GermanTeam 2006

The German National RoboCup Team

Thomas Röfer[1], Jörg Brose[2], Eike Carls[3], Jan Carstens[3], Daniel Göhring[4], Matthias Jüngel[4], Tim Laue[3], Tobias Oberlies[2], Sven Oesau[3], Max Risler[2], Michael Spranger[4], Christian Werner[2], and Jörg Zimmer[2]

[1] Deutsches Forschungszentrum für Künstliche Intelligenz, Safe and Secure Cognitive Systems, Robert-Hooke-Str. 5, 28359 Bremen, Germany
[2] Fachgebiet Simulation und Systemoptimierung, Fachbereich Informatik, Technische Universität Darmstadt, Hochschulstraße 10, 64289 Darmstadt, Germany
[3] Fachbereich 3 - Mathematik / Informatik, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany
[4] Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin, Rudower Chaussee 25, 12489 Berlin, Germany.

http://www.germanteam.org
germanteam@tzi.de

1 Introduction

The GermanTeam participates as a national team in the RoboCup Four-Legged League. It currently consists of students and researchers from the following three universities: the Humboldt-Universität zu Berlin, the Universität Bremen, and the Technische Universität Darmstadt. The members of the GermanTeam participate as individual teams in contests such as the RoboCup Dutch Open, but jointly line up as a national team for the international RoboCup World Cup. 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* (cf. fig. 1) carrying out specific tasks using well-defined interfaces. For each module, many different *solutions* can be developed which can be switched at runtime. This approach allows for easily comparing and benchmarking the solutions developed for the various tasks. It is used since the fall of 2001 and has been proven to be a very useful tool for development. At RoboCup 2006, 10 out of the 28 teams that will be present base their work on this architecture.

This paper gives an overview on the work done by the three sub-teams of the GermanTeam (Aibo Team Humboldt, Darmstadt Dribbling Dackels, and Bremen Byters) in the past year that is currently combined to form the code of the GermanTeam 2006. Further information can be found in this year's contributions of members of the GermanTeam to the RoboCup book:

– In [2], a Kalman filter-based approach for estimating the positions of other robots is presented. By combining different simple perceptions, accurate estimates of a robot's local environment can be computed.

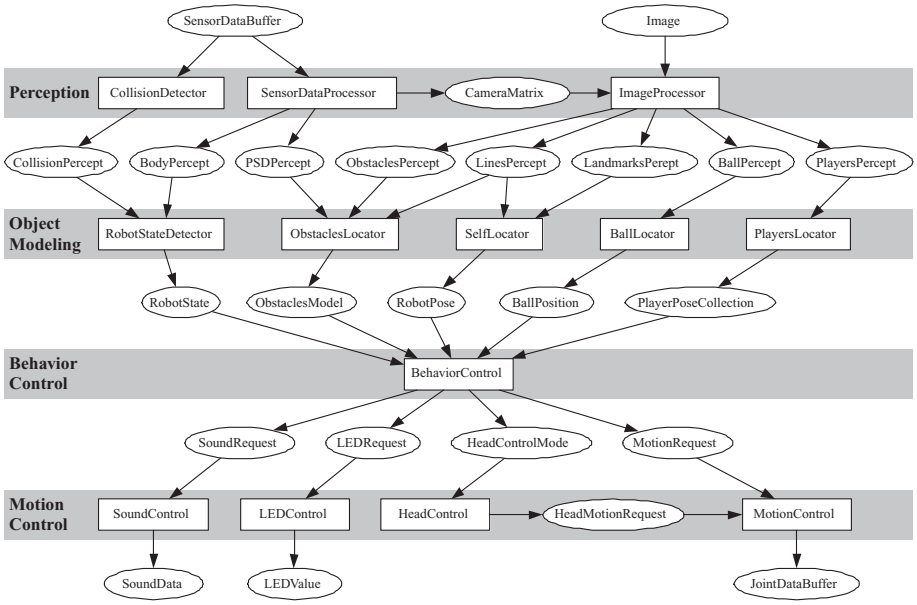

Fig. 1. Information processing in the robots of the GermanTeam. Boxes denote *modules*, ellipses denote the *representations* that are needed to exchange information between the modules.

- In [3] an approach for using and communicating relations between objects in robot images is presented that allows allocentrically localizing objects without the need for self-localization.
- An approach, how information about collisions can be used for an improved self localization was described in [4].

2 Perception

2.1 Hotspot-based Image Processing

The GT2005 image processing architecture was designed to find necessary information on a grid of scan lines in a single sweep. The central module calls so-called specialists to recognize percepts if enough evidence is gathered, e. g. if some orange pixels were found, this calls for further examination by the ball specialist. This approach has the benefit of scanning most pixels on scan lines only once, although specialists may check it again to verify their percept. The main disadvantage is that this central module grows more and more complicated while the image processor evolves.

To ease the development of new perceptors the central module with its set of scan lines was discarded and each perceptor got its own specialized set of scan lines. This approach made it possible to swap, e. g., only the orange ball

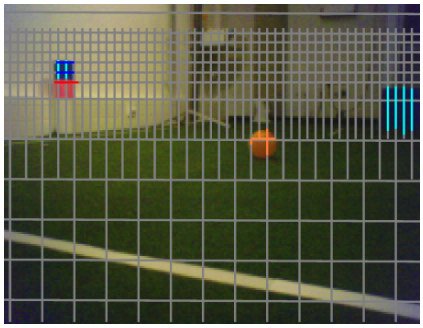

Fig. 2. Hotspots found (orange, pink, etc.) on a horizon-aligned grid (grey).

perceptor for a black and white ball perceptor. Its main disadvantage was that several lines were scanned more than once since the requirements of some of the perceptors are similar. Having split the image processing into several independent modules—the perceptors and a percept filter—improved maintainability but decreased execution time thereby making it unfeasible for competitive use.

The hotspot approach tries to combine both the versatility and maintainability of the split perceptors with the speed of the old architecture by introducing another layer of abstraction between the actual image and the percepts. This was accomplished by reintroducing the central instance for checking scan lines, the *hotspot finder*. This module finds interesting regions (cf. Fig. 2), so-called hotspots, on its scan lines and passes them to the perceptors. Interesting is defined for each percept in a function that can be easily modified. The perceptors use the hotspots as a starting point for finding the percept rather than merely verifying them as they did in the 2005 version of the image processor . Since each perceptor can access any kind of information gathered by the hotspot finder, this approach is rather easy to extend while keeping most functions small and simple.

2.2 Detecting Robots

To determine indications of other robots, vertical scan lines are searched for tricot colors. In order to avoid wrong scan lines (especially for blue robots), only those scan lines are used that begin with white color. In addition, all scan lines that end with green color, and the green appears to be too far away (based on an intersection with the field plane), are ignored.

After collecting all vertical scans of robot uniforms, these scans are clustered. If the distance between two neighboring uniform scans is too large, it is tested whether there is a horizontal white line connecting these scans. If this is not the case, the scans belong to different robots, otherwise it is assumed that they belong to the same one.

In order to determine the position of a robot, the contact point between the robot and the carpet is computed. Therefore, it is searched for the feet of the robot. This procedure scans the first and the last scan line belonging to the robot

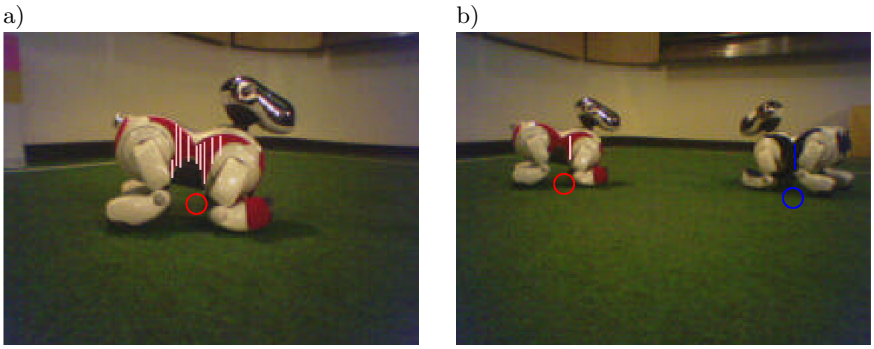

Fig. 3. Detecting Robots. The colored circles represent the positions of the robots. a) A close robot. b) Two robots further away.

downward using a scan tree along the white color. The lowest white/green edge found is assumed to be the foot point of the robot, i. e. it is on the height of the field.

Finally the robot's position is determined by constructing a point in the image by intersecting the middle of the vertical scans belonging to robot with a horizontal line on the height of the foot point. This image point is transformed to world coordinates by intersecting a ray from the camera origin through this image point with the field plane.

2.3 Detecting Black and White Balls

Related Work. To recognize a black and white ball, the algorithm to detect the orange ball is not applicable, because there is no unique color on the black and white ball such as orange for the original ball. The method developed by the GermanTeam in 2003 (Röfer et al., 2003, pages 66-67) follows a rather classical approach to detect the ball. It first detects edges using the Sobel operator followed by some kind of thresholding, so only edges of significant contrast are taken into account (cf. Fig. 4a). Then a Hough transformation is performed. The Hough transformation accumulates evidence for the presence of certain features at certain positions in the Hough space. In case of the ball detection, the Hough space consists of the dimensions (x, y, r), in which (x, y) is the location of the center point of the ball in image-coordinates and r is its radius. For each edge point with sufficient contrast in the image, the evidence of all entries in Hough space this point can be part of is increased. Basically, this means that circles of the different radii are drawn in the Hough space (or circle segments if the direction of the edge points can be detected and is used to narrow the possible position of the center point). After all suitable edge points have been inserted into Hough space, the largest cluster is searched and it is checked whether it provides enough evidence for being a ball or not, maybe by looking back in the image at this position and checking whether there is a ball of the assumed radius.

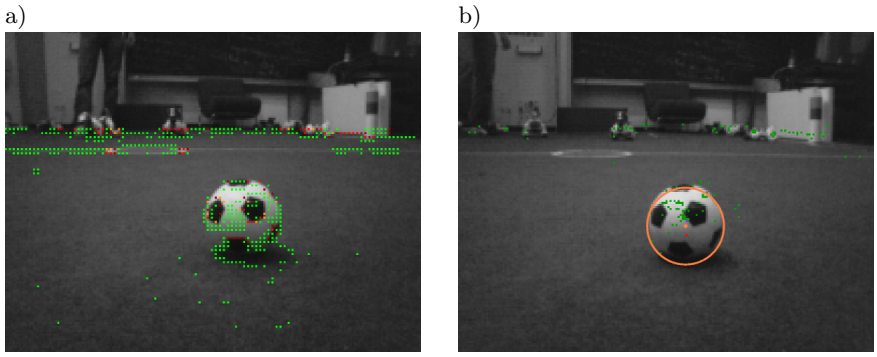

Fig. 4. Searching the image for edges. a) Edges detected. The intensity of the edges increases from green over red to orange. b) Searching for the ball center. Green points are possible candidates, red points were discarded. The orange circle depicts the resulting ball percept.

The major drawback of this approach is that it is quite slow. With many edge points present, many entries in the Hough space have to be modified. In addition, the number of possible radii has to be significantly limited to have any chance to run in real time.

Approach. To avoid a three dimensional Hough space, it is required to directly calculate a single ball center from the edge points found by the Sobel operator. The real ball radius and the orientation of the camera are known. So if one assumes that the ball is on the ground, it is possible to calculate the ball center point based on the location of the edge and the gradient. The calculation of the ball center point is quite expensive, but it allows the use of a two dimensional Hough space and only needs one entry in the Hough space per edge point. Another advantage is that it is able to work with all image space ball radii and not only with a few ones.

Calculating the Ball Center. The major problem is, given an edge point and its gradient, to determine where the center of a ball with this point on its edge would be. The position of the camera is given by a three-dimensional vector and a 3×3 matrix representing its rotation. Using this information and the x/y-coordinates of the edge points, the view ray from the camera origin through this particular pixel can be constructed. From this view ray and the gradient of the edge point a plane can be constructed that touches the ball from the outside. At the contact point, a vector perpendicular to the plane with a length of the ball radius directly points to the center of the ball.

The ball is also touched by a second plane, i. e. the field plane. Again, a vector perpendicular to that plane with a length of the ball radius that is simply $(0, 0, r_{ball})$, also points to the center of the ball. So we can both subtract the

vector perpendicular to the edge point plane and the vector $(0, 0, r_{ball})$ from the position of the camera. Starting from the resulting position, we intersect the view ray with the plane $z = 0$. The result is the x/y-offset of the center of the ball, relative to the origin of the robot. For visualization purposes, it can be projected back into the image (cf. Fig. 4b).

Optimizations. As the calculation of the ball center is quite expensive, its useful to minimize the number of edge points processed. In all cases when the AIBO is looking for the ball, the camera is above the ball. This makes it possible to ignore all edge points above the horizon in the camera image. Another optimization technique is to only care about green/white transitions and green/black transitions, but this only results in a small acceleration, because most edges found are of these types anyway (field lines/field, AIBO/field).

3 Ball Model

3.1 Motivation

Tracking the ball and to estimating its speed are some of the most important abilities when playing robot soccer. The motion of the ball can either be linear (the ball rests or moves steadily) or highly non-linear (the ball has been kicked by a robot). To be able to track the ball in these different situations, different motion models are used. The ball state can then be represented by a probability density in the five dimensional state space—four dimensions for the ball position and speed plus one dimension for the ball mode.

3.2 Method

To achieve this, we use a Rao-Blackwellised Particle Filter, a method first applied in the RoboCup domain by Kwok and Fox [5]. Such a filter maintains a set of particles that represent the posterior over the ball state. Each of them uses a Kalman filter to model the position and speed conditioned on the discrete motion mode of that particle. This means that there are different time updates for each ball mode while the measurement update is the same for all particles. Note that the two-dimensional position measurement is sufficient for the Kalman filters to update position and velocity.

3.3 Ball Modes

Currently we use following ball modes to represent the possible situations. Due to the modular framework of our ball locator, these list can be be easily changed.

None: The ball rests or moves unobstructed. In this state, a Kalman filter can be used efficiently to track the ball and to compute its velocity.

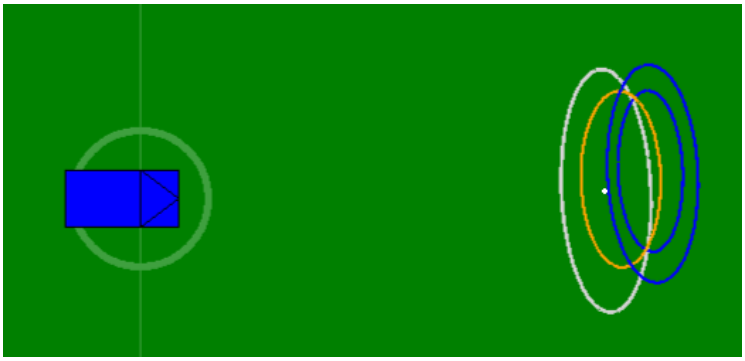

Fig. 5. A particle set. The ball has just been kicked and is seen at the white dot. The ellipses represent the position covariance by showing a (magnified) confidence region. The figure shows two particles in ball mode *none* (blue) and one in mode *manipulated* (orange). The white ellipse represents the uncertainty of the measurement.

Grabbed: The ball is grabbed by the robot, hence the ball movement is coupled to the movement of the robot.

Kicked: The robot kicks the ball. The direction and velocity of the following ball movement depends on the kick used.

Stopped: The ball hits an obstacle or stops because of its unbalanced mass or the friction of the carpet.

Manipulated: The trajectory of the ball changes suddenly, probably because another robot kicks the ball. In this case, there is very little knowledge about the ball velocity.

The transition probability between the ball modes is defined by a dynamic statistical matrix. It contains the probability that the ball behaves according to a certain mode given the previous mode and the current action of the robot.

3.4 Particle Management

The model update distinguishes two main cases, depending on whether the ball was seen in the current image or not.

In the case of a ball percept a set of new particles is generated from each existing particle. Each of them is labeled with the a-priori probability of a transition from the previous ball mode to the new one. The position and speed, initially a copy from the predecessor particle, is updated with the respective time update and the measurement update. The latter also computes the likelihood of that particle which forms the particle weight together with the a-priori label. Finally resampling is used to reduce the particles to a feasible number.

If the ball was not seen, only a reduced model is maintained. In the case of a non-linear ball movement, it is hardly possible to predict the position and speed of the ball. Therefore only the particles with the linear ball mode *none*

are updated and all other particles are discarded. However the probability of those discarded particles is accumulated to keep track of the probability actually represented by the current model. If the ball is rediscovered, a new particle is inserted at the seen position with unknown speed, and it is given the probability not covered by the current model.

Figure 5 shows an example of a set of particles.

4 Behavior Control

4.1 XABSL – Language Extensions

The behavior architecture of the GermanTeam is based on the *Extensible Agent Behavior Specification Language*(XABSL) [6]. XABSL is a formalism for the pragmatic design of agent behavior through hierarchies of finite state machines. It has been developed in the GermanTeam and is successfully applied by many RoboCup teams. In 2005 a new description language has been developed that uses a C-like syntax in order to specify XABSL state machines in a compact manner. Currently, the XABSL architecture and description language is being refined further. New features include typed and parameterized input functions and variable data types for output symbols and option parameters. Figure 6 shows an example of one of the state machines of the GermanTeam behavior specified in the new description language.

4.2 XABSL-Editor

One of the strengths of XABSL is its ability to generate a graphical documentation of the behaviors specified. However, so far this was only possible in a batch-like style, so there was no direct connection between the source file currently edited and the documentation. The *XabslEditor* (cf. Fig. 6) fills this gap. It allows navigating through the options[1] of the behavior easily. The source code can be manipulated and the resulting graph is visualized right away. This improves understanding the options and it helps to determine the context of several options. Functionality for source code editing such as syntax highlighting and code completion is also included.

4.3 Passing Behavior

The main problem when playing a pass is detecting the receiving robot. Trusting only in the positions communicated between both robots involved in a pass is not a successful approach. The inaccuracies in self-localization accumulate, so passes are never really precise.

To solve this task, players perceived visually are used to detect the receiving teammate, and the passing robot turns to the robot seen. Before the receiving robot is seen, the position communicated is used as a clue where to look for

[1] XABSL devides the behavior into separate state machines called *options*.

the receiving robot. The best results were achieved recognizing a robot when its waist jersey can be seen i. e. when the jersey area seen is as big as possible. So the pass-receiver turns to an angle of 80° to the sender to show its flank. Before the pass is executed, it turns back to receive the ball. If the pass-receiver cannot be seen, no pass is played.

4.4 Passing Challenge

If the robot playing the pass is not at its home position the pass is played anyway—even if it will not be counted. After the pass the robot walks back to its home position. It is the responsibility of the receiver of the pass to always be at its home position. So if that pass was successfully played, both robots should be back on their respective positions and at least the next pass will be counted as a correct one. Only if the ball rolls very far away from the home position it is grabbed and carried there safely. It is not advisable to play far passes because far robots can not be detected well and the risk of playing a bad pass is too high.

While the ball is carried by a robot, it is hard to detect the objects on the field. This deteriorates the self-localization. To avoid being punished for ball holding and to collect information for localization, the robot stops every three seconds during the passing challenge and lifts its head.

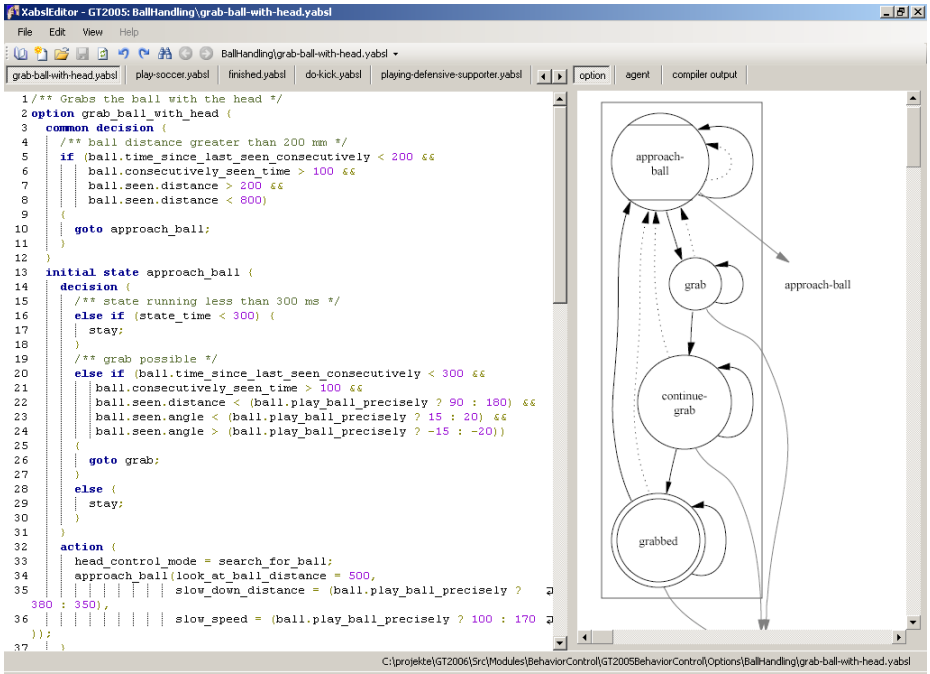

Fig. 6. Example XABSL source code for the option *"grab-ball-with-head"* opened with the XabslEditor.

5 Conclusion and Future Work

One of the main objectives of the GermanTeam is to define the state of the art in the Four-Legged League, and to catch up in areas, in which others have defined it. Playing soccer with a black and white ball is an example for the former, the Rao-Blackwellized ball model an example for the latter. Improvements in robot recognition and robot modeling allow for new styles of game play such as dodging and passing.

Another major objective of the team is the continuous improvement of the GermanTeam framework. This year, the image processing was reorganized, the behavior modeling language improved, and a graphical editor for this language was developed.

References

1. T. Röfer, "An architecture for a national robocup team," in *RoboCup 2002 Robot Soccer World Cup VI, Gal A. Kaminka, Pedro U. Lima, Raul Rojas (Eds.)*, no. 2752 in Lecture Notes in Artificial Intelligence, pp. 417–425, Springer, 2003.
2. T. Laue and T. Röfer, "Integrating Simple Unreliable Perceptions for Accurate Robot Modeling in the Four-Legged League," in *RoboCup 2006: Robot Soccer World Cup X*, Lecture Notes in Artificial Intelligence, Springer, 2007. to appear.
3. D. Göhring and J. Hoffmann, "Sensor Modeling Using Visual-Object Relation in Multi Robot Object Tracking," in *RoboCup 2006: Robot Soccer World Cup X*, Lecture Notes in Artificial Intelligence, Springer, 2007. to appear.
4. J. Hoffmann, "Proprioceptive Motion Modeling for Monte Carlo Localization," in *RoboCup 2006: Robot Soccer World Cup X*, Lecture Notes in Artificial Intelligence, Springer, 2007. to appear.
5. C. Kwok and D. Fox, "Map-based multiple model tracking of a moving object," in *RoboCup 2004: Robot World Cup VIII* (D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, eds.), no. 3276 in Lecture Notes in Artificial Intelligence, pp. 18–33, Springer, 2005.
6. M. Lötzsch, J. Bach, H.-D. Burkhard, and M. Jüngel, "Designing agent behavior with the extensible agent behavior specification language XABSL," in *RoboCup 2003: Robot Soccer World Cup VII* (D. Polani, B. Browning, A. Bonarini, and K. Yoshida, eds.), vol. 3020, Springer, 2004.