B-Smart

(Bremen Small Multi-Agent Robot Team)

Team Description for RoboCup 2006

Tim Laue, Torben Schindler, Florian Penquitt, Armin Burchardt, Oliver Birbach, Carsten Elfers, Kai Stoye

Center for Computing Technology (TZI), Faculty 3 Mathematics/Computer Science,
University of Bremen, Postfach 330440, D-28334 Bremen, Germany
`grp-bsmart@informatik.uni-bremen.de`

Abstract. This paper describes the current state of development and future plans of the B-Smart team since RoboCup 2005. Amongst other points, it will outline the existing B-Smart robot platform, introduce the currently constructed new robot generation prototype with all its enhancements and will also give an overview about new software components used for behaviour control, image processing, simulation, strategy and robot motion correction.

1 Introduction

The B-Smart team is a RoboCup project at the University of Bremen that currently consists of 18 student members (listed at the end of this document). The overall goal of this project is to build a team of robots that can compete in RoboCup tournaments. So far, B-Smart has already participated in the German Open and the RoboCup world championships in the years 2003, 2004 and 2005 and achieved its best result at the RoboCup 2005 in Osaka by reaching the quarter finals. A remaining big goal for the B-Smart team is a good result at the RoboCup 2006 in Bremen.

2 Hardware Architecture

2.1 Mechanical Design

2005 Since 2004, a robot platform similar to the *FU-Fighters'* design of 2003 [1], which is capable of omni-directional motion, has been used. There were several requirements for a robust platform and it was decided to use some approaches of the well known RoboCup teams as examples.

First of all, an omni-directional drive, which is capable of fast moving and easy control, was a necessity. Linear speed of about $2.0 - 2.5\ m/sec$ and an

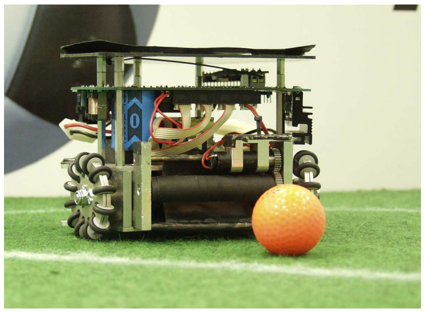
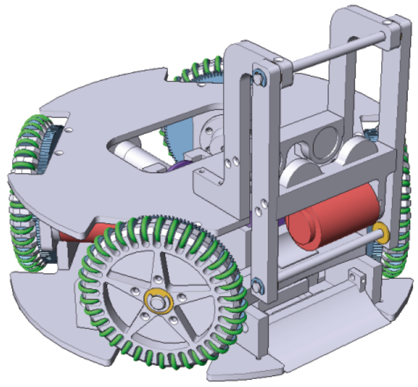



Fig. 1. Current B-Smart robot platform (left) and the new generation (right)

accordant acceleration are common among the successful teams in RoboCup Small Size competitions. Omni-directional motion can be achieved with three or four wheels which are arranged in a triangle or rectangle. Several variations are also possible. The B-Smart robots have three wheels in an arrangement which is optimised for straight forward motion, since the wheels are not aligned in uniform angles. Using this design, it is possible to achieve high speed in forward directions while moving sideways is a bit less precise and slower, but still reasonable in comparison to robots with a differential drive. With a wheel-diameter of about 54 mm, a 9.1:1 gear, an estimated motor speed of 6500 rpm and an angle of the front wheels of 150° a theoretical top speed of about 2.02 m/s is possible. Unlike several other teams it was decided to build appropriate proprietary wheels, which are again similar to the *FU-Fighters*' approach. The advantage is a higher friction on the field. The usage of *Faulhaber* DC-motors is a reasonable choice of most teams. These motors are equipped with 9.1:1 planetary gears.

For kicking the ball, a solenoid based kicking device has been used. Since a kicker can only be effective when the ball is clinging to the robot, a dribbling device also had to be integrated to hold the ball at a defined position.

The chassis is built out of laser cut aluminium plates and integrates pockets for the accumulator batteries and other hardware components. This design is very lightweight and easy to manufacture.

2006 Our next generation of robots needed a lot of new features and improvements. First of all, we wanted to use four wheels instead of three, with the triple of subwheels on each wheel. The previous robot generation was not able to kick the ball with the necessary speed, so we decided that a big change would be required to compete in RoboCup 2006. The new robot will have a diameter of 178mm and will be 145mm in height.

The new robot platform consists of four assembly groups: the chassis, the kicking system, the driving system and the dribbling system.

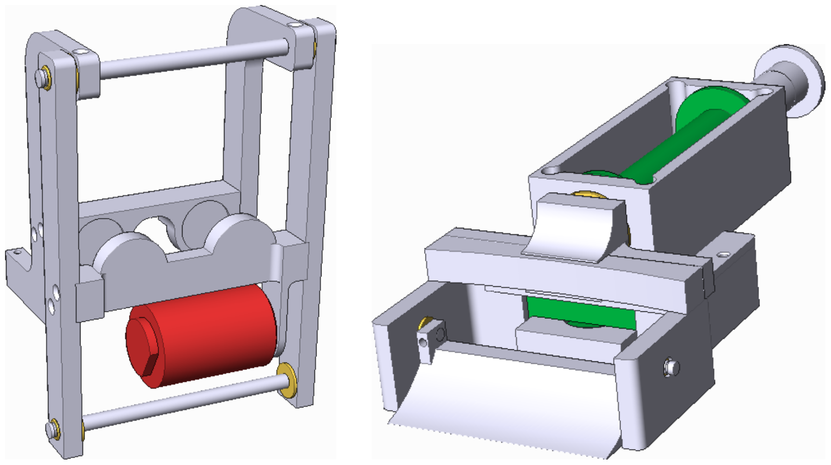


Fig. 2. 3D studies of the new dribbling system (left) and the new kicking device with straight- and chipkick (right)

The chassis is used to hold all the other components and also works as heatsink for some MOSFETs of the kicking circuit. It consists of two laser-cut aluminium plates, which are connected by the four motor mounts. The bottom plate has an open area to carry the kicking system. The new kicking system consists of the main kicker, which is designed for hard forward shooting, and the chipkick, which allows to kick the ball e. g. over a defense line of opposing robots.

Each kicking system (straight-kick and chipkick) has its own solenoid and works in almost the same way: the solenoid pushing a steel rod. While the straight kick hits the ball directly, the chipkick core hits the chipkick rocker, which then lifts the ball. Both kicking systems are still under development and testing. We are experimenting with the current and the distance of the kicking-head to the ball to find the best setup for optimal results. Currently we plan to use a voltage of 100V and two capacitors with $5000\mu F$ each.

The ball speed is up to about 10m/s with the current setup. This allows proper gameplay but can still be increased a little more for high-speed shooting at the opponent goal. Since we still want to be able to play passes, the power of the straight forward kick can be adjusted by the software. This is done by simply shortening the time the capacitor's voltage is induced into the solenoid.

The main improvement we made to the driving system is the use of four wheels, with 36 subwheels each. The wheels are actuated by four *Faulhaber* 2342S006CR DC-Motors. The gearbox consists of only two spur wheels. The transmission ratio of 10:1 is achieved by a small spur wheel with ten teeth and a big one on the wheel-axis with 120 teeth. This allows a wheel idle speed of about 900rpm. The arrangement of the wheels gave us some problems. The best

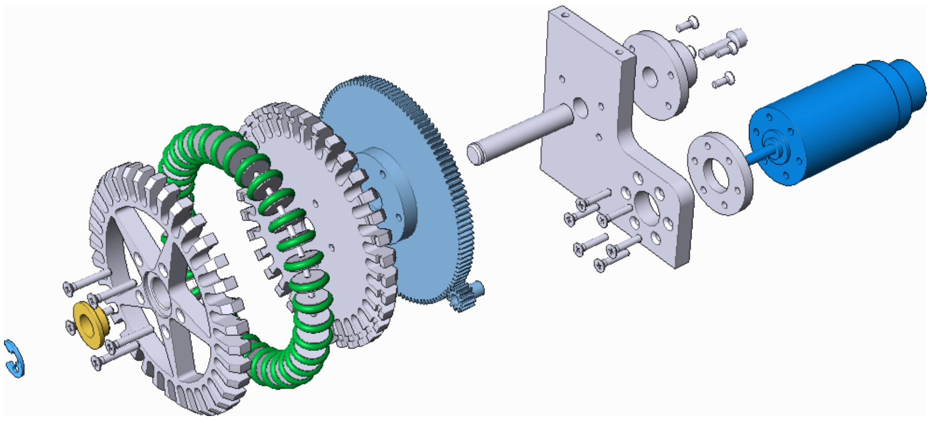


Fig. 3. Exploded view of the driving system

solution we found was an angle of 90° between the wheel axes. Because of the kicking device in the front part of the robot this angle was not practicable. We chose an angle of 45° to the roll-axis of the robot for the back wheels and an angle of 53° for the front wheels. This setup leaves enough space for the kicking system.

A motor with attached encoder is about $53,7mm$ long. The two front motors are placed near on the groundplate while the rear motors are located near under the top chassis plate, because we were not able to fit all four motors in the same level without violating the diameter rule. To control the ball in the game, the robot should again use a dribbling device. Simplified this consists of a swing frame and a rubber roll. The rubber or dribbling roll is driven by a *Faulhaber* 2224U006SR DC-Motor. The rotation of the roll gives the ball a backspin and moves it toward the robot. As this is standard in the SSL, no further description should be necessary.

The swing frame is elastically supported, which allows a better ball control and also decreases the number of rebounds when accepting a pass or blocking an opponent shot. The underlying design of the dribbling system is a modified version of *Cornell Big Red's* 2003 design. We favoured this design and customised it to match our demands. To match the SSL rules, the dribbling system and the chassis conceal only 15 percent of the ball.

2.2 Electrical Design

A microcontroller controls the speed of the four driving motors using a PID control loop. Radio commands are received using a Radiometrix RPC. To solve some issues with our current robots we made a few changes to the electrical design:

– A Philips LPC2129 (32 Bit, 60 MHz) instead of the Fujitsu MB90F594A Controller (16 Bit, 16 MHz) now allows better integration into our Linux

based environment as the GNU *gcc* compiler can produce binary files for this controller.
– Last years robots are using software interrupts to measure the motor speeds. As the Fujitsu controller is too slow to integrate the signals from the motor encoders in interrupt handlers we are using special hardware to solve this issue. Dedicated quadrature decoders (LS7366) with integrated counters and a serial interface will handle the input from the motor encoders, ensuring a better resolution for speed measurement without wasting CPU processing power.
– Our DC motors require stronger H-Bridges as the old ones, which were even too small for the old robot and required a large heatsink.
– The modified DC/DC-Converter supports higher voltages (up to 100V) for the kicker.

3 Software Architecture

The whole control software splits up in stand-alone components, which communicate over UDP network sockets. There is a component for each different task. This includes vision, building a representation of the world, agent control, radio transmission, strategy planning and logging of various transmitted data.

3.1 Vision

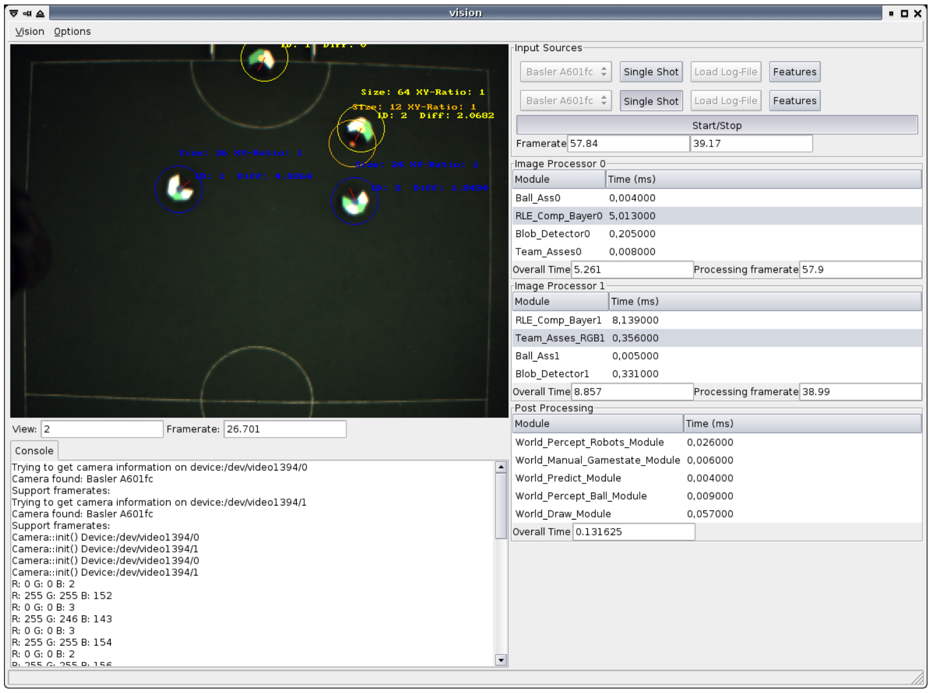
Due to the field size, the usage of more than one camera is required to do appropriate observations on the field. Two Basler A601fc fire-wire cameras are used by B-Smart. The percepts from the two different points of view are combined to provide an integrative world model, which is then distributed to the agents.

The whole image processing framework has been rewritten completely, shortly before RoboCup 2005. This has been necessary because of severe problems concerning the maintainability of the application and synchronization problems between our previously distributed vision processes. The current application follows a modular concept for image processing and the generation of representation for for behavior control. Single tasks, e. g. segmentation, robot detection, or the computation of the ball velocity, are implemented in small modules which may be combined freely. This allows an easy integration of new approaches.

Additionally, several debugging facilities have been built-in. Among others, an abstract concept for debug drawings facilitates the visualization of information from the modules.

3.2 Behavior Control

To control the robots on the field, we have adapted the *Extensible Agent Behavior Specification Language (XABSL)* [2, 3] to our demands and integrated it into a new version of the B-Smart *Agent*. The potential field approach [4] which



Framerate 57.84 39.17

Image Processor 0

Module	Time (ms)
Ball_Ass0	0,004000
RLE_Comp_Bayer0	5,013000
Blob_Detector0	0,205000
Team_Asses0	0,008000

Overall Time 5.261 Processing framerate 57.9

Image Processor 1

Module	Time (ms)
RLE_Comp_Bayer1	8,139000
Team_Asses_RGB1	0,356000
Ball_Ass1	0,005000
Blob_Detector1	0,331000

Overall Time 8.857 Processing framerate 38.99

Post Processing

Module	Time (ms)
World_Percept_Robots_Module	0,026000
World_Manual_Gamestate_Module	0,006000
World_Predict_Module	0,004000
World_Percept_Ball_Module	0,009000
World_Draw_Module	0,057000

Overall Time 0.131625

View: 2 Framerate: 26.701

```
Trying to get camera information on device:/dev/video1394/0
Camera found: Basler A601fc
Support framerates:
Trying to get camera information on device:/dev/video1394/1
Camera found: Basler A601fc
Support framerates:
Camera::init() Device:/dev/video1394/0
Camera::init() Device:/dev/video1394/1
Camera::init() Device:/dev/video1394/0
Camera::init() Device:/dev/video1394/1
R: 0 G: 0 B: 2
R: 255 G: 255 B: 152
R: 0 G: 0 B: 3
R: 255 G: 246 B: 143
R: 0 G: 0 B: 3
R: 255 G: 255 B: 154
R: 0 G: 0 B: 2
R: 255 G: 255 B: 156
```

Fig. 4. The image processing application.

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. It easily integrates into the *Extensible Agent Behavior Specification Language (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* [5]. The usage of *XABSL* grants a good overview to the available behaviors and also offers extended debugging features.

B-Smart is also already using the *YABSL* extension to specify behaviors. Behaviors had to be originally written in XML to work with the *XABSL* engine. Since this took a lot of time and was not very intuitive, the *GermanTeam* developed the *YABSL* extension [5], which allows to specify the behaviors in a C-like programming language. Behaviors are parsed and transfered to intermediate code, that is interpreted by the *XABSL* engine. This shortens the compile time of the system, which allows more effective behavior development and can

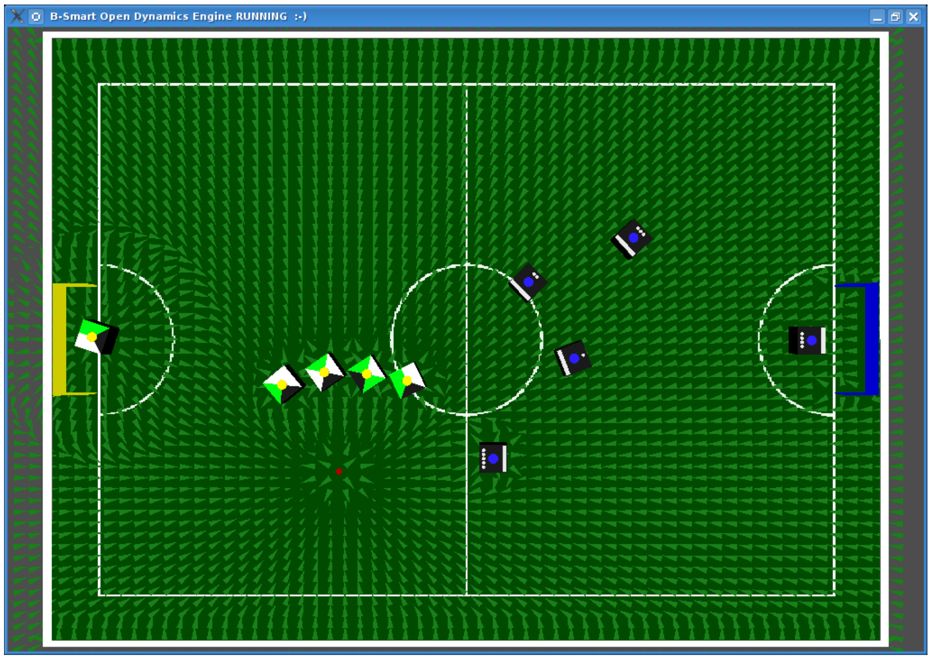


Fig. 5. A potential field for moving during the *STOP* state.

be very handy when changes have to be applied in the halftime of an actual game etc.

The existing behaviors are currently being ported to the new *Agent* and have already successfully been tested with our simulator.

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].

The approach used by B-Smart [4] combines several existing approaches inside a behavior-based architecture by realising single competing behaviors as potential fields. Such behaviors are e. g. *Move to ball*, *Move to defence position* or *Kick to goal*. The architecture has generic interfaces allowing its application on different platforms for a variety of tasks., e. g. it has already been used by the *GermanTeam* in the Four-legged Robot League.

All motion behaviors are mostly based on the standard motion planning approach by [6]. Some of the main extensions have been 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. Figure 5 shows a potential field in the Small Size domain.

Assigning force fields to single objects of the environment allows the avoidance of obstacles and the approach to desired goal positions. 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 a defence line is not possible directly. Therefore, a technique, quite similar to [9], has been integrated to map complex spatial configurations to potential fields.

The evaluation of actions and situations is based on the potential functions assigned to the objects in the environment. But instead of using a rasterisation with a large number of evaluated cells [8], a variant of the *Electric Field Approach* [7] has been implemented, as it is computationally more efficient and allows a direct evaluation of positions or actions like kicks or passes to team-mates.

To use this method to evaluate a certain action changing the environment, e. g. kicking a ball to a teammate, this action has to be mapped to a geometric transformation describing the motion of the manipulated object. A set of different transformations, inter alia including rotation, translation, and tracing the potential field gradient, has been implemented, together with mechanisms to check for collisions and practicability of the action, external mechanisms as planners are not needed. To describe more complex actions, e. g. turning with a ball and subsequently kicking to the goal or dribbling away from an opponent, also sequences of actions may be specified.

Plan Execution System A new optional component of the B-Smart software is a plan execution system for offensive situations. This offers more tactical decision capacity during the game. The implemented approach was inspired by [10]. It is divided into two different modules, the *Strategist* which is a tool to build plans that could be executed by the second module, the *Coach*.

The *Strategist* is a GUI-based stand-alone tool which offers the ability to define game situations and connect them with proper actions for the team. Therefore the field is divided into discrete areas to get an abstract representation for the robots‘ and the ball‘s position. To each area, different constraints may be added to define at what situation a specific plan should be chosen. If more than one situation should lead to a single plan, the user is able to define different invariants for the plan. Once these preconditions are defined, they can be assigned to actions. In the following, we refer to these assignments as plan steps. Every plan may contain different plan steps depending on the previously defined plan steps. 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.* Once at least one planbase (a file containing different plans) is loaded, the *Coach* is set to be active. 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 invariants are satisfied, the *Coach* “asks“a previously chosen set of *Experts* to execute the plan or not. If more than one plan is possible in the given situation the *Coach* determines a probability for execution for each plan based on the *Experts*‘ “opinions“.

If a plan is chosen to be executed, the standard behaviors of the involved agents will become overridden with the actions defined in that plan. To get dynamical, rational decisions from the *Experts*, they track the game and try to make useful conclusions for other plans, e. g. for passing the ball to other robots. The plan will be interrupted when losing the ball, shooting a goal or if the plan‘s goal can not be achieved anymore.

3.3 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. For this we use the approach by [11] to reduce the control delay.

To determine the delay, we use a behavior in which the robot rotates with a sinusoidal-like speed function. While rotating, the command sent to the robot and the actual rotation on the field of each frame are saved and plotted. The gap between the change of sign of the rotation speed and the point where the rotation changes the direction is the delay of our system. With the 2005 robot and vision system, a delay of about 5.5 frames (180 ms) had been measured.

The elimination of the delay is done by using multi-layer feed-fowards-networks for prediction. The example data set needed for the learning is obtained by driving on the field with a custom behavior sending typical driving commands to the robot. After recording all necessary data the preprocessing is done as proposed and the network is learned using the RPROP algorithm [12].

Because the new robot generation is currently not yet ready for testing and the vision system is undergoing some hardware changes (new cameras) no evaluation of the implemented prediction in our system can currently be done. But tests with earlier recorded data form the previous generation of robots show optimistic results which be can directly adapted to the new hardware later.

3.4 Simulation

To support the development of robot behaviours a minimalistic simulation based on the *Open Dynamics Engine* [13] has been developed. Thanks to the distributed design of the control software, the current robot commands are available to the simulation by listening to network packets. The positions and speeds of the robots and the ball are sent back to the control software. ODE is doing the collision detection and the integration of forces. Controllers for the speed of the objects are also provided by the library. Robots are represented as simple boxes. This allows dribbling the ball with one edge. Caused by this simple representation, the fine-tuning of the behaviors has to be done by using the real robots.

As the simulation allows dragging robots around the field, a visualisation method for the potential fields has also been included. This provides an easy way to check the potential field results for different robot positions.

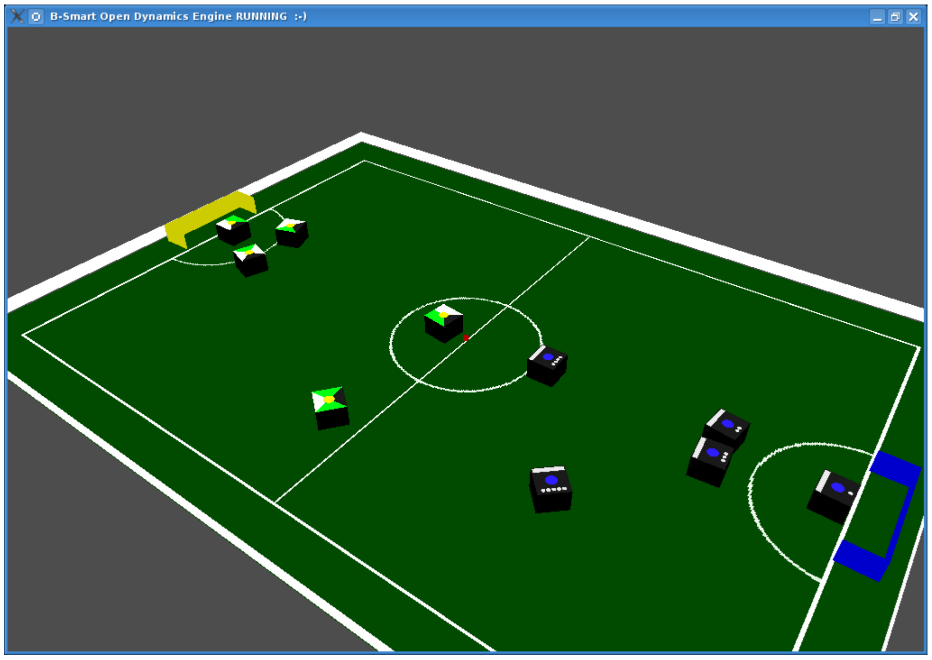


Fig. 6. Screenshot of the simulator.

4 Conclusion and Outlook

The Dutch Open 2006 will be an important event to test the new software components and the new robot generation prototype under real tournament conditions. The vision system needs to be optimized to properly work with the new cameras. Our existing behaviours are just being ported to the new XABSL based agent control system. The cooperation of the agent control system and the *Coach* component also needs to be tested and optimized in actual games.

As a backup solution and also for later testing purposes, we are also planning to upgrade our old robot generation with the new generation of electronics. This would enable the current robot generation to move more accurately and also give us the possibility to practise 5 vs. 5 situations.

Experiences and conclusions from games at the Dutch Open 2006 shall be the final touch to hard- and software before attending RoboCup 2006. We strongly hope that a complete team of the third robot generation will be completed until summer, so that we can compete with completely new robots and improved software right here in Bremen.

B-Smart team members

Oliver Birbach, Armin Burchardt, Shiyi Chen, Carsten Elfers, Ralph Freudrich, Sascha Gerl, Sandro Knauß *Jörg Kurlbaum*, *Tim Laue*, Torsten Möllenbeck, Florian Penquitt, Christian Peper, *Thomas Röfer*, Torben Schindler, Sebastian Schleusener, Kai Stoye, *Ubbo Visser*, Marian Weirich, Haijing Wei, Jie Xu and Jun Zhang

References

1. Egorova, A., Gloye, A., Liers, A., Rojas, R., Schreiber, M., Simon, M., Tenchio, O., Wiesel, F.: FU-Fighters 2003 (Global Vision). In Browning, B., Polani, D., Bonarini, A., Yoshida, K., eds.: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
2. Lötzsch, M., Bach, J., Burkhard, H.D., Jngel, M.: Designing Agent Behavior with the Extensible Agent Behavior Specification Language XABSL. In Browning, B., Polani, D., Bonarini, A., Yoshida, K., eds.: 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences). Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
3. Lötzsch, M.: XABSL-Homepage (2006) http://www.informatik.hu-berlin.de/ki/XABSL/index.html.
4. 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.
5. 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)
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. Bowling, M., B., B., Veloso, M.: Plays as Effective Multiagent Plans Enabling Opponent-Adaptive Play Selection. American Association for Artificial Intelligence (2004)
11. Behnke, S., Egorova, A., Gloye, A., Rojas, R., Simon, M.: Predicting away the delay. In: Proceedings of 7th RoboCup International Symposium. (2003)
12. 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
13. Smith, L.: Open Dynamics Engine (2004) http://www.ode.org/.