GermanTeam 2003

Thomas Röfer[1], Ingo Dahm[2], Uwe Düffert[3], Jan Hoffmann[3], Matthias Jüngel[3], Martin Kallnik[4], Martin Lötzsch[3], Max Risler[4], Max Stelzer[4], and Jens Ziegler[5]

[1] Bremer Institut für Sichere Systeme, Technologie-Zentrum Informatik, FB 3, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
[2] Lehrstuhl für Systemanalyse, FB Informatik, University of Dortmund, Joseph-von-Fraunhofer-Strasse, 44221 Dortmund, Germany
[3] Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin, Rudower Chaussee 25, 12489 Berlin, Germany
[4] Fachgebiet Simulation und Systemoptimierung, FB 20 Informatik, Technische Universität Darmstadt, Alexanderstraße 10, 64283 Darmstadt, Germany
[5] Lehrstuhl für Datenverarbeitungssysteme, FB Elektrotechnik und Informationstechnik, University of Dortmund, Otto-Hahn-Strasse 4, 44221 Dortmund, Germany

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

1 Introduction

The GermanTeam participates as a national team in the Sony Legged Robot League. It currently 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 separate teams in the national contests such as RoboCup German Open, but jointly line up for the international RoboCup championship as a single team. To support this cooperation and concurrency, the GermanTeam introduced an architecture that provides mechanisms for parallel development [1]. The entire information processing and control of the robot is divided into *modules* that have well-defined tasks and interfaces. For each module, many different *solutions* can be developed. Solutions for a module can be switched at runtime. Currently, for most modules various solutions exist. Approaches to a problem can thereby easily be compared and benchmarked.

This paper gives a brief overview of the work done in the past year. A detailed description of the entire system used in the competition—including its underlying architecture—can be found at [2].

2 Vision

The vision module determines so-called percepts from the images taken by the robot's camera. Percepts are the position of the ball, the field lines, the goals, the flags, the other players, and the obstacles, all given in an egocentric frame of reference.

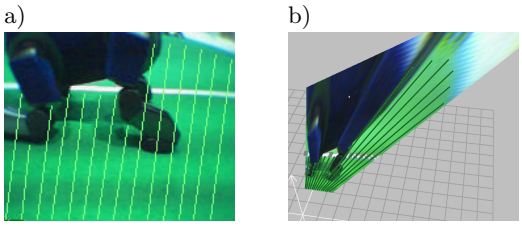

Fig. 1. Obstacles Percept. a) An image with an obstacle. Green lines: projection of the obstacles percept to the image. b) The projection of the image to the ground. Green lines: obstacles percept.

Goals and flags are represented by four angles. These describe the bounding rectangle of the landmark (top, bottom, left, and right edge) with respect to the robot in angular coordinates. Field lines are represented by a set of 2-D points on a line. The ball position and also the other players' positions are represented in 2-D coordinates. The free space around the robot is represented in the *obstacles percept* that consists of a set of lines described by a *near point* and a *far point* on the ground. The lines describe green segments in the projection of the camera's image to the ground. The lines usually start at the image bottom and end where the green of the ground ends or where the image ends (cf. Fig. 1b). If the part of the projection of the image that is close to the robot is not green, both points are identical and lie on the rim of the projection. If the far point is not on the border of the image, there is an obstacle behind that point. The area between the near and the far point is free space. It is unknown whether there is an obstacle before the near point.

The YUV images taken by the camera are processed using the high resolution of 176×144 pixels, but looking only at a grid of less pixels. The idea is that for feature extraction, a high resolution is only needed for small or far away objects. In addition to being smaller, such objects are also closer to the horizon (see below). Thus only regions near the horizon need to be scanned at high resolution, while the rest of the image can be scanning using a relatively wide spaced grid.

Grid Construction and Scanning. First the position of the horizon in the image is calculated (cf. Fig. 2b). The horizon is the line that is parallel to the field plane, but at the height of the camera. The grid is constructed based on the horizon line, to which grid lines are perpendicular and in parallel. The area near the horizon has a high density of grid lines, whereas the grid lines are coarser in the rest of the image (cf. Fig. 2b).

Each grid line is scanned pixel by pixel from top to bottom. During the scan each pixel is classified by color. A characteristic series of colors or a pattern of colors is an indication of an object of interest, e. g., a sequence of some orange pixels is an indication of a ball; a sequence or an interrupted sequence of pink pixels followed by a green, sky-blue, yellow, or white pixel is an indication of a

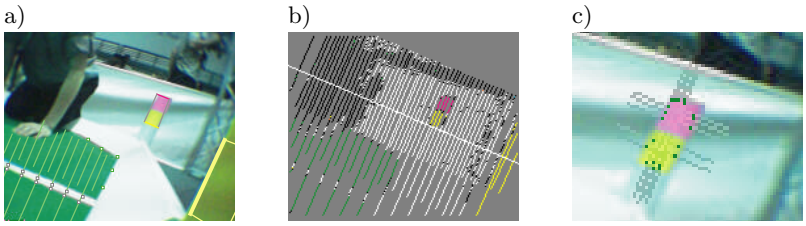

Fig. 2. Percepts. a) An image and the recognized objects. Dots: pixels on a field line or a border, Flag: pink above yellow, Goal: One edge inside the image and three edges that intersect with the image border. Green lines: obstacles percept. b) The used scan lines. c) Recognition of the flag. Only the gray pixels have been *touched* by the flag specialist. The green pixels mark the edges recognized.

flag; an (interrupted) sequence of sky-blue or yellow pixels followed by a green pixel is an indication of a goal, a sequence of white to green or green to white is an indication of an edge between the field and the border or a field line, and a sequence of red or blue pixels is an indication of a player. All this scanning is done using a kind of state machine; mostly counting the number of pixels of a certain color class and the number of pixels since a certain color class was detected last. That way, beginning and end of certain object types can still be determined although some pixels of the wrong class are detected in between.

Detecting Points on Edges. As a first step toward a more color table independent classification, points on edges are only searched at pixels with a big difference of the y-channel of the adjacent pixels. An increase in the y-channel followed by a decrease is an indication of an edge. If the color above the decrease in the y-channel is sky-blue or yellow, the pixel lies on an edge between a goal and the field. The differentiation between a field line and the border is a bit more complicated. In most of the cases the border has a bigger size in the image than a field line. But a far distant border might be smaller than a very close field line. For that reason the pixel where the decrease in the y-channel was found is assumed to lie on the ground. With the known height and rotation of the camera the distance to that point is calculated. The distance leads to expected sizes of the border and the field line in the image. For the classification these sizes are compared to the distance between the increase and the decrease of the y-channel in the image. The projection of the pixels on the field plane is also used to determine their relative position to the robot.

All vertical scan lines are searched for edges between the goals and the field, because these objects are small and often occluded by robots. In contrast, only every fourth vertical scan line is searched for edge points on the border or on the field lines, but also a few horizontal scan lines are searched for these edge types, because they can appear in vertical direction in the image. As the scanning algorithm assumes to find the border before the field, which is not always true for horizontal scanning, the horizontal scan lines are scanned either from left to

right or from right to left by random. If less then three points of a certain edge type are detected in the image, these points are ignored to reduce noise. If many more points on field lines than on the border are detected, the points on the border are dropped, because they are assumed to be misclassified.

Detecting the Ball. For balls, upper and lower points on their boundaries are detected during scanning. Points on the border of the image are ignored. During scanning, red pixels below a reasonable number of orange pixels are also treated as orange pixels, because shaded orange often appears as red. Although only a single ball exists in the game, the points are clustered before the actual ball position is detected, because some of them may be outliers on the tricot of a red robot. To remove outliers in vertical direction, upper points are ignored if they are below many other lower points in the same cluster, and lower points are ignored if they are above many other upper points in the same cluster.

If enough points have been found, the center of the ball is determined by intersecting the middle-perpendiculars. Otherwise, the center is estimated as the middle between all points found. If the ball is not below the horizon or if the camera position is not stable because the robot is currently kicking, the distance to the ball is determined from its radius. Otherwise, the distance is determined from the intersection of the ray that starts in the camera and points to the center of the ball with a plane that is parallel to the field, but on the height of the ball center.

Detecting Flags. All indications for flags found during scanning the grid are clustered. In each cluster there can actually be indications for different flags, but only if one flag got more indications than the others, it is actually used. The center of a cluster is used as a starting point for the *flag specialist*. It measures the height and the width of a flag. From the initialization pixel the image is scanned for the border of the flag to the top, right, down, and left where top/down means perpendicular to the horizon and left/right means parallel to the horizon (cf. Fig. 2c). This leads to a first approximation of the size of the flag. Two more horizontal lines are scanned in the pink part and if the flag has a yellow or a sky-blue part, two more horizontal lines are also scanned there. The width of the green part of the pink/green flags is not used, because it is not always possible to distinguish it from the background. To determine the height of the flag, three additional vertical lines are scanned. The leftmost, rightmost, topmost, and lowest points found by these scans determine the size of the flag.

Detecting Goals. A goal *specialist* measures the height and the width of a goal. The image is scanned for the borders of the goal from the left to the right and from the top bottom, where again top/down means perpendicular to the horizon and left/right parallel to the horizon. To find the border of the goal the specialist searches the last pixel having the color of the goal. Smaller gaps with unclassified color are accepted. The maximal size in each direction determines the size of the goal.

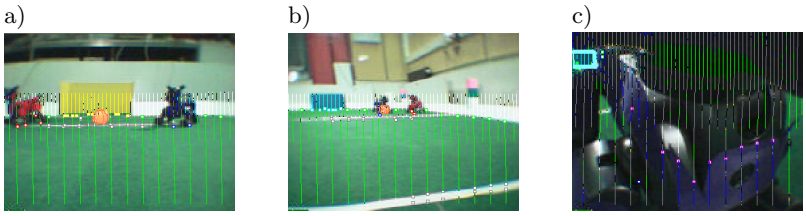

Fig. 3. Recognition of other robots. a) Several foot points for a single robot are clustered (shown in red and blue). b) Distant robots are still recognized. c) Close robots are recognized based on the upper border of their tricot (shown in pink).

Detecting Robots. To determine the indications for other robots, the scan lines are searched for the colors of the tricots of the robots. If the number of pixels in tricot color found on a scan line is above a certain threshold, it is assumed that the other robot is close. In that case, the upper border of its tricot (ignoring the head) is used to determine the distance to that robot (cf. Fig. 3c). As with many other percepts, this is achieved by intersecting the view ray through this pixel with a plane that is parallel to the field, but on the "typical" height of a robot tricot. As the other robot is close, a misjudgment of the "typical" tricot height does not change the result of the distance calculation very much. As a result, the distance to close robots can be determined.

If the number of pixels in tricot color found is smaller, it is assumed that the other robot is further away. In that case, the scan lines are followed until the green of the field appears (cf. Fig. 3a, b). Thus the *foot points* of the robot are detected. From these foot points, the distance to the robot can be determined by intersecting the view ray with the field plane. As not all foot points will actually be below the robot's feet (some will be below the body), they are clustered and the smallest distance is used.

Detecting Obstacles. While the scan lines are scanned from top to bottom, a state machine determines the last begin of a green section. If this green section meets the bottom of the image, the begin and the end points of the section are transformed to coordinates relative to the robot and written to the obstacles percept; else or if there is no green on that scan line, the point at the bottom of the line is transformed and the near and the far point of the percept are identical. Inside a green segment, an interruption of the green that has the size of $4 \cdot width_{fieldline}$ is accepted to assure that field lines are not misinterpreted as obstacles ($width_{fieldline}$ is the expected width of a field line in the image depending on the camera rotation and the position in the image).

3 New Approaches to Color Classification

Many image processing approaches in the color labeled RoboCup domain utilize a static color table to color segment the camera image, e. g. the one that was

Fig. 4. Color classification of two camera images (left) as done by a conventional YUV-approach (middle) compared with an optimized classification (right)

still used by the GermanTeam in RoboCup 2003. Creating such a color table is a time consuming, tedious job. Furthermore, a color table created for a certain lighting situation will produce unsatisfactory results when lighting conditions change.

Therefore, the GermanTeam has laid the foundation for new approaches to color classification. On the one hand, we developed an evolutionary chrominance-space optimization: Starting from the TSL-chroma space, we evolve an optimized transformation, such that the main colors (green, pink, yellow etc.) are located in easy-to-separate subspaces [3]. This reduces the algorithmic complexity of color segmentation and improves classification accuracy significantly, since the evolved chrominance space is robust against luminance variation (cf. Fig. 4).

A different approach utilizes a qualitative color table where colors are labeled with respect to a reference color [4]. The reference color is calibrated using simple geometric heuristics in the camera image (cf. Fig. 5). We use green as a reference color because it is found in all images. The color calibration is done in real time providing the vision system with auto-adaptation capabilities. In the color calibration process, simple heuristics are used to ensure that only pixels that are thought to be green are used; pixels not satisfying these strict conditions are omitted. The qualitative approach to classify colors means that rather then segmenting individual pixels, edges are classified. Classification of edges can be done reliably even when using relatively vague color information. Similar to [5], only pixels on a grid are classified to increase performance.

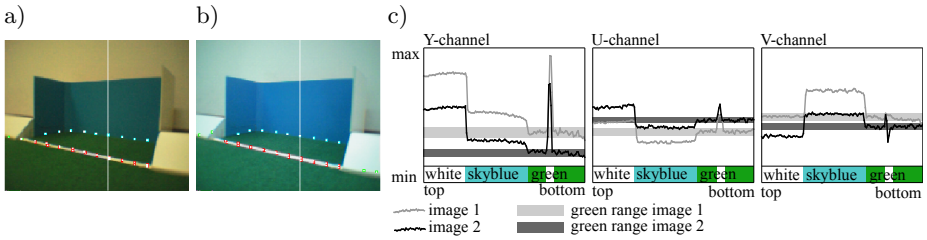

Fig. 5. Images recorded under different lighting conditions with a highlighted scan line and recognized points (border, field line, goal). a) Day light. b) Artificial light. c) Intensities of the YUV-channels along the scan line for both images. The gray bars show the range of green after auto-adaptation for left and right image.

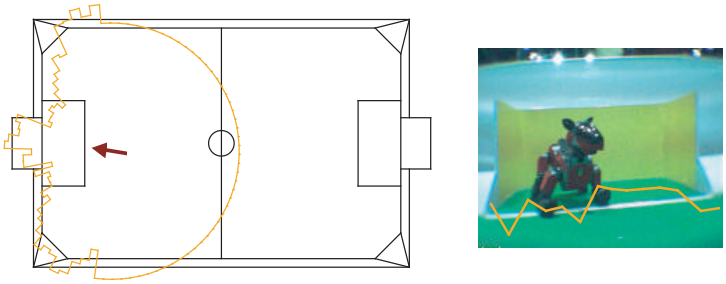

Fig. 6. The obstacle model as seen from above and projected into the camera image. The robot is in front of the opponent goal.

4 Obstacle Model

In the obstacles model, a radar-like view of the surroundings of the robot is created. To achieve this, the surroundings are divided into 90 (micro-) sectors. For each of the sectors the free distance to the next obstacle is stored (see Fig. 6). In addition to the distance, the actual measurement that resulted in the distance is also stored (in x,y-coordinates relative to the robot). These are called representatives. Each sector has one representative.

For most applications, the minimum distance in the direction of a single sector is not of interest but rather the minimum value in a number of sectors. Usually, a sector of a certain width is sought-after, e. g. to find a desirable direction free of obstacles for shooting the ball. Therefore, the obstacle model features a number of analysis functions (implemented as member functions) that address special needs for obstacle avoidance and ball handling. One of the most frequently used functions calculates the free distance in a corridor of a given width in a given direction. This can be used to check if there are any obstacles in the direction the robot is moving in and also if there's enough room for the robot to pass through.

The obstacle model is updated using sensor data and robot odometry data. Good care has to be taken when integrating new sensor data into the model as is illustrated in Fig. 7. Early versions also used the PSD distance sensor. This was later omitted because the information gained from the camera image was more than sufficient.

Obstacle avoidance based on the described model was used in the RoboCup "Obstacle Avoidance" challenge in which it performed extraordanary well (we won the challenge). It did, however, prove to be difficult to make good use of the information in non-static, competitive situations. One example to illustrate this is the case of two opposing robots going for the ball: in this case, obstacle avoidance is not desirable and would cause the robot to let the other one move forward. Studies have to be conducted to find ways of using obstacle avoidance in a way suitable for the RoboCup environment.

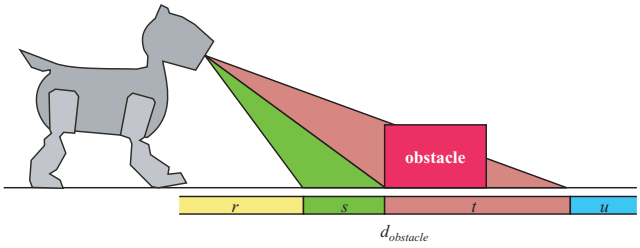

Fig. 7. The above diagram depicts the robot looking at an obstacle. The robot detects some free space in front of it (s) and some space that is obscured by the obstacle (t). The obstacle model is updated according to the diagram (in this case the distance in the sector is set to $d_{obstacle}$ unless the distance value stored lies in r).

5 Localization

Already in 2002, the GermanTeam had a robust and precise localization method based on the famous Monte-Carlo localization approach [6]. For instance, it enabled the robots to autonomously position themselves for a kick-off. The method relies on the recognition of colored landmarks and goals around the field. However, as there are no colored marks on a real soccer field, the GermanTeam developed a Monte-Carlo localization approach that is based on the recognition of edges between the field and lines, goals, and the border. The method especially takes account of the fact that different types of edges provide different information on the robot's position and that they are seen with variable frequency. During experiments, in which the position calculated by the robot were compared to one delivered by an external reference, it proved to reach an average error of less than 10.5 cm on a continuously moving robot, and it was able to reach goal positions with an average error of less than 8.4 cm [7]. At RoboCup 2003, the GermanTeam played with a combined self-locator using both landmarks and edges, resulting in high precision. In addition, a game without colored landmarks used for localization was demonstrated, just before the RoboCup "Localization" challenge, in which the GermanTeam reached the third place using the edge-based localization approach.

6 Behavior Control

For behavior control the GermanTeam uses the *Extensible Agent Behavior Specification Language* XABSL [8, 9] since 2002 and improved it largely in 2003. For the German Open 2003, each of the four universities of the GermanTeam used XABSL for behavior control and continued the behaviors that were developed by the GermanTeam for Fukuoka. After the German Open 2003 the behaviors of the teams could easily be merged into a common solution that was continued until the RoboCup 2003 in Padova.

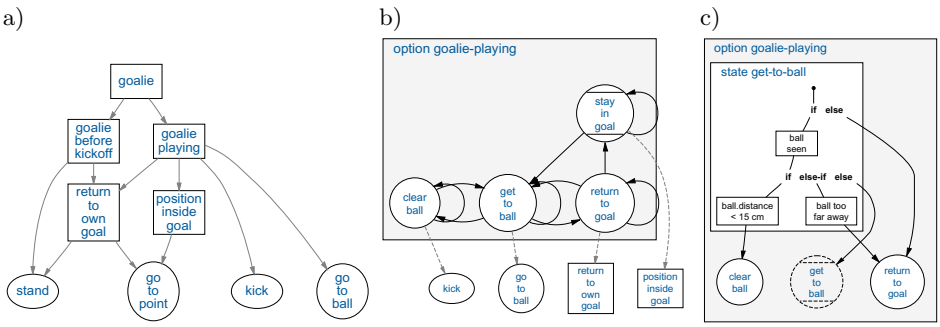

Fig. 8. a) The option graph of a simple goalie behavior. Boxes denote options, ellipses denote basic behaviors. The edges show which other option or basic behavior can be activated from within an option. b) The internal state machine of the option *"goalie-playing"*. Circles denote states, the circle with the two horizontal lines denotes the initial state. An edge between two states indicates that there is at least one transition from one state to the other. The dashed edges show which other option or basic behavior becomes activated when the corresponding state is active. c) the decision tree of the state *"get-to-ball"*. The leaves of the tree are transitions to other states. The dashed circle denotes a transition to the own state.

The Extensible Agent Behavior Specification Language XABSL is an XML based behavior description language. XABSL can be used to describe behaviors of autonomous agents. XABSL simplifies the process of specifying complex behaviors and supports the design of both very reactive and long term oriented behaviors. The runtime system XabslEngine executes the behaviors on a target platform.

In XABSL, an agent consists of a number of behavior modules called *options*. The options are ordered in a rooted directed acyclic graph, the *option graph* (cf. 8a). The terminal nodes of that graph are called basic behaviors. They generate the actions of the agent and are associated with basic skills.

The task of the option graph is to activate and parameterize one of the basic behaviors, which is then executed. Beginning from the root option, each active option has to activate and parameterize another option on a lower level in the graph or a basic behavior.

Within options, the activation of behaviors on lower levels is done by state machines (cf. 8b). Each state has a subsequent option or a subsequent basic behavior. Note that there can be several states that have the same subsequent option or basic behavior.

Each state has a *decision tree* (cf. 8c) with transitions to other states at the leaves. For the decisions the agent's world state, other sensory information and messages from other agents can be used. As timing is often important, the time how long the state is already active and the time how long the option is already active can be taken into account.

The execution of the option graph starts from the root option of the agent. For each option the state machine is carried out one times, the decision tree of the active state is executed to determine the next active state. This is continued for the subsequent option of the active state and so on until a basic behavior is reached and executed.

The Behavior of the GermanTeam is documented in an extensive HTML documentation of the GermanTeam behaviors generated automatically that can be found under *http://www.ki.informatik.hu-berlin.de/XABSL/examples/gt2003/index.html*. Here, only three aspects of the behavior are emphasized:

Negotiations and Dynamic Role Assignment. The three field players negotiate which of them is the striker, the offensive supporter, or the defensive supporter. This is done depending on the estimated time that each robot needs to approach the ball. This time is influenced by the distance to the ball, the angle to the ball, the time since the ball was seen last, and the obstacle model.

Use of Obstacle Model. The GermanTeam uses the obstacle avoidance that was shown in the obstacle avoidance challenge during the whole games. It is used during positioning, ball searching and ball approaching. Only in the near of the ball (less than 70 cm) it is switched off.

The kicks are also chosen dependent on the obstacle model. If there are many other robots in the near of the robot, the striker does not try to grab the ball and kick it exactly. If there are obstacles in line to the opponent goal, the robot tries to kick the ball to a teammate.

Continuous Basic Behaviors. Besides simple basic behaviors that execute a discrete action like performing a kick or walking in one direction, there are more complex behaviors that allow pursuing multiple objectives simultaneously while leading over from one action to another continuously. Such a basic behavior, e. g., could move towards the ball, while at the same time it is avoiding to run into obstacles by moving away from them.

These behaviors are called continuous basic behaviors and are implemented following a potential field approach. The potential field defining one basic behavior is configured by putting together several so-called rules, i. e. components representing single aspects of the desired behavior. Each rule is a simple potential field either attractive to a target position or repulsive from one or multiple objects. These rules may be e. g. going to the ball or avoiding running into the own penalty area.

The potential fields of all rules for one basic behavior are superposed resulting in one field which is evaluated at the position of the robot to generate the current walking direction and speed (cf. Fig. 9).

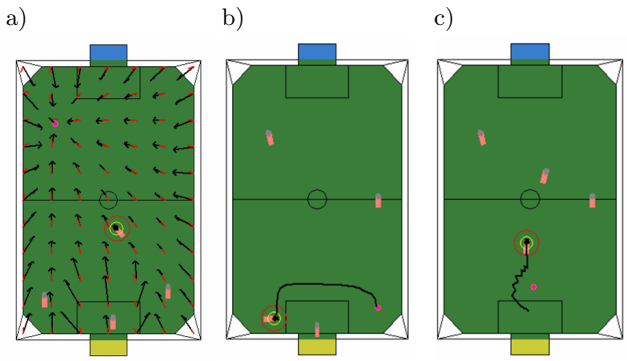

Fig. 9. Continuous basic behaviors. a) Potential field for going to the ball while avoiding running into the own penalty area. b) Resulting walking path demonstrating the go to ball basic behavior. c) Walking path for a basic behavior going to the own goal without running over the ball.

7 Conclusion and Future Work

In 2003, the main innovations by the GermanTeam were the introduction of using edges for self-localization, detecting and modeling obstacles to improve game play, using potential fields to implement basic behaviors, and using dynamic role assignments. Additional innovations, not used during the games this year, but that will hopefully be used next year, are the automatic color calibration, and the detection of collisions.

The GermanTeam finished the round robin as winner of its group, even against the later runner-up, the UPennalizers. In the quarter final, the GermanTeam lost in a 29 minutes penalty shootout against CMPack'03. However, the GermanTeam won the RoboCup Challenge with 70 out of 72 possible points, i. e. it reached the first places in the "Black and White Ball" challenge and the "Obstacle Avoidance" challenge and the third place in the "Localization without Colored Beacons" challenge.

The robot vision and localization used deliver highly accurate results [5, 7]; they do, however, highly rely on the use of manually calibrated color tables. Work on an auto-adjusting vision system has been started [4] and it is planned to use it in next year's competition.

We are currently working on the implementation of optimal gait trajectories, which are solutions of an optimal control problem involving a dynamical model of the robot and several boundary and nonlinear implicit conditions [10, 11]. Efficient dynamics algorithms like Articulated Body Algorithm are used to evaluate the dynamics of the full three dimensional dynamical model efficiently. The problem formulation for generating stable gaits that minimize time or energy subject to the legged robot dynamics leads to an optimal control problem, which is solved numerically by a parameterization, collocation and sparse nonlinear optimization approach. First experiments have started and lead to an

improvement of the model by avoiding slipping and taking into account more detailed characteristics of the motors. These optimizations are computed offline. The resulting optimized trajectories are then reproduced on the robot.

References

1. T. Röfer, "An architecture for a national robocup team," in *RoboCup 2002: Robot Soccer World Cup VI* (G. A. Kaminka, P. U. Lima, and R. Rojas, eds.), Lecture Notes in Artificial Intelligence, (Fukuoka, Japan), pp. 417–425, Springer, 2003.
2. T. Röfer, H.-D. Burkhard, U. Düffert, J. Hoffmann, D. Göhring, M. Jüngel, M. Lötzsch, O. v. Stryk, R. Brunn, M. Kallnik, M. Kunz, S. Petters, M. Risler, M. Stelzer, I. Dahm, M. Wachter, K. Engel, A. Osterhues, C. Schumann, and J. Ziegler, "GermanTeam RoboCup 2003," tech. rep., 2003. Only available online: http://www.robocup.de/germanteam/GT2003.pdf.
3. I. Dahm, S. Deutsch, M. Hebbel, and A. Osterhues, "Robust color classification for robot soccer," in *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, (Padova, Italy), Springer, 2004. to appear.
4. M. Jüngel, J. Hoffmann, and M. Lötzsch, "A real-time auto-adjusting vision system for robotic soccer," in *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, (Padova, Italy), Springer, 2004.
5. T. Röfer and M. Jüngel, "Vision-based fast and reactive Monte-Carlo localization," in *Proc. of IEEE International Conference on Robotics and Automation*, (Taipei, Taiwan), pp. 856–861, IEEE, 2003.
6. 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.
7. T. Röfer and M. Jüngel, "Fast and robust edge-based localization in the Sony four-legged robot league," in *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, (Padova, Italy), Springer, 2004. to appear.
8. 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, (Padova, Italy), Springer, 2004. to appear.
9. M. Lötzsch, "The XABSL web site," http://www.ki.informatik.hu-berlin.de/XABSL.
10. M. Hardt and O. von Stryk, "The role of motion dynamics in the design, control and stability of bipedal and quadrupedal robots," in *RoboCup 2002: Robot Soccer World Cup VI* (G. A. Kaminka, P. U. Lima, and R. Rojas, eds.), Lecture Notes in Artificial Intelligence, (Fukuoka, Japan), pp. 206–223, Springer, 2003.
11. M. Hardt, M. Stelzer, and O. von Stryk, "Efficient dynamic modeling, numerical optimal control and experimental results for various gaits of a quadruped robot," in *CLAWAR 2003 - 6th International Conference on Climbing and Walking Robots*, (Catania), 2003.