GermanTeam 2005

The German National RoboCup Team

Thomas Röfer[1], Ronnie Brunn[2], Stefan Czarnetzki[3], Marc Dassler[2], Matthias Hebbel[3], Matthias Jüngel[4], Thorsten Kerkhof[3], Walter Nistico[3], Tobias Oberlies[2], Carsten Rohde[3], Michael Spranger[4], and Christine Zarges[3]

[1] Bremer Institut für Sichere Systeme, Technologie-Zentrum Informatik, FB 3, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
[2] Fachgebiet Simulation und Systemoptimierung, Fachbereich Informatik, Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany
[3] Institute for Robot Research, Universität Dortmund, Otto-Hahn-Straße 8, 44221 Dortmund, Germany
[4] Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin, Rudower Chaussee 25, 12489 Berlin, Germany

http://www.germanteam.org
germanteam@informatik.hu-berlin.de

1 Introduction

The GermanTeam participates as a national team in the Sony Legged Robot League. It consists of students and researchers from the following four universities: the Humboldt-Universität zu Berlin, the Universität Bremen, the Technische Universität Darmstadt, and the Universität Dortmund. The members of the GermanTeam participate as individual teams in contests such as the RoboCup German Open, but jointly line up as a national team for the international RoboCup World Cup. This paper gives an overview on the work done by the four sub-teams of the GermanTeam (Aibo Team Humboldt, Darmstadt Dribbling Dackels, Bremen Byters, and Microsoft Hellhounds) in the past year that is currently combined to form the code of the GermanTeam 2005. Further information can be found in this year's contributions of members of the GermanTeam to the RoboCup book: how to improve the vision system [1], how to model the Aibo's gait [2], about the self-locator used in 2004 [3], how to improve self-localization using negative information [4], and about the general physical robotics simulator SimRobot 2005 [5].

2 Perception

The new rules for RoboCup 2005 introduced changes in the dimensions and structure of the field, including a rearrangement of the beacons and the removal of both the field wall and the outer white wall. This has introduced several new challenges: the existing self-localization often gets lost if the robot walks in areas in which neither beacons nor goals are visible, such as the corners of the field.

Also, misperceptions have become more likely, since there are no more walls around the field shielding the view of the robots onto the audience.

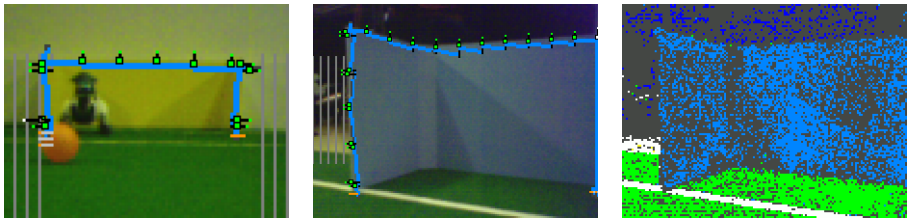
Furthermore, a new issue with the ERS-7 platform has been discovered: different robots seem to have a different perception of the colors under strong lighting conditions, possibly due to variations in the IR filters of the cameras. Currently this makes individually calibrated color tables necessary, but we plan to extend our camera calibration approach (cf. [1]) to compensate for these differences.

2.1 Goal Recognizer

Motivation. The goals are important beacons for self-localization because they are within the robots' view very often. Therefore it is important to distinguish them from goal-colored areas, which might be seen in the audience. Our approach to this problem is to use more knowledge about the goal: there is a strong edge between the goal and the background, the goalposts, i. e. the vertical edges of the goal, are straight, they are connected by the cross bar, usually without interruption, there is green carpet below the goal, and the goal has a minimum height, absolute and with respect to the goal's width.

Scanning strategy. To examine these properties, a new analysis procedure has to be implemented. It is triggered when several goal-colored pixels were discovered on the common image processing grid. From the scanning direction on that grid, it is known that the starting point will be near the left goalpost. We make the focus slide along the edge by alternate scans on horizontal and vertical lines (cf. Fig. 1). This yields the top and bottom points of the goalpost and a set of edge points. We fit a line to those points and calculate the residuals, which are a (inverse) measure of the straightness of the edge. Furthermore we look for green pixels below the bottom point.

In the second step, the procedure scans along the cross bar towards the other goalpost. We chose to follow the top edge because it is the least likely to be obstructed by the goalie or shadows within the goal. Having reached the second goalpost, it is analyzed like the first one.



(a) (b) (c)

Fig. 1. (a)-(b) Scanning strategy to analyze the goal; (c) Color table classification of image b, containing many unclassified pixels (gray)

Fig. 2. The goal recognizer realizes that the left goalpost is occluded.

Extended color classification and edge detection. The above strategy requires that scans on single lines can detect edges while respecting the connectivity of the goal-colored area. This is not possible only using the color table classification because it is optimized to not misclassify any pixels (cf. Fig. 1(c)). To fill the gaps, we make use of the Mahalanobis distance of the pixel's color c to an adaptive reference color c_{ref}. It is calculated with the formula

$$d = (c - c_{ref})C^{-1}(c - c_{ref})$$

where C is the covariance of the goal color within one reference picture (a matrix to be determined through calibration). The reference color c_{ref} is an average over goal-colored pixels that were encountered on the current scan line. Thus the distance measure reflects the deviation to the local goal color shade taking normal variation into account.

Two thresholds are applied to the distance values in order to distinguish between pixels with "small", "medium", and "large" color deviations. Pixels in the latter two classes (and of course pixels that were classified as a different color class) are an indication that the scan line has exceeded the goal-colored area. In order to differentiate between strong and weak edges, we compare the number of pixels with medium and large deviations within a certain range behind the edge. That way we yield information about the edge quality without calculating the gradient explicitly.

Interpretation. With the criteria described above, it is very well possible to disambiguate between the goals and other goal-colored areas. Furthermore, with the straightness and edge quality condition, the robot can decide for each goalpost whether it has been seen or not. This allows using the goal width that is very valuable information for self-localization, if and only if both goalposts were detected.

2.2 Ball Recognition

To allow for filtering of false ball percepts, the existing algorithm for ball detection has been extended to calculate the reliability for each percept, which is then taken into account in the ball model. This is calculated based on heuristics for the shape, color, position and the neighborhood of the ball.

Unlike the previous solution, the detection process is not only triggered at one candidate point per image, but on all clusters of orange found during the scanning process. Therefore, large orange objects in the background do not prevent a ball in medium distance from being recognized. This way, the ball can be tracked even in presence of similarly colored objects present in the audience.

2.3 Line and Center Circle Detection

As in the years before, the vision system scans the image on a horizon-aligned grid. During this scanning process it is recognized whether a field line is touched by a scan line. Until now, a few randomly chosen line points were forwarded to the self-localization without further processing, i. e. without using them to build line hypotheses. In our new approach, these points are clustered into line fragments. The line fragments are fused into lines which are used to calculate line crossings. All this is done using filtering heuristics to reduce the number of false positives. To lessen the sensor ambiguity, the crossings are characterized (if necessary with additional scanning) as L-shaped or T-shaped crossings or "virtual" crossings if they lie outside of the image. This information, together with the orientation of each crossing, is finally forwarded to the self localization, which can use them as additional landmarks.

As a part of this algorithm, the line fragments are projected on the field and pair wise considered as possible tangents of a circle of a known radius. If enough tangents of a possible center circle are found, the candidate's center is verified with the middle line crossing it and with additional scans. This way, the center circle can be detected from a distance of up to 1.5 m with high accuracy, and considering the orientation provided by the center line, it leaves only two possible symmetrical robot locations on the field.

3 Object Modeling

3.1 Self Locator

The GT2004 self-locator [3] was restructured into the following components:

- the sample, representing one particle of the particle filter
- the template generation, computing template samples for sensor resetting
- a motion model
- an observation model
- the particle filter based self-locator

To improve the localization, two new types of percepts are used: the line crossing and the center circle. As the center circle is a unique item on the field its advantage is obvious. Using both its position and orientation, the robots position is determined to be in one of two areas, symmetric to the center circle.

Line crossings can be found at 14 positions on the field. They are either of an L- or a T-form. Crossings provide two advantages over just using points

on lines as done before. One is that they are less ambiguous than lines. Figure 3 is showing this: the robot is at the position in the upper left corner, facing the L-crossing. The equally probable positions in the goal area resulting from just using the lines are denoted by the red triangles. This number is halved when using the crossing characterization (yellow circles), reducing the risk of delocalization. The other advantage is that more information can be used in a single frame. A crossing already provides the information that there are (at least) two lines, while it cannot be guaranteed that points from all lines are used by randomly selecting few points from the lines visible.

Using an increased number of different percepts results in a much larger number of parameters (up to 3 per type: distance, angle, and orientation) which complicates adjustments. Therefore the parameters are split up in a measurable part (the variances of the information) and a weight (which accounts for observation frequency, probability of false positives and negatives, etc.) so that for each kind of percept there is only one parameter to be adjusted. Using a ceiling mounted camera the variances can be easily measured, but as the variances change with the motion parameters of the robot and its position, acceptable variances need to be estimated from the measurements. To avoid this estimation, the modeling of the percept variances needs to be extended to represent a function calculating the variances from the parameters mentioned above. Once the variances are determined, the weights for the different percepts can be adapted by an evolutionary algorithm.

3.2 Ball Locator

For the filtering of ball percepts a new approach based on particles was realized. This implementation is adapted to the particular needs of a ball locator: the state evolves very quickly, the sensor aliasing is nearly zero in most situations, and the sensor noise can be very small, especially when the ball is near to the robot. To achieve a good reactivity of the filter and avoid the known problem of the *filter degeneracy*, our approach strongly relies upon the *sensor resetting* ([6]) and *dual Monte-Carlo* ([7]) ideas.

The advantage of a particle filter over, e. g., a Kalman filter is the possibility to deal with ambiguous situations by tracking different candidate ball positions and consider more than a single ball percept in a frame (cf. sec. 2.2). Therefore

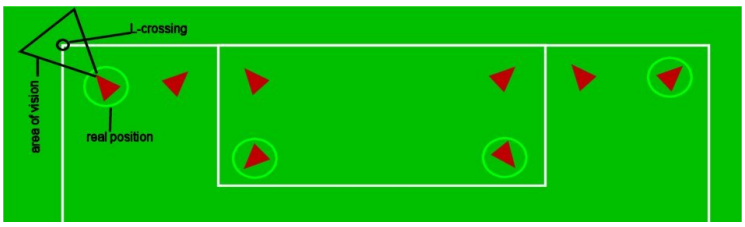


Fig. 3. Possible poses derived from lines and crossings

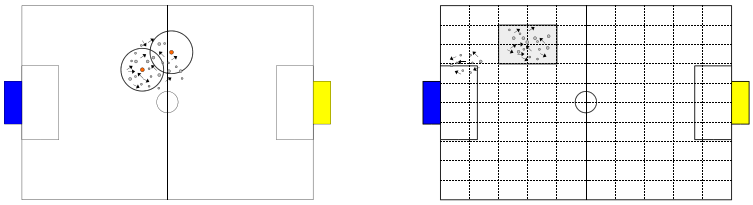
each particle represents a possible ball position and velocity. For each of these values the reliability is calculated. The update of the system is done in two phases: a prediction (time update), which is always done, and a correction step (measurement update) if a ball was seen in the current frame.

In the time update the process is represented by a constant-speed model with negative acceleration, so the new position of the particle is predicted depending on its current movement and the velocity is reduced due to the friction on the carpet. Furthermore the validity is decreased in this step so that the entire reliability of the ball state becomes lower if the ball is not seen for a long period of time.

A measurement update is done for every percept in the multiple percept list, starting with the one with the highest validity. If the ball is grabbed by the robot, the robot's position is taken as an additional percept. In this case the robot can probably not see the ball, but for sure we know that the position must be similar to the robots position.

For the update, it is distinguished between particles which are in a circular neighborhood of the current percept and those farther away (cf. fig. 4(a)). The near particles are pushed closer to the percept's position. Their reliability is increased as their position becomes more likely. The particles outside the neighborhood are not moved, but the reliability is decreased. The influence of the percept in the calculations depends on the reliability given by the image processor, the current panning velocity and the distance to the percept and these are indicators for the trust in the results of the image processor. Finally the velocity of the particles is updated depending on the last estimated ball position and the new particle position.

If a particle has a validity below a certain threshold, it is removed and thrown in again in a circular neighborhood around the last percept or—if there was no percept for a longer period of time—near the last estimated ball position. The size of the neighborhood depends on the time since the last ball percept. Moreover we make sure that at least five percent of the particles are near the



(a) Circular neighborhood in the measurement update with multiple ball percepts.

(b) Calculation of the estimated ball position.

Fig. 4. Particle representation of the ball locator.

current percept to speed-up the recovery of the estimate after a teleporting (e.g. when the ball is thrown in).

The calculation of the estimated ball position and velocity is done using a clustering approach similar to the one used in the self-locator. The field is segmented into a 10×10 grid and we only take into consideration the particles which are inside the 2×2 grid with the most particles in it (cf. fig. 4(b)). This is especially important when the ball suddenly is moved from one position to another considering all the particles would lead to an estimated position in between the two main clusters.

3.3 Team Ball Locator

The basic idea of the *team ball locator* is to send particles of the ball locator to teammates. One problem to deal with is that the network traffic would be too demanding if all robots sent all their particles. Therefore, a sub-sampled representation of the probability distribution of the ball is obtained from the set of particles (cf. fig. 5): First the particles are transformed from robot coordinates into field coordinates. This is done for a set of candidate robot poses (cf. Sect. 3.1) forwarded by the self-locator, and the probability of each robot pose is used to update the probability of the ball particles. Then the field is divided into cells. The algorithm begins splitting the field into four cells. If the particles in a cell have a low cumulative probability, the cell is not further considered. If the cell has a cumulative probability within a certain range, a representative particle is calculated by a weighted average. If the added-up probability is too high, the cell is further split up into four sub-cells and the process is recursively iterated. The algorithm stops in any case at the fourth recursive level. The representative particles are sent to the other robots in the own team, and together with those received by the other robots, a distributed ball estimate is calculated with the same algorithm as used in the ball locator.

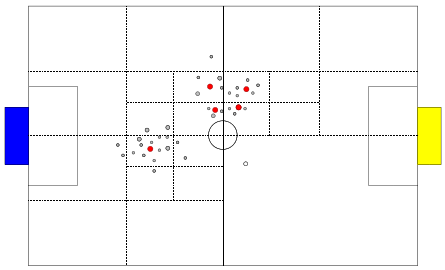
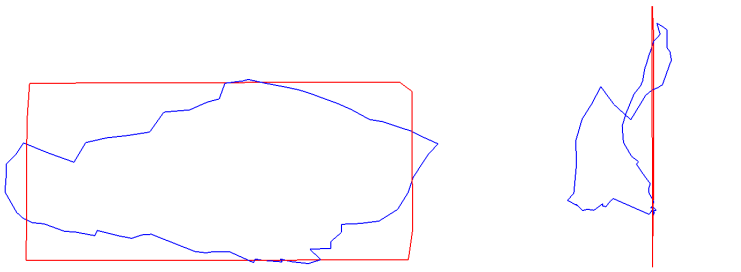


Fig. 5. Calculation of representative particles.

4 Motion Modeling and Gait Evolution

In the RoboCup domain most of the top teams' walk is inspired by the so called pwalk [8] invented by rUNSWift. This approach moves each robot leg on a two-dimensional squared locus. The required joint angles for the movement of the legs are then calculated using inverse kinematics. For omni-directional walking the legs are treated as wheels and the walking loci of the feet are rotated accordingly to the desired direction.

We compared the controlled locus of the feet and the real locus, which each foot describes with its motion, and found out that they differ significantly; the fastest walks were differing the most. These differences may result out of the idleness of the motors, kinematics errors etc. Consequently we created a more flexible walking engine which controls the feet on a path described by a three-dimensional polygon with less restrictions regarding the locus, see figure 4. This new approach increases the number of parameters significantly and thus the finding of parameters for a walk by hand is not feasible. So we applied different learning techniques to this problem and found out that the so called (μ, λ) evolution strategy with self-adapting mutation strength [9] led very quickly to a (local) optimum in this high dimensional search space. We chose a population size of $\mu = 6$ parents and $\lambda = 24$ offspring. As fitness for the individuals we used the resulting maximum speed of the robot for the according parameters. The speed was measured with a camera mounted at the ceiling. Thus, we were able to identify the fitness within three seconds for every individual. To be able to deal with measurement noise of the fitness, we generated the following generation only out of the best offspring, not including the parents. It is not feasible to search the whole parameter space, so we used a multi-start approach, searching the neighborhood of "promising" starting candidates. Finding the fastest walk with 50 cm/s needed 45 minutes of training.



(a) Hind leg XZ–plane (b) Hind leg YZ–plane

Fig. 6. Controlled (red) and real (blue) locus of a hind leg

Due to the ability to find fast walking patterns very time efficiently and the fact that different walking parameters result in different walking properties, we

decided to use different walking parameters for different requirements and to interpolate between these parameters. A set of parameters that lets the robot walk very fast straight forward in general is not very efficient for walking sideward. Thus, our current walk consists out of nine different parameter sets.

The self-localization of the robot makes intensive use of the odometry of the walk. More accurate information about the covered distance results in a better tracking of the position. We found that the correlation of the controlled and the real speed of a walking robot is due to different factors highly non-linear. To improve the self-localization we created correction tables with the effective walking speeds. The tables are 3 dimensional because of the ability to combine walking in x and y direction as well as rotation. The odometry correction has improved our self-localization significantly giving us the advantage to focus the attention more on game relevant objects such as the ball rather than objects needed to localize.

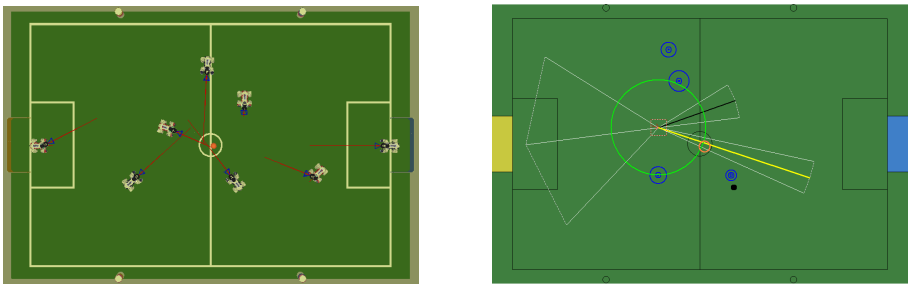
5 Behavior

5.1 XTC

The behavior architecture of the GermanTeam is still based on the successful *Extensible Agent Behavior Specification Language* (XABSL) ([10]); however, in order to reduce source code size and facilitate the learning process for new users, a new C-like language front-end has been developed, called *XTC*. This new language has the same expressive power as the previous XML-based version, generates the same intermediate code, uses the same debugging tools, and can be converted on the fly from and to the previous format. The total number of code lines for our behavior is reduced to only 47% with the new language, while the code size (in KB) is reduced to 31%.

5.2 Pass Playing

Due to the change in the RoboCup rules, last year, passing is more useful than it was in the previous years. Without the field wall, the teams which can pass the ball from one player to another have a tactical advantage. To achieve this, we developed a player model which gathers information about all seen robots—robots seen by a robot itself or by a teammate—and calculates the position of the opponent players based on this data. Then, the player model determines the free space around the robot and the possible pass corridors to the teammates. The best corridor is provided to the behavior, which decides whether a pass is played. In this case, the leading player sends out a request to his selected teammate, which turns towards the pass sender and makes himself ready for receiving. We also developed some special block movements, which minimize the ball bouncing of the robot body. In 90% of all cases, the ball lies directly in front of the robot, ready for further actions.



(a) The current game situation. (b) The pass sender players model.

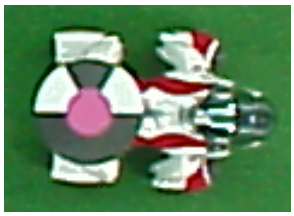
Fig. 7. Blue circles mark the opponent players. The green circle shows the free space around the robot. The rhombs are the pass corridors to the teammates. Black indicates the best rated corridor, yellow is another possible corridor, which gives a tactical advantage.

6 Tools

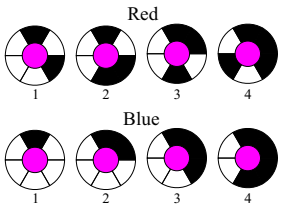
6.1 Ceiling Camera

During the year 2004, the Darmstadt Dribbling Dackels and the Microsoft Hellhounds, two of the four sub-teams the GermanTeam consists of, independently developed software which builds a global world model by interpreting the image from ceiling mounted cameras above the field. In the following the approach of the former is outlined. A detailed description can be found in [11] in German.

A high resolution fire wire camera is mounted at the ceiling of the lab, and it is equipped with a wide angle lens to enable the camera to view the whole field. The camera image is disturbed by the perspective and the optical distortions of the camera lens. These distortions are compensated for and the objects on the field are recognized.



(a) A robot equipped with a marker.



(b) Different markers allow identification and recognition of position and heading.

Fig. 8. Markers for recognition of robots by the ceiling mounted camera.

The ball is detected in a similar way as by the algorithms used by the robots themselves. The detection of the robots required the robots to be equipped with markers on their backs (cf. Fig. 8(a)). These markers were designed to minimize the distraction and obstruction of the autonomous robots. Therefore, only the colors pink, black and white were used and the marker was kept quite small. A pink disk in the center of the circular marker is used for initial detection within the image. The black and white sectors circling the pink provide a binary code which is unique even if rotated. Therefore, this code allows both the identification of the robot, i. e. team color and player number, and the computation of the rotation and by this the heading of the robot (cf. Fig. 8(b)).

The global world model containing position, heading, and speed of all recognized objects on the field is broadcasted and can be received by both the robots and PCs running the debug application "RobotControl". This allows for comparing the robots' world models with ground truth data in real time. The delay between data computed by the robots and data delivered by the ceiling-camera system has been developed is measured by using the lighting of a light bulb as a common event in both data streams and compensated for.

6.2 RobotControl 2

RobotControl 2 is a consistent enhancement of the existing debugging tool RobotControl [12], and it is the user interface to a series of new debugging mechanisms. It is written in C♯ which keeps the source easy to understand and maintain. The goal of renewing our debugging tool was to solve reoccurring debugging tasks once and for all and to decouple the source code of the debug tool from the robot code. The tool allows connecting to a real robot, to a simulated robot, or to local simulated processes that are triggered with data from a log-file. As its predecessor, RobotControl 2 includes tools for all the tasks that are required to support developing a team of robots playing 4-legged robot soccer. This involves color table creation, showing images as well as some GermanTeam-specific debugging approaches. These include the ability to (de)activate certain parts of the code, draw geometric elements on virtual drawing papers for presentation on the monitor, as well as changing the value of variables from a remote location. With the new tool the enhanced debugging mechanisms can be enabled using string based requests. Thus for example a *debug drawing* can be added anywhere with a single line of code. We implemented watch/debug mechanisms that are similar to the ones known from development tools for programming languages such as C++.

7 Conclusion

The introduction of the new field poses new challenges to teams in the league. Since the four sub-teams of the GermanTeam reached the first four places at the RoboCup German Open 2005, and the Microsoft Hellhounds won against the winner of the RoboCup American Open 2005, it seems that the techniques presented in this paper are suitable to tackle the problems in the new environment.

References

1. W. Nistico and T. Röfer, "Improving percept reliability in the Sony Four-Legged League," in *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Artificial Intelligence, Springer, 2005. to appear.
2. U. Düffert and J. Hoffmann, "Reliable and precise gait modeling for a quadruped robot," in *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Artificial Intelligence, Springer, 2005. to appear.
3. 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*, Lecture Notes in Artificial Intelligence, Springer, 2005. to appear.
4. J. Hoffmann, M. Spranger, D. Göhring, and M. Jüngel, "Exploiting the unexpected: Negative evidence modeling and proprioceptive motion modeling for improved markov localization," in *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Artificial Intelligence, Springer, 2005. to appear.
5. T. Laue, K. Spiess, and T. Röfer, "SimRobot - a general physical robot simulator and its application in RoboCup," in *RoboCup 2005: Robot Soccer World Cup IX*, Lecture Notes in Artificial Intelligence, Springer, 2005. to appear.
6. S. Lenser and M. Veloso, "Sensor resetting localization for poorly modeled mobile robots," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.
7. S. Thrun, D. Fox, and W. Burgard, "Monte carlo localization with mixture proposal distribution," in *Proc. of the National Conference on Artificial Intelligence*, pp. 859–865, 2000.
8. B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut, "Omnidirectional locomotion for quadruped robots," in *RoboCup 2001 Robot Soccer World Cup V, A. Birk, S. Coradeschi, S. Tadokoro (Eds.)*, no. 2377 in Lecture Notes in Computer Science, pp. 368–373, Springer, 2002.
9. H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.
10. M. Lötzsch, J. Bach, H.-D. Burkhard, and M. Jüngel, "Designing agent behavior with the extensible agent behavior specification language XABSL," in *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, Springer, 2004. to appear.
11. R. Brunn and M. Kunz, "A global vision system to support development in autonomous robotic soccer," 2005. Available online: http://www.sim.informatik.tu-darmstadt.de/publ/da/2005-Brunn-Kunz.pdf (Diploma thesis published in German language only).
12. T. Röfer, B. Altmeyer, R. Brunn, H.-D. Burkhard, I. Dahm, M. Dassler, U. Düffert, D. Göhring, V. Goetzke, M. Hebbel, J. Hoffmann, M. Jüngel, M. Kunz, T. Laue, M. Lötzsch, W. Nisticó, M. Risler, C. Schumann, U. Schwiegelshohn, M. Spranger, M. Stelzer, O. von Stryk, D. Thomas, S. Uhrig, and M. Wachter, "GermanTeam RoboCup 2004," tech. rep., 2004. Available online: http://www.germanteam.org/GT2004.pdf.