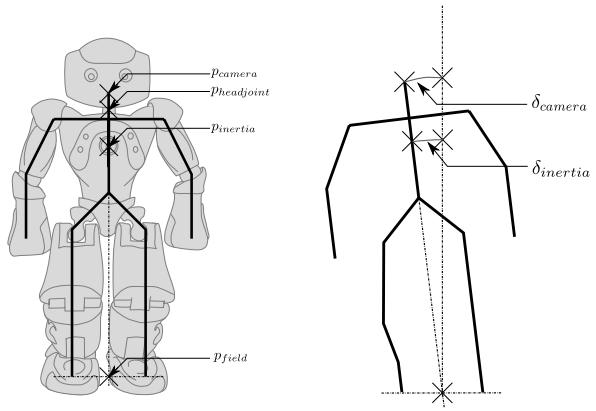

Fig. 1. Body structure of the Nao robot including the positions significant for perspective computations: The origin p_{field}, the inertia sensor $p_{inertia}$ and the p_{camera}. The camera's position offset δ_{camera} resulting from walking motions can be compensated by the measurable offset $\delta_{inertia}$.

II. Estimating the Perspective

For computing an object's position relative to the robot, it is necessary to determine the camera's parameters as precise as possible. In addition, imprecision resulting from the robot's walking motion or the head-mounted camera's pan and tilt movements needs to be compensated.

A. General Approach

A camera's parameters can be divided into two groups: intrinsic – including, e. g., focal length and lens distortion – and extrinsic – describing the coordinate transformation between the camera and a given point of reference – parameters. In general, the intrinsic parameters are provided by the manufacturer, or can be determined easily using a standard calibration approach, e. g. the one by [4].

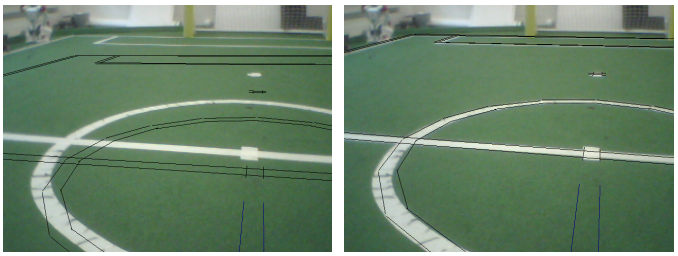

Fig. 2. Projection of the field geometry into the camera image before (left) and after (right) calibrating the additional camera parameters.

Most calibration approaches also allow determining extrinsic parameters, but since a soccer robot's camera is in constant motion, a different solution is necessary. A reasonable approach is to define an origin of the robot relative world coordinate system – in our case the center between the two feet, depicted as p_{field} in Fig. 1 – and to follow the robot's kinematic chain from this point to the camera.

B. Additional Camera Calibration

In addition to the previously described parameters, some additional robot-specific parameters are required two overcome two problems: the camera cannot be mounted perfectly plain – this is especially the cases for self-assembled robots but also for the commercially manufactured Nao – and the torso is not always standing perfectly vertical due to backlash which cannot be measured. In both cases, small variations can lead to significant errors when projecting more distant objects onto the field, as shown in Fig. 2.

These parameters, correcting the camera roll and tilt as well as the overall body roll and tilt, are currently determined in a manual calibration process.

C. Filtering and Integrating Inertia Data

During walking, a computation of the extrinsic camera parameters is still possible but subject to heavy noise. The sensor data provided by the motors does not reflect the swinging of the robot's body during certain walk phases. Therefore, the resulting camera offset δ_{camera} needs to be compensated (cf. Fig. 1). This is realized by estimating the torso's offset $\delta_{inertia}$ based on the measurements of the inertia sensor in the Nao's chest.

Estimating the pose of the torso consists of three different tasks. First, discontinuities in the inertial sensor readings are excluded. Second, the calibration offsets for the two gyroscopes are maintained. Third, the actual torso pose is estimated using an Unscented Kalman filter (UKF) [5].

Excluding discontinuities in the sensor readings is necessary, because some sensor measurements provided by the Nao cannot be explained by the usual sensor noise. This malfunction occurs sporadically and affects most measurements from the inertia board within a single frame (cf. Fig. 3). The corrupted frames are detected by comparing the difference of each value and its predecessor to a predefined threshold.

Gyroscopes have a bias drift, i. e., the output when the angular velocity is zero drifts over time due to factors such as

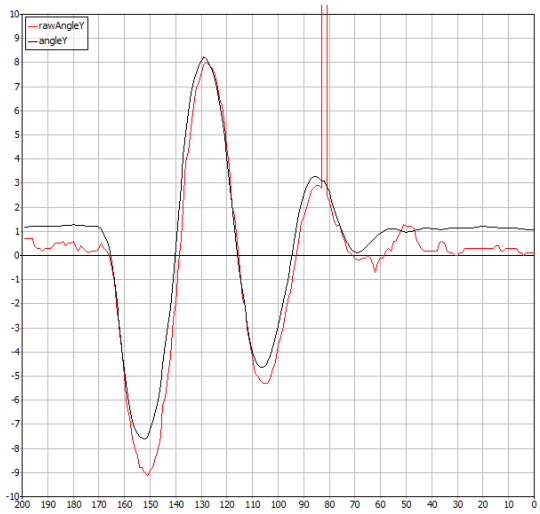

Fig. 3. The difference between the estimated pitch angle *angleY* and the pitch angle *rawAngleY* provided by the inertia board of the Nao. Between the frames 85 and 80, the Nao provided corrupted sensor values.

temperature that cannot be observed. The temperature changes slowly as long as the robot runs, so that it is necessary to redetermine the bias continuously. Therefore, it is hypothesized that the torso of the robot and thereby the inertial measurement unit has the same pose at the beginning and the end of a walking phase (i. e. two steps). Therefore, the average gyro measurement over a whole walking should be zero. This should also apply if the robot is standing. So either, the average measurements over a whole walking phase are determined, or the average over 1 sec for a standing robot. These averages are filtered through one-dimensional Kalman filters and used as biases of the gyroscopes. The collection of gyroscope measurements is limited to situations in which the robot is either standing or walking slowly and has contact to the ground (determined through the force sensitive resistors in the Nao's feet).

The UKF estimates the pose of the robot torso (cf. Fig. 3) that is represented as three-dimensional rotation matrix. The change of the rotation of the feet relative to the torso in each frame is used as process update. The sensor update is derived from the calibrated gyroscope values. Another sensor update is added from a simple absolute measurement realized under the assumption that the longer leg of the robot rests evenly on the ground as long as the robot stands almost upright. In cases in which this assumption is apparently incorrect, the acceleration sensor is used instead.

It is not only possible to get the orientation from the UKF, but also to get a "filtered" version of the gyroscope measurements from the change in orientation, including a calculated z-gyroscope value that is actually missing on the Nao.

D. Compensating Image Distortion

The Nao robot – as well as most humanoid soccer robots – is equipped with a simple CMOS camera. Such cameras have

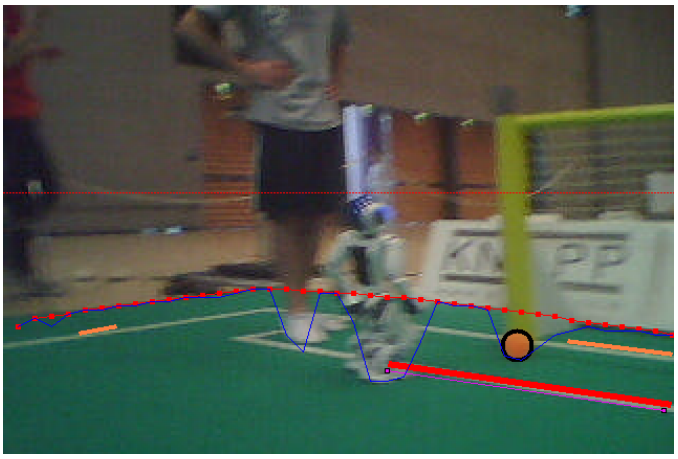

Fig. 4. The field border: the thin dashed red line depicts the robot's horizon, the blue line connects the green points found, and the red dotted line is the upper part of the convex hull around the field.

a central weakness, the so-called rolling shutter. Instead of taking images at a certain point in time, a rolling shutter takes an image pixel by pixel, row by row. Thus the last pixel of an image is taken significantly later than the first one. By moving its head, the Nao can point the camera in different directions. Since an image is not taken all at once, the camera may point to a different direction when the first pixel is recorded than when the last pixel is taken. This results in a distorted image and thus in inaccurate perceptions.

To overcome this problem, a mechanism – originally developed for the AIBO robot – based on the image recording time and the speed of the head joints is applied. For a detailed description see [6].

III. Feature Extraction

The most time-consuming and often also most error-prone software component of a soccer robot is its vision system. By integrating knowledge about the spatial context, our approach provides robust results without being too computationally expensive.

A. General Approach

For computer vision in this domain, two main approaches are popular: blob-based and grid-based systems. For extracting blobs from an image, full color segmentation is necessary. After the segmentation, connected regions of the same color class become determined. A common solution for this task is *CMVision* by [7]. This approach provides robust results but is quite time-consuming (on robots such as the Nao) since every pixel of the image needs to be examined. Grid-based approaches can be significantly faster, since only a fraction of all pixels becomes interpreted namely those on a (often horizon-aligned) grid. This technique was introduced in the Standard Platform League by [8]. Since the grid lines are only one-dimensional, only a small context of a pixel can be considered and thus makes this approach more sensitive to outliers. To overcome this problem, often so-called *specialists* need to be applied, examining a feature's region in more detail.

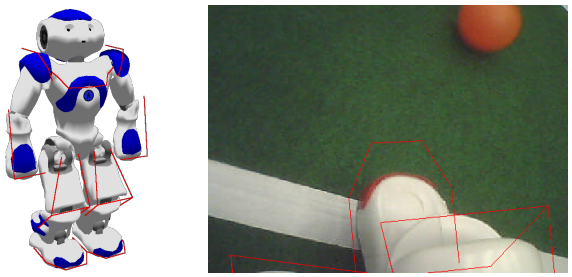

Fig. 5. Body contour in 3-D (left) and projected to the camera image after kicking the ball (right).

Our approach is a combination of both techniques: In a first step, significant segments are searched on a grid. This grid is bounded by the robot's spatial context, i. e. the border of the field and its own body contour (cf. Sect. III-B). In a second step, the segments become merged to regions according to a set of constraints. These regions are the base for the final feature detection.

B. Context of Field and Body

Since the robot soccer environment is flat and all features that need to be detected (ball, field lines, and goal post) have their base on the ground, the segmentation only needs to be done for the part of the image that is below the visible field border. The field border itself is also on the ground and therefore must be below the horizon in the image that serves as a starting point for detecting the field border. This is done by running scan lines, starting from the horizon, downwards until a green segment of a minimum length is found. From these points the upper half of the convex hull is used as field border. Figure 4 depicts the necessary elements for this computation.

If the robot sees parts of its own body, it might confuse white areas with field lines and – under certain conditions – red parts with the ball. However, by using forward kinematics, the robot can actually know where its body is visible in the camera image and exclude these areas from image processing. This is achieved by modeling the boundaries of body parts that are potentially visible in 3-D (cf. Fig. 5 left) and projecting them back to the camera image (cf. Fig. 5 right). The part of that projection that intersects with the camera image or is above it is used by the image processor as lower clipping boundary. The projection relies on the image coordinate system (cf. Sect. II-D), i. e., the linear interpolation of the joint angles to match the time when the image was taken. However if joints accelerate or decelerate, the projection may not be accurate, as can be seen in Fig. 5 right), where the foot just stopped after a kick. In that case, the clipping region might be too big or too small.

C. Segmentation and Region-Building

After all boundaries have been computed, scan lines within the remaining valid area can be used to create segments. For the goal detection two special scans are done to detect vertical and horizontal yellow or blue segments above the horizon.

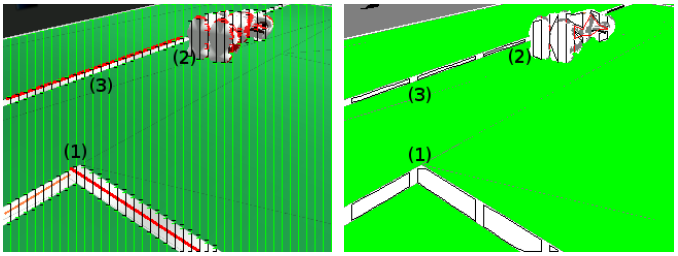

Fig. 6. Segmentation and region-building: The left image depicts the segments found on scan lines and the finally computed lines (accepted red lines and one orange line segment that is too short for later consideration). The right image shows the regions that result from merging single segments. Some failed constraints are denoted with numbers: (1) not connected because of change in direction, (2) not connected because of the length ratio, (3) maximum size reached

The horizontal scan lines are continued below the horizon until there is a scan line in which no yellow or blue segment is found. Based on these segments, regions are created by uniting touching segments of the same color to a region. White segments (being candidates for field lines) touching each other need to fulfill certain constraints to become united to a region:

- There is a maximum region size
- The length ratio of the two touching segments may not exceed a certain maximum
- The change in direction may not exceed a certain maximum (the vector connecting the middle of the segments connected to the middle of the next segment is treated as direction)
- If two segments are touching each other and both already are connected to a region, they are not united

These restrictions are needed because we do not want to have a single big region containing all field lines and robots. The result of these restrictions is that we most likely get small straight white regions (cf. Fig. 6).

This process is not applied to potential goal segments. Their merging is described in cf. Sect. III-E.

The following region classification is done by iterating over all regions and filtering all white regions to determine whether the region could be a part of a line. Thus, a white region needs to fulfill the following constraints:

- It has to have a certain minimum size.
- The axis of orientation must be determinable (since this is the base information passed to further modules).
- The amount of neighboring uncolored regions must be very small (since robot parts are classified as uncolored).
- It has to have a certain amount of green around.

D. Detecting Lines

For each white region that was classified as line region the start and end point of the axis of orientation is transformed to field coordinates. These two points form a line segment. The lines are built by clustering the associated segments, as shown in algorithm 1. The basic idea of the algorithm is similar to the quality threshold clustering algorithm introduced by [9], but it ensures that it runs in the worst-case-scenario in $\mathcal{O}(n^2)$ runtime. Therefore, it is not guaranteed to find optimal clusters. Since the number of line segments is limited by the field setup, practical usage showed that the algorithm has an acceptable runtime and delivers satisfiable results.

Algorithm 1 Clustering LineSegments

```
while lineSegments ≠ ∅ do
  s ← lineSegments.pop()
  supporters ← ∅
  for all s' ∈ lineSegments do
    if similarity(s, s') < similarityThreshold then
      supporters.add(s')
    end if
  end for
  if supporters.size() > supporterThreshold then
    createLine({s} ∪ supporters)
    lineSegments ← lineSegments\supporters
  end if
end while
```

All remaining line segments are taken into account for the circle detection. For each pair of neighboring segments the intersection of the perpendicular from the middle of the segments is calculated. If the distance of this intersection is close to the real circle radius, for each segment a spot is generated which has the distance of the radius to the segment. Now the same clustering algorithm used for the lines is used to find a cluster for the circle. For all line segments that were neither clustered to a line nor to the circle and have a certain minimum size, additional lines are created. This is necessary because a vertical line might create one big single region. For all lines the intersections are calculated and classified as L, T, or X intersection.

E. Detecting Goals

To detect goal posts, a Hough Transformation on the horizontal goal segments is performed to search for long vertical blue or yellow lines. The lines found are then checked whether they match some characteristic properties of a goal post, to make sure not to accept blue robot parts or things outside the field as goal posts. The most important checks are:

- Is there some green below the post?
- Is the bottom of the percept under the calculated horizon and the top above it?
- Does the expected width of the goal post match the width of the percept?
- Is the calculated distance of the goal post within a plausible range?
- Do we see goal posts of only one color?

Percepts that do not fulfill these criteria are completely discarded since for the state estimation process avoiding false positives is more important than integrating new information. For the remaining percepts, a determination of the side is performed. If there are exactly two remaining percepts of the same color, this can be done easily. Otherwise, the surrounding of the head point of the only remaining percept is scanned in order to try to detect parts of the crossbar and to determine the side of the post.

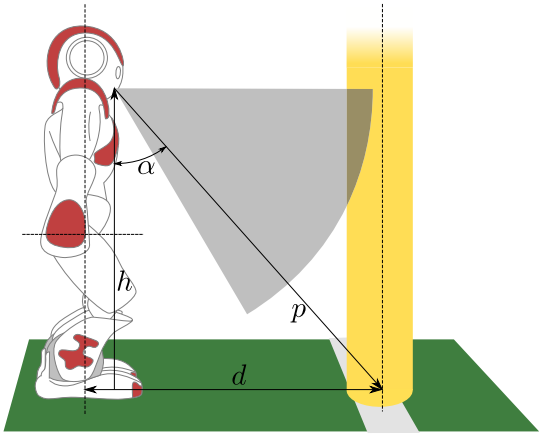

Fig. 7. Two-dimensional example of a robot observing the base point of a goal post. The distance d can be computed given the intrinsic (contributing to α) and extrinsic (providing h, contributing to α) camera parameters.

IV. Self-Localization

Subsequent to the feature extraction, the self-localization integrates the perceptions into the robot pose estimate. For modeling the uncertainties in the probabilistic approach applied, especially the remaining errors of the perspective estimation and the shortcomings of the vision system need to be taken into account. In addition, the constraints resulting from the computational limits demand a careful parameterization.

A. General Approach

For self-localization, we use a particle filter based on the Monte Carlo method [10] as it is a proven approach to provide accurate results in such an environment [11]. Additionally, it is able to deal with the kidnapped robot problem that often occurs in RoboCup scenarios. For a faster reestablishment of a reasonable position estimate after a kidnapping, the *Augmented MCL* approach by [12] has been implemented. A comprehensive description of our general state estimation implementation – applied to a Humanoid League scenario – is given in [13], in the meantime extended by a new clustering approach for pose extraction [14].

An alternative standard approach for robot self-localization would be the application of a Kalman filter, but the major drawback of this kind of technique is its missing ability of recovering from a kidnapping situation. However, recent results from [15] indicate the usage of Multiple Model Kalman filters as a reasonable alternative to a particle filter. The techniques described in the following sections might also be applicable with reservations to that approach.

In the following, the state X_t to be estimated and thus the content of every particle $x_t^{[m]}$ within a set of M particles ($M = 100$) is a simple pose in 2-D:

$$x_t^{[m]} := \langle px_t^{[m]}, py_t^{[m]}, \theta_t^{[m]} \rangle \tag{1}$$

A sample's weighting is denoted by $w_t^{[m]}$.

B. Modeling Uncertainty

When performing the Monte Carlo sensor update step, each sample's weighting $w_t^{[m]}$ becomes computed based on the deviation of the model $x_t^{[m]}$ from the current perceptions. Let $p_1, \ldots, p_n$ be the perceptions made within one execution cycle and $w(x, p)$ a function computing the likelihood of an observation given a sample's state, a weighting can be computed as follows:

$$w_t^{[m]} = w(x_t^{[m]}, p_1) w(x_t^{[m]}, p_2) \ldots w(x_t^{[m]}, p_n) \tag{2}$$

To determine $w(x_t^{[m]}, p)$ for a perception p, two deviations from the expected model need to be taken into account: the deviation of the direction on the ground as well as the difference of the distances. The crucial aspect of the likelihood computation is its coordinate system. A vision system might provide the perception as a vectorial offset or in polar coordinates, but in both systems the distance uncertainty resulting from the humanoid robot's body shaking can only be modeled insufficiently. As depicted in Fig. 7, the distance depends on the angle α relative to the robot's camera position. During walking, α is subject to noise and small deviations can obviously cause huge distance errors for far perceptions. Since close perceptions are not affected by this problem and similar problems occur for directional deviations in the ground plane, it is a reasonable choice to compute $w(x_t^{[m]}, p)$ in angular coordinates.

Given the angle α_p determining the distance to a perception (as depicted in Fig. 7), the angle β_p determining the directional deviation on the ground, the accordant model angles $\alpha_{x_t^{[m]}}$ and $\beta_{x_t^{[m]}}$, and the percept type-related standard deviations σ_{α_p} and σ_{β_p}, $w_t^{[m]}$ can be computed as follows:

$$\delta_\alpha^{[m]} = |\alpha_p - \alpha_{x_t^{[m]}}| \tag{3}$$

$$\delta_\beta^{[m]} = |\beta_p - \beta_{x_t^{[m]}}| \tag{4}$$

$$w_t^{[m]} = \mathcal{N}(\delta_\alpha^{[m]}, \sigma_{\alpha_p}^2) \mathcal{N}(dist_t^{[m]}, \sigma_{\beta_p}^2) \tag{5}$$

To save computation time, the current implementation does not use all perceptions of an execution cycle but selects n random perceptions to become integrated (currently, $n = 6$); goalposts are preferred since they are the only unique elements providing a global direction.

C. Preventing Particle Depletion

Since the computational resources are limited, a particle filter can only be operating on a small particle set to remain efficient. Our implementation is currently configured to use only 100 samples. But even when using e. g. 500 samples, only a sparse coverage of the state space can be realized. Thus, the problem of particle depletion might occur, i. e. during the MCL resampling step the probability distribution reduces to a small set of particles. Especially in this scenario, having robots with a strongly limited field of view, it might happen that certain local observations – not necessarily being exceedingly compatible to the global hypothesis – fit perfectly a small

subset of samples and thereby rule out the majority of other samples approximating the current probability distribution.

To avoid such fluctuations without the need of keeping weightings over multiple cycles, two mechanisms have been incorporated: the configuration of high standard deviations as well as the usage of base probabilities. Both lead to a more balanced distribution of weightings and thus avoid situations in which single samples accidentally (e. g. through perfectly fitting a single observation) eliminate significant parts of the sample set.

These approaches also stabilize the used Augmented MCL approach that relies on the change of the sample set's total weighting over time.

D. Using Context to Exclude False Positives

For a precise localization near a goal, it is not only necessary to perceive the goal posts – which are rarely seen to a utilizable extent in such a situation – but also to avoid confusing the goal net with field lines. The exclusion of this kind of false positives turned out to be almost impossible using the current image processing approach: When looking into a goal in a certain angle, the projection of the net to the ground might appear as a perfect line at a certain distance.

Being unsolvable for the vision system, this task has been moved one level up to the self-localization since it sets perceptions in context with its field model. Whenever a sample's weighting $w_t^{[m]}$ is updated by line information, it is possible to check whether it might be a goal net segment given the context of $x_t^{[m]}$. In that case, $w_t^{[m]}$ will not be updated. To avoid any inconsistencies within the sample set, all omitted samples become updated afterwards using the average weighting update of all other samples.

To realize an efficient computation, the self-localization component has access to a precomputed look-up table that provides the maximum valid distance to a field line for a given $x_t^{[m]}$. As all false positives (given a robot position inside the field) resulting from the goal net lie beyond the goal line, this is an effective way of excluding them from the sensor update.

Figure 8 shows an extract of the table used.

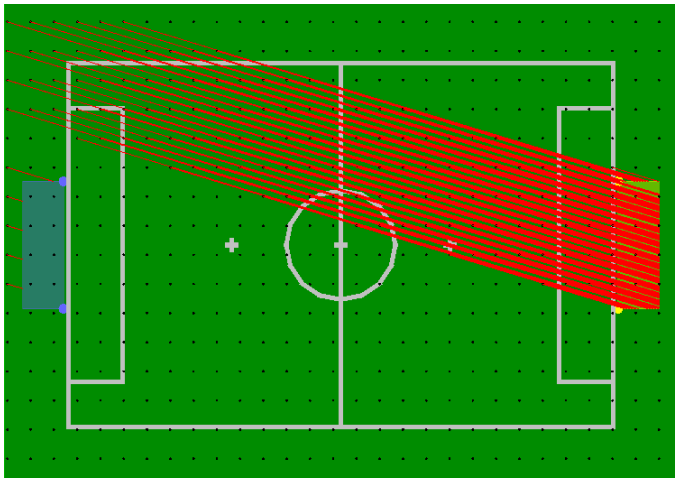

Fig. 8. Depiction of one layer of the goal net look-up table at a low resolution. After rasterization of the environment, the maximum distance d_g to a goal is computed for a given angle (here: about -15°) and drawn as a red line.

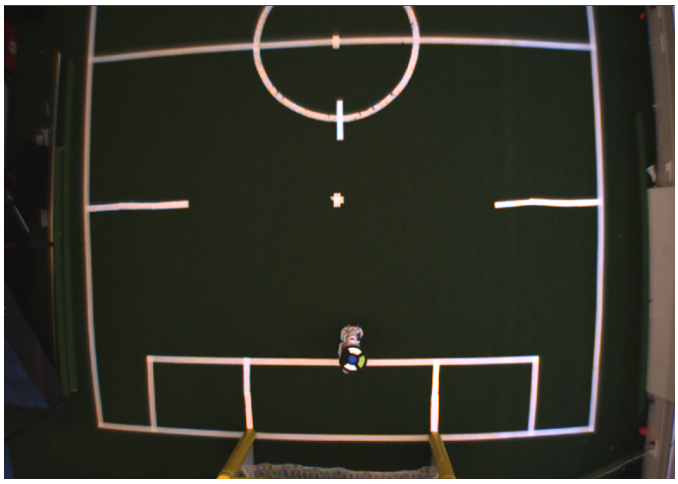

Fig. 9. Image taken by one of the cameras providing ground truth data. In the lower center, a tracking pattern on a robot's head can be seen. The field has intentionally a wrong line layout (cf. Tab. I, experiment 6).

V. Experimental Results

To evaluate the overall performance of the approaches described, several experiments have been carried out. In addition, the system has also been used successfully in different RoboCup competitions.

A. Quantitative Evaluation

To determine the system's metrical precision, the self-localization's output has been compared to the ground truth about the robot's position within different experimental setups.

As source for ground truth data, a global tracking system has been used. For this purpose, a unique marker has been fixed on the robot's head (cf. Fig. 9) and been tracked by a camera hanging above the field; the software for this purpose is the standard vision system of the RoboCup Small Size League [16]. The system provides the position as well as the rotation (which is fused with the robot's head rotation) of the robot on the field.

An overview of all experiments and the self-localization precision achieved is given in Tab. I. In all experiments in which the robot was walking around alone and scanning the environment for features (No. 1 – 4), the average localization error was significantly below 150 mm. Surprisingly, even when walking on a changed field (which was not modeled in software) – having a displaced goal (No. 5) or additional field lines (No. 6, shown in Fig. 9)– the error remained significantly below 200 mm.

The remaining experiments (No. 7 – 11) measured the precision during normal soccer play. They included the presence of a second, adversarial robot, periods of time focusing on the ball instead of features for self-localization, as well as several tumbles. Due to these difficulties, the average error increased but still stayed below 250 mm. The results of experiment 11

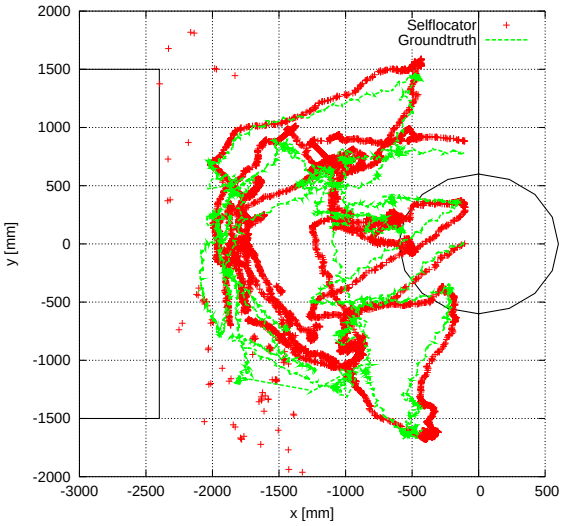

Fig. 10. Walking trajectory in experiment 11. The red crosses denote the estimate computed by the self-localization. The dashed green line depicts the ground truth.

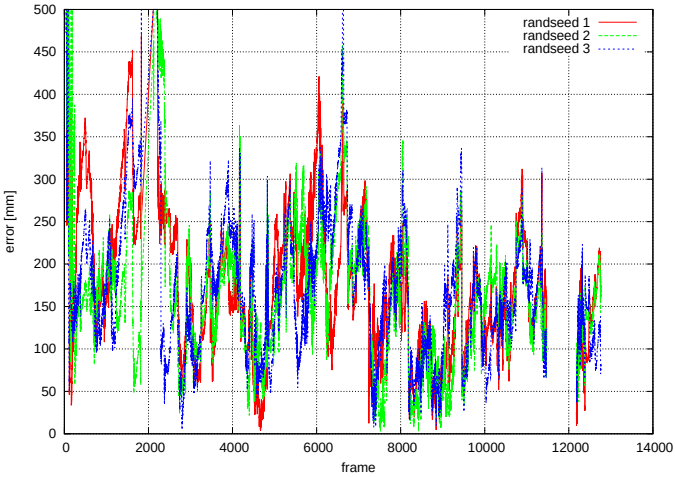

Fig. 11. Localization error during experiment 11. The perceived data has been processed offline multiple times using a different random seed to evaluate its effect on the result. The robot leaves the visible area of the ceiling camera around frame number 11500.

are depicted in detail: Figure 10 shows the difference between estimated and the trajectory really walked, the error over time is shown in Fig. 11, and the distribution of the mean error is presented in Fig. 12.

The overall computing time of the system's cognitive part (i. e. image processing, state estimation, and action selection) is always significantly below 33 ms to keep up with the camera's frame rate of 30 Hz. For self-localization, the computing time is about 6 ms on average.

B. Qualitative Evaluation

The approaches described in this paper have not only been tested under laboratory conditions but also in real RoboCup competitions. For this participation, no ground truth data that allows computing an average error exists. Thus, the overall

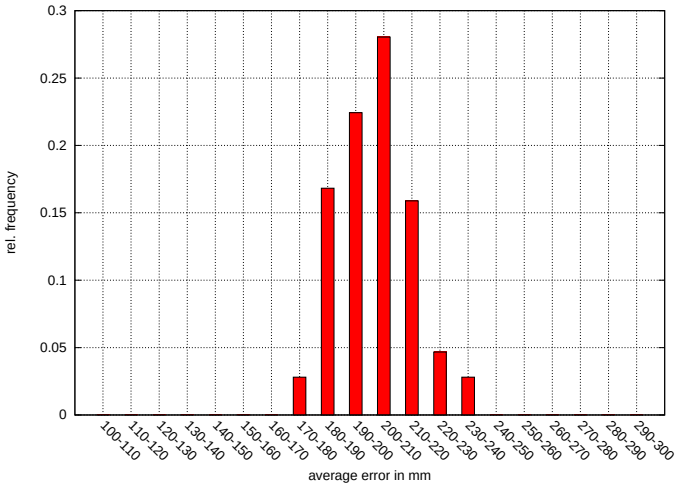

Fig. 12. Histogram showing the distribution of the mean localization error in experiment 11.

team performance has to be considered as an indicator.

Our team has successfully participated in the RoboCup German Open 2009 as well as in RoboCup 2009. During the soccer competitions, the precise and robust self-localization provided a huge benefit, not only for kicking in the right direction but also for approaching the ball in a configuration allowing an immediate shoot, allowing us to score a total of 91 goals in 13 matches. Additionally, the localization enabled an automatic robot placement before kickoff – which provides a clear advantage – as well as a proper goalie position adjustment.

We also participated in the *Localization with Obstacle Avoidance Challenge* that required a robot to walk as close as possible to three specified (but previously unknown) positions within a given amount of time without colliding with any obstacle. The setup of this challenge is shown in Fig. 13. We have been the only team accomplishing this challenge.

The software used by B-Human at RoboCup 2009 can be downloaded from the team website (`http://www.b-human.de`), including the implementation of the algorithms described in this paper.

VI. Conclusion and Current Work

In this paper, we presented different approaches for processing the sensorial input in a humanoid soccer robot scenario to obtain robust and precise world state estimates. This included a more precise estimation of the robot's perspective, the vision system, as well as the integration into the state estimation process. In several different experiments as well as during real competitions, the overall precision of the system is shown.

One ongoing project is the optimization of a number of self-localization parameters, e. g. different standard deviations, the number of samples, or parameters for motion noise, by applying a *Particle Swarm Optimizer* [17].

No.	Description of experiment	average error	StdDev.
1	Walking eight figure; head moved by active vision	$119.9mm$	19.8%
2	Walking eight figure; fixed head motion pattern	$116.7mm$	6.2%
3	Walking eight figure; head mostly focused on goals	$146.9mm$	7.3%
4	Walking circles; fixed head motion pattern	$116.5mm$	9.4%
5	Walking circles; fixed head motion pattern; one goal displaced by about 0.5m	$187.2mm$	23.2%
6	Walking circles; fixed head motion pattern; field contains additional fake lines	$125.2mm$	18.7%
7	Robot *Penny* plays soccer	$231.0mm$	8.8%
8	Robot *Penny* plays soccer	$204.2mm$	5.5%
9	Robot *Leonard* plays soccer	$220.8mm$	5.7%
10	Robot *Leonard* plays soccer	$204.9mm$	5.2%
11	Robot *Penny* plays soccer	$201.5mm$	6.7%

TABLE I

PRECISION OF SELF-LOCALIZATION IN DIFFERENT SCENARIOS.

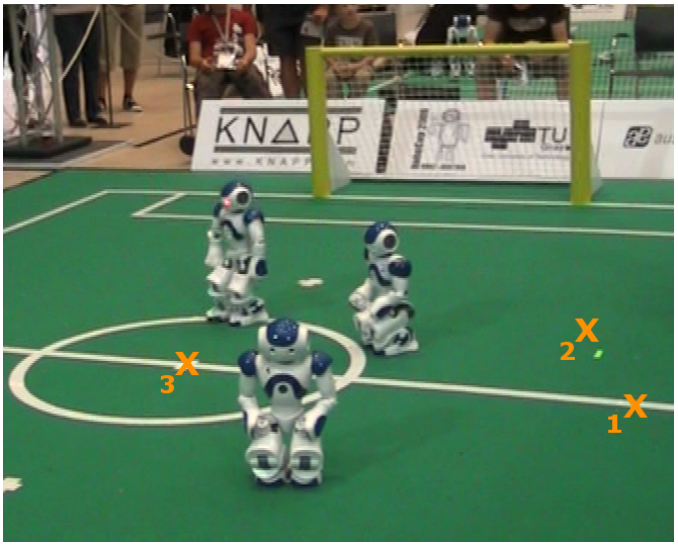

Fig. 13. B-Human accomplishing the *Localization with Obstacle Avoidance Challenge* at RoboCup 2009, the numbered crosses denote the target positions in visiting order.

ACKNOWLEDGEMENTS

The authors would like to thank all B-Human team members for providing the software base for this work.

REFERENCES

[1] T. Röfer, T. Laue, A. Burchardt, E. Damrose, M. Fritsche, J. Müller, and A. Rieskamp, "B-Human team description for robocup 2008," in *RoboCup 2008: Robot Soccer World Cup XII Preproceedings*, L. Iocchi, H. Matsubara, A. Weitzenfeld, and C. Zhou, Eds. RoboCup Federation, 2008.

[2] T. Röfer, T. Laue, O. Bösche, I. Sieverdingbeck, T. Wiedemeyer, and J.-H. Worch, "B-Human team description for robocup 2009," in *RoboCup 2009: Robot Soccer World Cup XII Preproceedings*, J. Baltes, M. Lagoudakis, T. Naruse, and S. Shiry, Eds. RoboCup Federation, 2009.

[3] D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier, "The NAO humanoid: a combination of performance and affordability," *CoRR*, vol. abs/0807.3223, 2008.

[4] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[5] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference, 1995. Proceedings of the*, vol. 3, 1995, pp. 1628–1632. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=529783

[6] T. Röfer, "Region-based segmentation with ambiguous color classes and 2-D motion compensation," in *RoboCup 2007: Robot Soccer World Cup XI*, ser. Lecture Notes in Artificial Intelligence, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Springer, 2008.

[7] J. Bruce, T. Balch, and M. Veloso, "Fast and inexpensive color image segmentation for interactive robots," in *Proceedings of IROS-2000*, Japan, October 2000.

[8] J. Bach and M. Jüngel, "Using pattern matching on a flexible horizon-aligned grid for robotic vision," *Concurrency Specification and Programming - CSP'2002*, vol. 1, pp. 11–19, 2002.

[9] L. Heyer, S. Kruglyak, and S. Yooseph, "Exploring expression data: identification and analysis of coexpressed genes," *Genome research*, vol. 9, no. 11, pp. 1106–1115, 1999.

[10] 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.

[11] T. Röfer, T. Laue, and D. Thomas, "Particle-filter-based self-localization using landmarks and directed lines," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence, A. Bredenfeld, A. Jacoff, I. Noda, and Y. Takahashi, Eds., no. 4020. Springer, 2006, pp. 608–615.

[12] J.-S. Gutmann and D. Fox, "An experimental comparison of localization methods continued," *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002. [Online]. Available: http://www.informatik.uni-freiburg.de/~gutmann/

[13] T. Laue and T. Röfer, "Particle filter-based state estimation in a competitive and uncertain environment," in *Proceedings of the 6th International Workshop on Embedded Systems*. VAMK, University of Applied Sciences; Vaasa, Finland, 2007.

[14] T. Laue and T. Röfer, "Pose extraction from sample sets in robot self-localization - a comparison and a novel approach," in *Proceedings of the 4th European Conference on Mobile Robots - ECMR'09*, I. Petrović and A. J. Lilienthal, Eds., Mlini/Dubrovnik, Croatia, 2009, pp. 283–288.

[15] M. J. Quinlan and R. H. Middleton, "Multiple model kalman filters: A localization technique for robocup soccer," in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Artificial Intelligence, J. Baltes, M. G. Lagoudakis, T. Naruse, and S. Shiry, Eds. Springer, to appear in 2010.

[16] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, "SSL-vision: The shared vision system for the RoboCup Small Size League," in *RoboCup 2009: Robot Soccer World Cup XIII*, ser. Lecture Notes in Artificial Intelligence, J. Baltes, M. G. Lagoudakis, T. Naruse, and S. Shiry, Eds. Springer, to appear in 2010.

[17] R. C. Eberhart and J. Kennedy, "A new optimizer using particles swarm theory," in *Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.