B-Smart

(Bremen Small Multi-Agent Robot Team)

Team Description for RoboCup 2007

Tim Laue, Armin Burchardt, Kai Cierpka, Sebastian Fritsch, Nils Göde, Kamil Huhn, Teodosiy Kirilov, Bianca Lassen, Markus Miezal, Eyvaz Lyatif, Malte Schwarting, Andreas Seekircher, Ruben Stein

Fachbereich 3 - Mathematik / Informatik
Universität Bremen, Postfach 330440, 28334 Bremen, Germany
bsmarter@informatik.uni-bremen.de

Abstract. This paper documents the current technical situation and upcoming changes at B-Smart's hard- and software in the near future. As an overview, existing systems and their principles will be outlined briefly, whereas most texts concentrate on planned and already completed modernizations since summer 2006 [1].

1 Introduction

B-Smart is a project for students in the advanced study period at the Universität Bremen. The team is competing in the RoboCup Small Size League, a mechanically simplified but fast and tactically challenging version of robot football. After the RoboCup world championship in 2006, the project was re-launched as a sequel to several similar named projects. B-Smart takes part in world championships since 2003 and hopes to improve its best ranking, which was reaching the quarter-finals in 2005.

2 Hardware Architecture

2.1 Mechanical Design

Resulting from the need of several reconstruction issues, we decided to rebuild the chassis and most parts of the kicking units. For that reason, everything described here is currently in development and might differ from the final version being used in the competition.

Chassis The chassis is used to hold all other components and also works as heatsink for the voltage converter. It consists of two laser-cut aluminum plates, which are connected by the four motor mounts (c. f. fig. 2). The top plate serves to hold the electronic tower, which consists of several different electronics boards. To carry the electromagnets for normal and chipped kicks, the bottom plate is caved in the middle.

The robot chassis has a diameter of $176mm$ and a height of $145mm$.

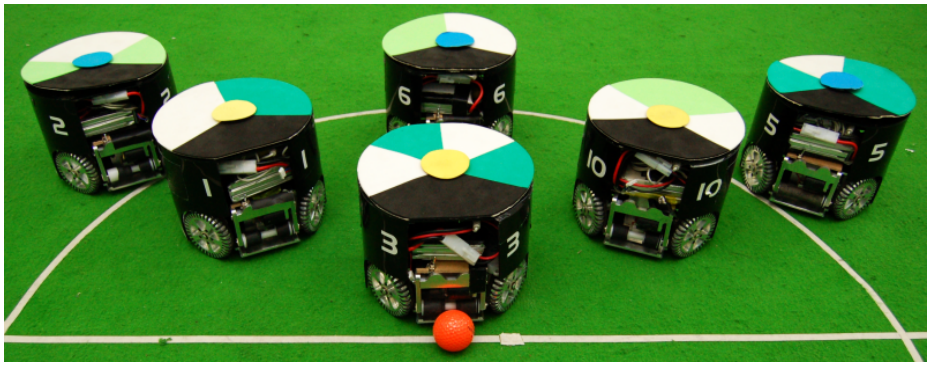

Fig. 1. The current B-Smart robots with old top covers. See 3.3 for the new ones.

Kicking system The kicking system consists of the main kicker (c. f. fig. 3), which is designed for hard forward shooting, and the chipkick, which allows the ball to be kicked over a defense line of opponent robots. Both kicking systems have their own electromagnets. The magnet of the straight kick has a maximum stroke of 17.2mm and hits the ball directly, the chipkick magnet (with a stroke of 12.3mm) has the same construction (pushing) and lifts the chipkick shovel by pushing it down on the other side like a compensator over two joints.

The mechanical changes necessary for the new chipkick and dribbler have reduced the amount of space inside. Because of this, we have replaced the coil for the straight kick with the above-mentioned electromagnets. In the course of this all the data for the chipkick's coil also had to be reviewed.

In the meantime we are experimenting with the current kicking module, the straight kick electromagnet, in order to optimise the distance between the kickinghead and ball, to find the best setup. Eventually both kicks will be executed by two capacitors with $5000\mu F$, each loaded at about $100V - 400V$, depending on how the robot has been adjustmented. Using this, the ball reaches speeds of up to 10m/s, allowing for proper gameplay but also high-speed shooting in case of a shot on goal.

Since we still want to be able to make passes, the power of the straight kick can be adjusted by the software. This is done simply by adjusting the voltage, which is induced from the

capacitor to the electromagnets, shortening the time the capacitor's voltage is induced into the electromagnet. Unfortunately, at this time we cannot make any further specifications about the chipkick because it is still in development.

Driving system The Driving system is based on four wheels with 36 subwheels mounted on each of them. The wheels are actuated by four *Faulhaber* 2342S006CR DC-Motors, powering the gearbox that consists of only two spur

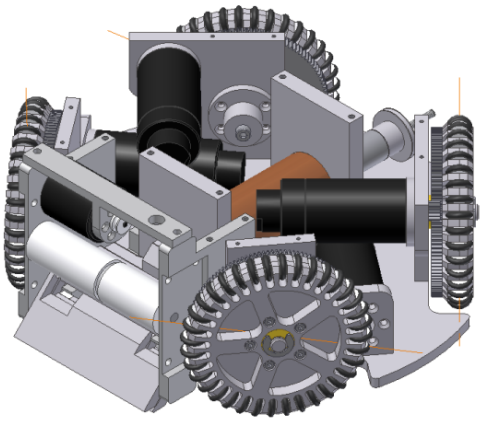

Fig. 2. 3D CAD figure of the B-Smart robot's base

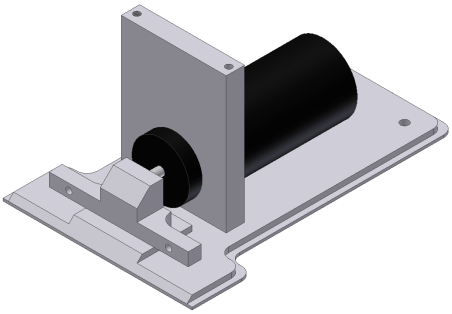

Fig. 3. 3D CAD figure of the main kicking system

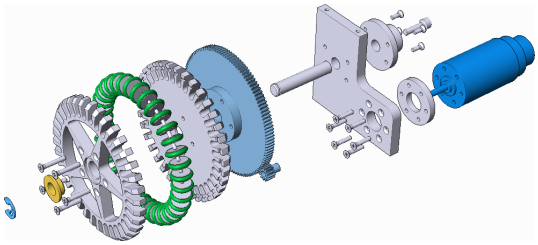

Fig. 4. 3D CAD exploded view of the driving system

wheels. These form a transmission ratio of $10:1$ achieved by a small gear with ten teeth and a big gear on the wheel-axis with 120 teeth (c. f. fig. 4).

Besides the good results, the arrangement of the wheels led to some worries. The best solution we found was an angle of 90 between the wheel axes. Unfortunately, the kicking device in the front part of the robot prevented us from using this angle. We have chosen an angle of 45 to the roll-axis of the robot for the back wheels and an another one of 53 for the front wheels. This setup leaves enough space for the kicking system.

For the sake of completeness we mention that a motor with attached encoder is about $53.7mm$ long. The two front motors are placed near the groundplate while the rear motors are located under the top chassis plate, because we were not able to put all four motors at the same level without violating the diameter rule.

Dribbling system The last system is a dribbling device (c. f. fig. 5) for the robot to handle the ball in-game. This dribbling system consists of a rubber cylinder made out of foam rubber with a grade of 15 Shore. This foam rubber was punched out in rings which were slid on the dribblers axis, which is driven by a *Faulhaber* 2224U006SR DC-Motor.

To comply with SSL rules, the dribbling system and the chassis conceal only 15 percent of the ball.

2.2 Electronic Design

The electronic design consists of four major parts: The *Foxboard* for high-level control and communication, the *Rabbitboard* for low-level motor control and sensory usage as well as the *Motorboard* for actually driving and monitoring the four attached motors via H-Bridges (VNH2SP30). The fourth part is the *Kickerboard*, which is used for for activating the shooting-mechanism.
This modular design gives us the opportunity to switch damaged devices fast, an advantage not just usable for competitions. Furthermore the error-detection is more accurate, because checking just one of four parts is faster and not as error-prone as a complete system check.

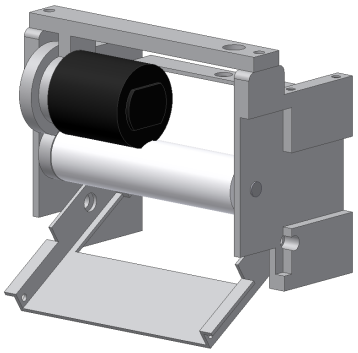

Fig. 5. 3D CAD figure of the dribbling system

For communication with the control software we use an embedded Linux device - the *Foxboard* (http://www.acmesystems.it). This development board includes a USB controller to which a USB WLAN adapter is attached.

The *Rabbitboard* is controlled by an Atmega128, which controls the speed of the four motors using a PID control loop at 100Hz, continuously monitors all important hardware states and manages the activation of the kicking device. To control the battery voltage, an Attiny13 is placed on the board, which detects the voltage and communicates via the SPI interface with the Atmega128.

To improve the kicking, we decided to integrate a light barrier. It detects, if the ball is in a position to be kicked. This is controlled by a second Attiny13 which sends an interrupt to the Atmega128.

For a better control of the wheelspin, we also integrated a gyroscope and an accelerometer. These two chips are also be connected via the SPI interface. We are just working on the software for integrating them which will grant an automatic way to adjust the PID values.

The *Rabbitboard* and the *Foxboard* communicate with each other over a serial interface.

The *Kickerboard* basically consists of two parts: A supercharger for loading the capacitor and an actuator for releasing the power via time triggered IGBTs. The capacitor is currently charged to a maximum of 100 Volts, but we are working on an improved version with higher voltages.

3 Software Architecture

3.1 Communication

We use UDP multicast sockets for communication between the system components. This allows to start the programs on different computers without changing parameters and work in small groups in the same physical net without influencing each other.

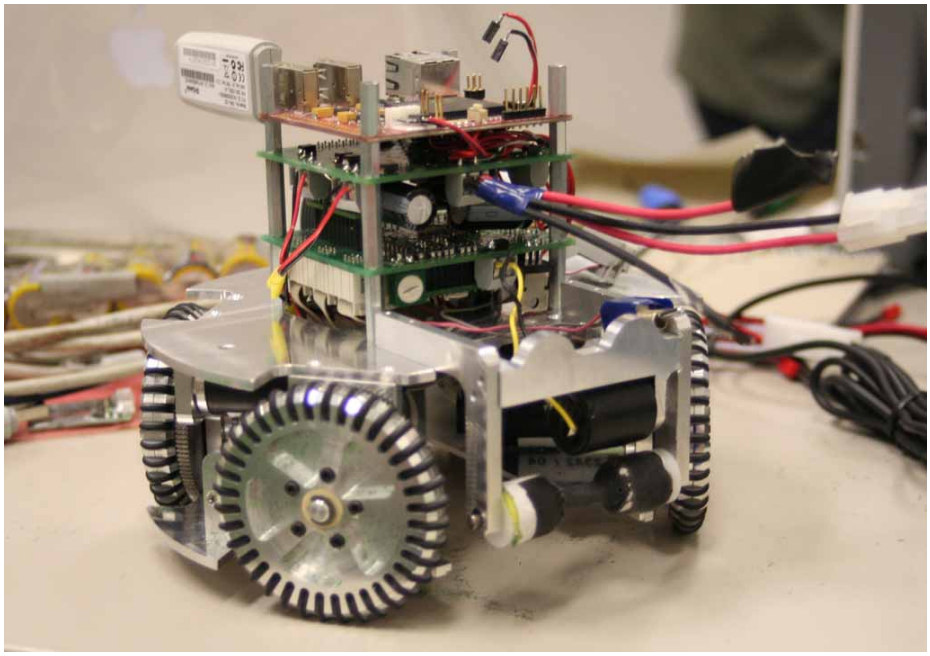

Fig. 6. A robot without cover.

The robots receive their commands via WLAN IEEE 802.11b. Current development tries to speed up the communication and explore more failure tolerant systems as the b-standard WLAN may be disturbed on RoboCup events too easily.

In the future, arbitrary information (e. g. acceleration and rotation) should be sent by the robot to adjust mechanical parameters, improve position prediction, and simplify debugging. Additionally, a webserver on the robots should host an interface to control the robots directly and manipulate robot specific parameters as every robot has its own physical behavior slightly different from others.

3.2 Behavior Control

To control the robots on the field, we use the *Extensible Agent Behavior Specification Language (XABSL)* [2] to make decisions in the B-Smart *Agent*. The potential field approach [3], which has been used during the past years, is kept since it complements the new architecture with its abilities of continuous motion planning and action evaluation. A new component is a planning system for offensive situations. Both easily integrate into the *XABSL* framework.

XABSL *XABSL* is an XML based behavior description language and can be used to describe behaviors of autonomous agents. It simplifies the process of

specifying complex behaviors and supports the design of both very reactive and long term oriented behaviors. It uses hierarchies of behavior modules that contain state machines for decision making. *XABSL* has been successfully applied in the Four-Legged league by the *GermanTeam* [4]. The usage of *XABSL* grants a good overview to the available behaviors and also offers extended debugging features.

Behaviors are specified in a C-like programming language and transformed to an intermediate code which is interpreted by the *XABSL* engine. This shortens the compile time of the system, allows more effective behavior development, and gets very handy when changes have to be applied in the halftime of a running game.

Plan Execution System In offensive situations, the standard behavior control can be interrupted by a plan execution system. This offers more tactical decision capacity during the game. Especially in conjunction with improved passing abilities through hardware enhancements (see 2.2), this system should improve the cooperative gameplay, based on *Plays as Effective Multiagent Plans Enabling Opponent-Adaptive Play Selection* [5]. The system works in two steps.

The *Strategist* is a GUI-based stand-alone tool (c. f. fig. 7) which offers the ability to define game situations and connect them with proper actions for the team. Figure 7 shows the visualization in the *Strategist*. To each area, different constraints may be added to define at what situation a specific plan should be chosen. These definitions will be saved into a XML file which can be loaded by the *Coach* component.

The *Coach* component is realized as a module in the B-Smart *Agent*. If our team is in ball possession, it searches for matching plans at runtime like a case-based reasoner. When an initial plan step's invariant is satisfied, the *Coach* "asks" a previously chosen set of *Experts* to execute the plan or not. Their opinion is based on the success of each plan in former situations.

Potential fields Artificial potential fields, originally developed by [6], are a quite popular approach in robot motion planning, because of their capability to act in continuous domains in real-time. Especially in the RoboCup domain, there also exist several applications of potential functions for the purposes of situation evaluation and action selection [7, 8]. In the B-Smart software, the movement of an agent is influenced by artificial potential fields after a robot action has been selected by the behavior control.

All motion behaviors are mostly based on the standard motion planning approach by [6]. Some of the main extensions are the integration of relative motions that allow the robot to behave in spatial relations to other objects, e. g. to organize in multi-robot formations, and the implementation of a path planner to avoid local minima.

Assigning force fields to single objects of the environment allows avoidance of obstacles and provides an approach to reach desired positions e. g. near the goal. Nevertheless, moving to more complex spatial configurations, e. g. positioning between the ball and the penalty area or lining up with several robots to build

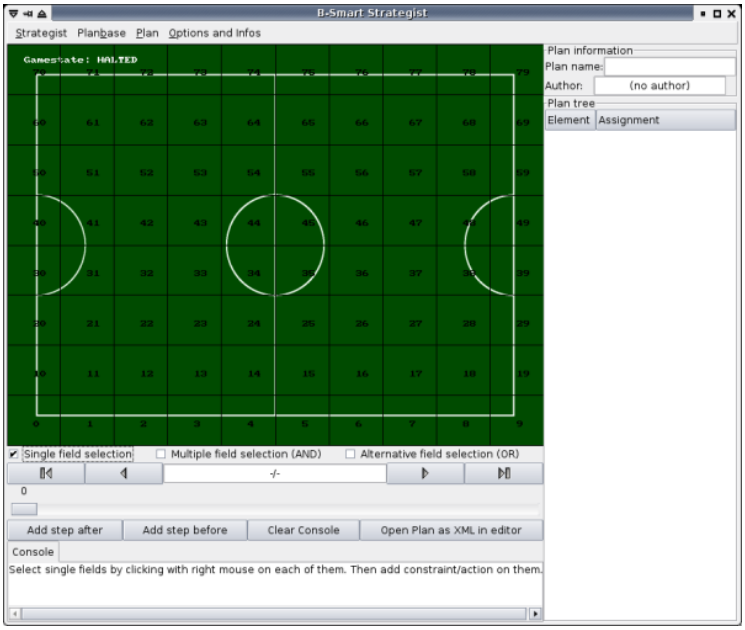

Fig. 7. GUI of the Strategist

a defense line is not directly possible. Therefore a technique, quite similar to [9], has been integrated to map complex spatial configurations to potential fields.

Due to the increasing robot speed, it will probably soon be necessary to extend this approach for a more anticipatory obstacle avoidance. For example force fields could be influenced by the movement of an object.

3.3 Vision

As almost every team in SSL, we use a global vision system. Pictures from two *AVT Marlin F046C* fire-wire cameras are combined to provide an internal world model to other software components. Therefore, simultaneous threads read, convert, process and provide the image data in a modular vision framework.

New features are currently implemented to improve the quality of the world model data. One of these will be a chipkick detection in the way the *FU-Fighters* [10, 11] team introduced it. We are currently trying to adapt this solution to improve our chances against strong teams with sophisticated chipkick modules.

Another already built feature is a more flexible module for the computation of the camera perspective. With this module we hope to reach more independence from a good camera position for the object recognition to work. Theoretically, it might now be possible to place cameras on lower angles than the standard 90 degrees without losing possibility to recognize all objects.

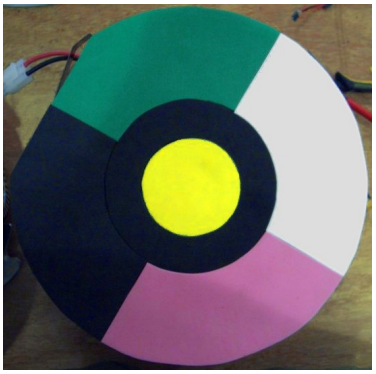

Fig. 8. Latest cover of B-Smart robots

Additionally, a shading correction was implemented that adjusts the color of the blue and yellow blobs on the robots. The results are uniform blue and yellow color values independent of the robots position on the field. This method decreases problems with dark edges and static shadows.

A new concept of top cover profiles improves the detection of the robots. The orientation as well as the identity of a specific robot is now detected more accurate and reliably. The circular cover (c. f. fig. 8) consists of four colored areas of the same size. The order of the colors (black, white, magenta, green) encodes the ID, whereas the black field is located left of the front to detect the robots orientation. Furthermore to ensure a reliable detection of the blobs center a black ring separates the four fields from the blob.

3.4 Driving Dynamics Prediction

Due to the fact that there is a significant delay between sending commands to a robot and seeing the corresponding movement on the field, prediction methods had to be implemented to improve the driving capabilities. To cope with this issue we use an approach by [12] to reduce the control delay. This is done by utilizing multi-layer feed-forward networks for prediction. After recording an example data set with typical driving commands, the network can be learned using the RPROP algorithm [13].

3.5 Simulation

Our old simulator - based on the Open Dynamics Engine - simulates the robot as one rigid body using ODE motors controlling the speed in world coordinates. This gives good performance but ignores several effects, e. g. friction between subwheels and ground.

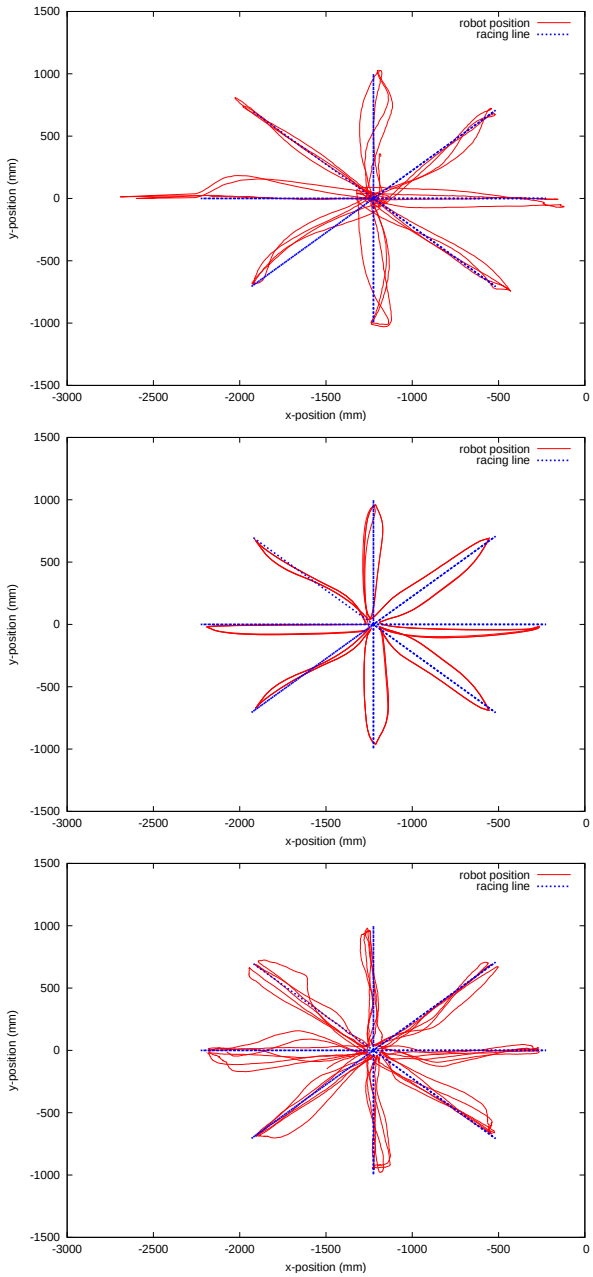

Fig. 9. Comparison between a real robot (top), the ODE-Simulator (middle) and the new Neuralsim (bottom). Task in all three measurements was to drive a star shaped figure whereas the edges are arranged in 45 degree angles. You can see the far more realistic simulation with the new Simulator compared to the old one.

Being aware of the significant differences between the simulated movements of our robots and the behavior in reality, a new simulator based on neural networks has been developped (following the idea of [14]).

The neural networks used for the simulation are created from a logfile containing the world model and the commands sent to the robot for each frame. The logfile should contain records of a robot moving in all directions at different speeds. Using this log data, we train the network.

Additionally to the more realistic behavior, we could also bridge the delay from the captured picture to the robot with this simulation. This was ignored within the old ODE-based simulator. This is because the last 10 driving commands and robot parameters are given as input to the networks.

Of course this method is not a perfect solution and there are still problems to solve, especially as the quality of the simulation is significantly influenced by the provided log data.

A comparison of the driving behavior in the two simulators and on a real robot is shown in fig. 9.

B-Smart team members

Christoph Budelmann, Armin Burchardt, Kai Cierpka, *Ubbo Visser*, Sebastian Fritsch, Nils Göde, Sven Hinz, Kamil Huhn, Teodosiy Kirilov, Bianca Lassen, *Tim Laue*, Eyvaz Lyatif, Alexander Martens, Marc Michael, Markus Miezal, Markus Modzelewski, Ulfert Nehmiz, Florian Penquitt, Denis Pingin, *Thomas Röfer*, Sebastian Schleusener, Malte Schwarting, Andreas Seekircher, Ruben Stein

References

1. Birbach, O., Burchardt, A., Elfers, C., Laue, T., Penquitt, F., Schindler, T., Stoye, K.: B-Smart - Team Description for RoboCup 2006. In Lakemeyer, G., Sklar, E., Sorrenti, D., Takahashi, T., eds.: RoboCup 2006: Robot Soccer World Cup X Preproceedings, RoboCup Federation (2006)
2. Lötzsch, M., Risler, M., Jüngel, M.: XABSL - A Pragmatic Approach to Behavior Engineering. In: Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS), Beijing, China (2006) 5124–5129
3. Laue, T., Röfer, T.: A Behavior Architecture for Autonomous Mobile Robots Based on Potential Fields. In: 8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
4. Röfer, T., Brunn, R., Czarnetzki, S., Dassler, M., Hebbel, M., Jüngel, M., Kerkhof, T., Nistico, W., Oberlies, T., Rohde, C., Spranger, M., Zarges, C.: GermanTeam 2005. In: RoboCup 2005: Robot Soccer World Cup IX. Lecture Notes in Artificial Intelligence (2005)
5. Bowling, M., B., B., Veloso, M.: Plays as Effective Multiagent Plans Enabling Opponent-Adaptive Play Selection, American Association for Artificial Intelligence (2004)
6. Khatib, O.: Real-time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research **5** (1986) 90–98
7. Johannson, S.J., Saffiotti, A.: Using the Electric Field Approach in the RoboCup Domain. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: RoboCup 2001: Robot Soccer World Cup V. Volume 2377 of Lecture Notes in Artificial Intelligence., Springer (2002)
8. Meyer, J., Adolph, R.: Decision-making and Tactical Behavior with Potential Fields. In Kaminka, G.A., Lima, P., Rojas, R., eds.: RoboCup 2002: Robot Soccer World Cup VI. Volume 2752 of Lecture Notes in Artificial Intelligence., Springer (2003)
9. Balch, T., Arkin, R.C.: Behavior-based Formation Control for Multi-robot Teams. IEEE Transactions on Robotics and Automation **14** (1999) 926–939
10. Simon, M.: Ein robustes Echtzeit-Vision-System für die Robocup F180 SmallSize Liga. Diploma thesis, Freie Universität Berlin (2006)
11. Rojas, R., Simon, M., Tenchio, O.: Parabolic flight reconstruction from multiple images from a single camera in general position. (2005)
12. Behnke, S., Egorova, A., Gloye, A., Rojas, R., Simon, M.: Predicting away the delay. In: Proceedings of 7th RoboCup International Symposium. (2003)
13. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA (1993) 586–591
14. Gloye, A.: Lernmethoden für autonome mobile Roboter (2005)