BreDoBrothers

Team Description for RoboCup 2006

Thomas Röfer[1], Martin Fritsche[3], Matthias Hebbel[2], Thomas Kindler[2], Tim Laue[3], Cord Niehaus[3], Walter Nistico[2], and Philippe Schober[3]

[1] Deutsches Forschungsinstitut für Künstliche Intelligenz GmbH, Sichere Kognitive Systeme, Robert-Hooke-Str. 5, 28359 Bremen, Germany
E-Mail: `Thomas.Roefer@dfki.de`

[2] Robotics Research Institute, Dortmund University, Otto-Hahn-Str. 8, 44227 Dortmund, Germany.
E-Mail: {*forename.surname*}`@uni-dortmund.de`

[3] Faculty 3 - Mathematics / Computer Science, Universität Bremen, Postfach 330 440, 28334 Bremen, Germany.
E-Mail: {`fritsche,timlaue,cniehaus,harlekin`}`@tzi.de`

1 Introduction

The BreDoBrothers are a joint team of researchers and students from the Universität Bremen and the Universität Dortmund. Members of both teams have a common track record in the GermanTeam [1–4], which has become world champion in the Four-Legged League twice.

The overall approach of the team is to transfer as much code and experiences from the Four-Legged League to the Humanoid League as possible. Thus the implementation of the general architecture (cf. Sect. 3) and the simulation environment (cf. Sect. 4) as well as several algorithms (cf. Sect. 7) have been adopted unmodified for the most parts.

The BreDoBrothers team consists of computer scientists and thus is not primarily interested in robot construction. For having a stable robot platform, standard robot construction kits are used and modified (cf. Sect. 2) for accomplishing the requirements of robot soccer.

2 The Robots

The commercially available KHR-1 robot kit from Kondo is used as a basis for our robots, since this kit has already been used in RoboCup competitions by different other teams. Due to some shortcomings concerning the reliability and the processing speed of the original controller board, we designed a new board using an Atmel ATMega128 controller.

For all on-board computations, we equipped the robots with standard PDAs from Dell and Fujitsu Siemens respectively. This widespread approach has been described comprehensively by [5].

All technical details about our robots are documented in a seperate robot questionnaire. Figure 1 shows two of our robots.

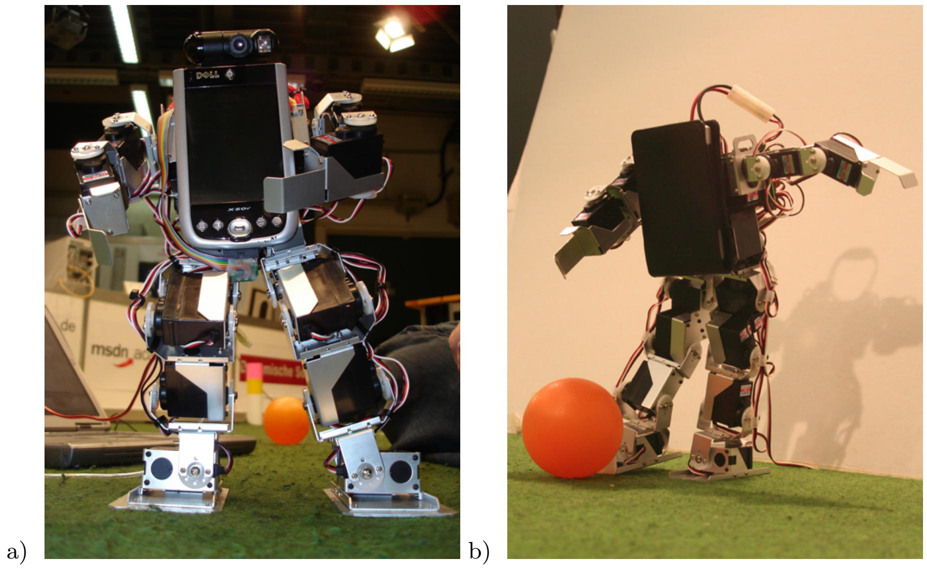


Fig. 1. The two Kondo KHR-1 based robots from a) Dortmund and b) Bremen (not having a camera and a mirror yet).

3 Software Framework

The control software of the BreDoBrothers is based on the software framework of the GermanTeam [4]. It is possibly the architecture [6] used most often for controlling real robots in RoboCup. Currently, more than 25% of all teams in the Four-Legged League base their own code on this framework and its tools. In addition, the behavior description language XABSL [18] that is part of the framework has already been used in different leagues such as the Middle-Size League (by the team *COPS Stuttgart*), and the Small-Size League (by *B-Smart*).

The BreDoBrothers use PDAs running Microsoft Windows Mobile to control their robots. Thereby this platform is the forth one supported by the framework besides the Sony Aibo, Sony's Open-R Emulator running under Cygwin, and Microsoft Windows (under the Simulator SimRobot, cf. Sect. 4). The framework running under the desktop version of Windows can also directly control the Kondo robots using a serial cable, and is able to record and replay log files.

3.1 Development Support

One of the basic ideas of the architecture is that multiple solutions exist for a single task, and that developers can switch between them at runtime. In addition, it is possible to include additional switches (*debug requests*) into the code that can also be triggered at runtime. These switches work similar to C++ preprocessor directives for conditional compilation, but they can be toggled at runtime. A special infrastructure called *message queues* is employed to transmit requests to

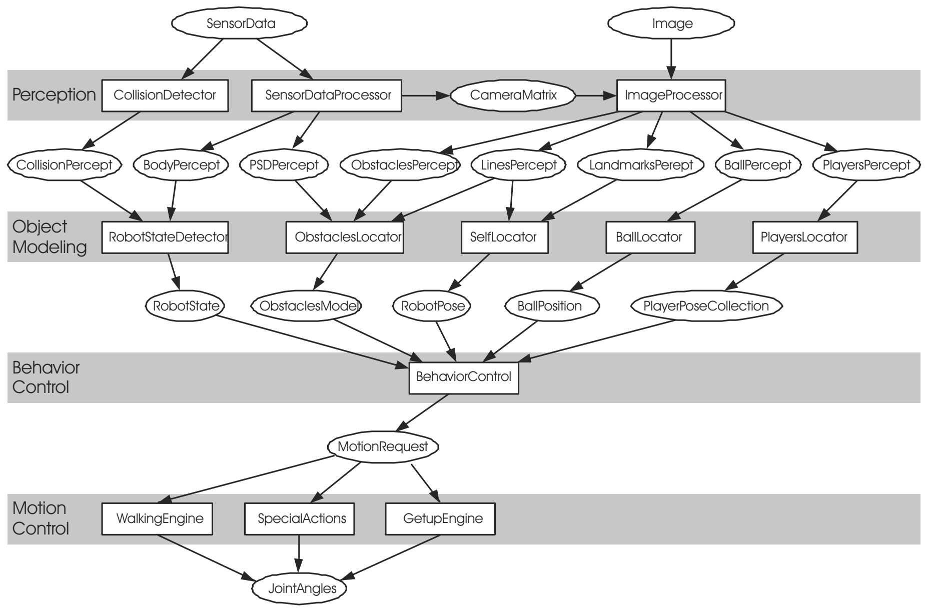


Fig. 2. Tasks and representations used for playing robot soccer.

all processes on a robot to change this information at runtime, i. e. to activate and to deactivate debug requests and to switch between different solutions. The message queues are also used to transmit other kinds of data between the robots and the graphical front-end on the PC (cf. Sect. 4). For example, motion requests can directly be sent to the robot, images, text messages, and even drawings (2-D and 3-D) can be sent to the PC. This allows for visualizing the state of a certain module, textually and even graphically. These techniques work both on the real robots and on the simulated ones (cf. Sect. 4).

3.2 Tasks

Figure 2 depicts the tasks and representations enabling the BreDoBrothers to play soccer. They can be structured into four levels:

Perception. On this level, the current states of the joints are analyzed to determine the position of the camera. The camera image is searched for objects that are known to exist on the field, i. e. landmarks (goals and flags), field lines, other players, the ball, and general obstacles such as the referees. The sensor readings that were associated to objects are called *percepts*. In addition, further sensors can be employed to determine whether the robot has been picked up, or whether it fell down.

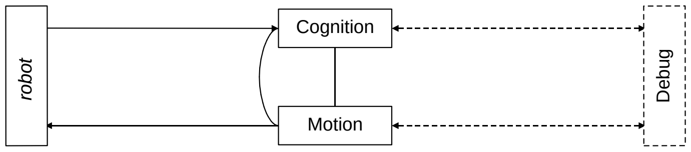


Fig. 3. The processes running on the PDA

Object Modeling. Percepts immediately result from the current sensor readings. However, most objects are not continuously visible, and noise in the sensor readings may even result in a misrecognition of an object. Therefore, the positions of the dynamic objects on the field have to be modeled, i. e. the location of the robot itself, the poses of the other robots, the positions of further obstacles, and the position of the ball. The result of this level is the estimated *world state*.

Behavior Control. Based on the world state, the role of the robot, and the current score, the third level generates the behavior of the robot. This can either be performed very reactively, or deliberative components may be involved. The behavior level sends requests to the fourth level to perform the selected motions.

Motion Control. The final level performs the motions requested by the behavior level, i. e. walking, standing up, or performing so-called special actions (kicks, cheering moves, demo motions). The motion module also performs dead reckoning and provides this information to other modules.

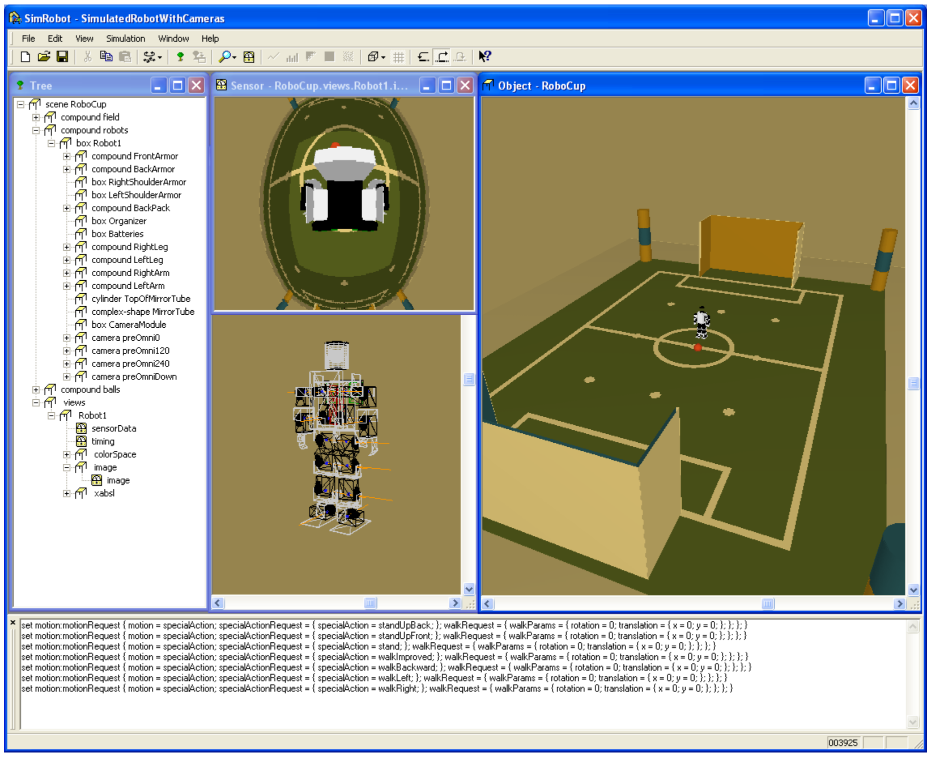
3.3 Processes

Dividing the whole problem of playing soccer in smaller tasks and grouping them together to the levels shown in Fig. 2 does not define which of the modules solving these tasks are running sequentially and which are running in parallel. A well-established approach (cf. Fig. 3) is to have one process running at video frame rate (*Cognition*) executing all modules of the first three levels, and another one running at the frequency required for sending the motion commands (*Motion*) executing the modules of the fourth level. A third process distributes and collects debugging information and communicates them with an off-board PC. This process is only used during software development and is inactive during actual RoboCup games.

Processes communicate through a fixed communication mechanism with each other (*senders/receivers*) that does not involve any queuing, because the processes should always work on the most actual data packages, skipping older ones.

4 Robot Simulation

When working with robots, the usage of a simulation is often of significant importance. On the one hand, it enables the evaluation of different alternatives



```
set motion:motionRequest { motion = specialAction; specialActionRequest = { specialAction = standUpBack; }; walkRequest = { walkParams = { rotation = 0; translation = { x = 0; y = 0; }; }; }; }
set motion:motionRequest { motion = specialAction; specialActionRequest = { specialAction = standUpFront; }; walkRequest = { walkParams = { rotation = 0; translation = { x = 0; y = 0; }; }; }; }
set motion:motionRequest { motion = specialAction; specialActionRequest = { specialAction = stand; }; walkRequest = { walkParams = { rotation = 0; translation = { x = 0; y = 0; }; }; }; }
set motion:motionRequest { motion = specialAction; specialActionRequest = { specialAction = walkImproved; }; walkRequest = { walkParams = { rotation = 0; translation = { x = 0; y = 0; }; }; }; }
set motion:motionRequest { motion = specialAction; specialActionRequest = { specialAction = walkBackward; }; walkRequest = { walkParams = { rotation = 0; translation = { x = 0; y = 0; }; }; }; }
set motion:motionRequest { motion = specialAction; specialActionRequest = { specialAction = walkLeft; }; walkRequest = { walkParams = { rotation = 0; translation = { x = 0; y = 0; }; }; }; }
set motion:motionRequest { motion = specialAction; specialActionRequest = { specialAction = walkRight; }; walkRequest = { walkParams = { rotation = 0; translation = { x = 0; y = 0; }; }; }; }
```

Fig. 4. The user interface of SimRobot while simulating a robot on a KidSize field. The internal frames show (from left to right, top to bottom): a tree of all simulated objects, a simulated image of a camera pointed on an omni-directional mirror, a view of the whole scenario, a close view of the physics of a single robot, and the console which is used to enter interactive commands.

during the design phase of robot systems and may therefore lead to better decisions and cost savings. On the other hand, it supports the process of software development by providing an replacement for robots that are currently not on-hand (e. g. broken or used by another person) or not able to endure long running experiments (e. g. learning tasks). Furthermore, the execution of robot programs inside a simulator offers the possibility of directly debugging and testing them.

The BreDoBrothers use SimRobot [7], a robot simulator which is able to simulate arbitrary user-defined robots in three-dimensional space. It includes a physical model which is based on rigid body dynamics. Thus it is possible to create an accurate simulation of a Kondo KHR-1 robot which is playing soccer (cf. Fig. 4).

SimRobot is able to simulate an arbitrary number of robots. The complete source code that was developed for a robot is compiled and linked into this application. Additionally, SimRobot provides several different visualizations for data generated by the different modules and allows direct actuator manipulation

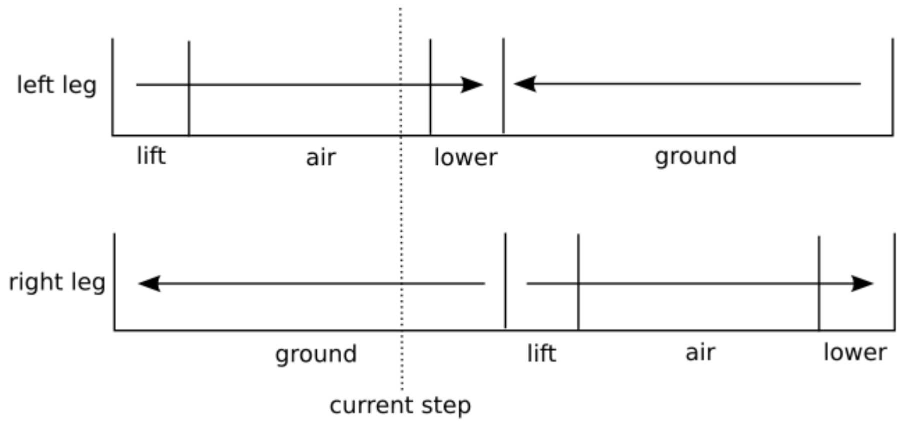


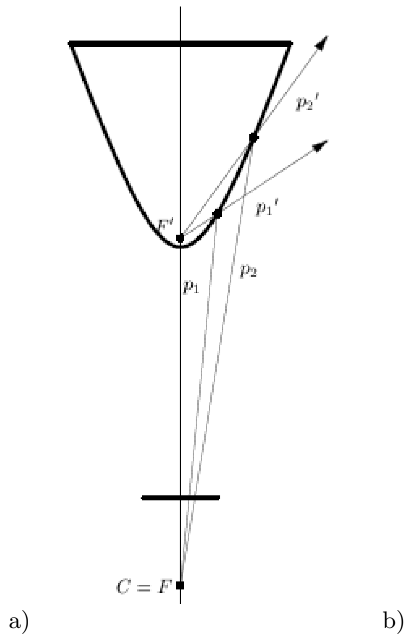
Fig. 5. The dashed line shows the current step of the motion cycle. The right leg uses an offset so that one foot is on the ground at all time. The arrows display the moving direction of the foot during the individual phases. The lengths of the phases are part of the parameters that are learned using evolutionary algorithms.

as well as the interaction with movable objects in the scene (i. e. the ball and robots) to create different situations to be tested.

5 Robot Motion

The BreDoBrothers use different approaches to generate gaits for the robot. One of these approaches is the use of inverse kinematics to calculate the joint angles which are necessary for the foot plates to reach points specified by a give walk pattern. The easiest of those patterns moves the feet in a rectangular way, where one of them is always on the ground. One cycle is divided into four different phases: foot on the ground, lifting foot, moving foot through the air, lowering foot(cf. Fig. 5). Depending on the move direction the foot is moving forwards or backwards while it is on the ground or respectively in the air. This allows for a lot of different parameters which affects the stability and speed of the movement. By applying different patterns to both feet, omni-directional walking motions are generated.

Since using only a handtuned parameter set so far, the BreDoBrothers intend to optimize the gait parameters to increase stability and speed of the robot gait. It is planned to optimize them by using an approach based on the work of [8], where the author successfully applied evolutionary algorithms to optimize the gait parameter sets on the Sony AIBO models ERS-210 and ERS-7. Beside this approach, the BreDoBrothers intend to apply another approach which uses the Particle Swarm Optizimation (PSO) introduced in [9]. PSO is a population-based stochastic optimization algorithm, motivated by social behavior of bird flocking and fish schooling. In the beginning, a population of random solutions is



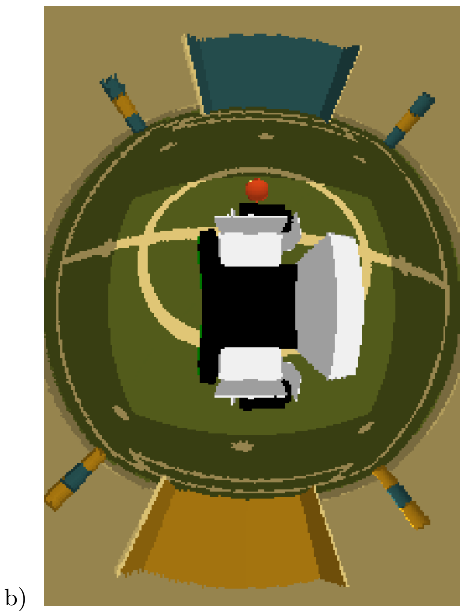


Fig. 6. a) A schematic of an omnidirectional vision system using a hyperbolic mirror.[12] F marks the focal point of the camera, F' marks the focal point of the mirror. The mirror deflects the rays p_1 and p_2 to $p_1{}'$ and $p_2{}'$. b) The omni-directional view from the simulated camera.

initialized each with position and velocity in the hyperdimensional search space. At each of the following optimization steps, each particle's fitness is calculated by a given fitness function. Then the velocity and position for each particle is updated up on the best position in the swarm found so far and the best ever position of the particle. So the particles search for the global optimum by making use of the experiences of the particle swarm.

PSO is planned to be used with a small population for the gait optimization to keep it usable and comparable to the other mentioned approach. As fitness criteria beside the velocity, we intend to take the conditions for stability like Center of Gravity (COG) and Zero Moment Point (ZMP) and furthermore the smoothness of the gait, which can be read of the acceleration sensors.

The same approaches will be used for kicking the ball. Given the position and the desired direction of the ball, the robot will be able to calculate a three dimensional vector from his current foot position to the ball that allows a fast, high or very precise kick. Again this is affected by a lot of parameters like the positions of the arms to increase the stability and the height to which the robot's foot is lifted. Those will also be optimized using the same framework that is used for optimizing the walking parameters.

Other motions like standing up are represented by so called special actions. Those contain a sequence of sets of joint angles that are executed in a specified interval, performing the desired action. Each set gets executed for a number of milliseconds as defined by the special action. During this time the joint angles may be interpolated to allow fluid movements or be simply set, ignoring the previous values of the servos.

6 Vision

The main sensor to observe the environment is a vision system. We intend to equip our robots with an omni-directional vision system using a camera which is directed at a hyperbolic mirror. (cf. Fig. 6a) This mirror allows the camera to receive a 360° image that contains areas very close to the robot and areas that are far away. The main drawbacks of this system are the distortion of the image and the smaller size of objects compared to a normal directed camera.

We also implemented this sensor in our simulation (cf. Fig. 6b). This is realized using four virtual cameras (three horizontal, one vertical) whose images are used as input to calculate the omni-directional image.

Currently, one of our robots (cf. Fig.1a) uses directed vision. In this case, the approach of [10], which uses a horizon-aligned grid for analyzing only parts of an image, has been well proven. Recognition algorithms for different features (e. g. lines, goals, and the ball) are already part of the GermanTeam's vision system, which is described comprehensively in [11]

7 World Modeling and Behavior Control

For self-localization, the BreDoBrothers will use a particle filter based on the Monte Carlo method [13]. This approach has already been proven to provide accurate results in a similar environment [14, 15]. Additionally, it is able to deal with the kidnapped robot problem, which often occurs in RoboCup scenarios. The tracking of movable objects, i. e. the ball and other robots will be realized via Kalman filters [16, 17].

The robot's behavior is currently programmed using the XABSL engine [18] in combination with YABSL [11], a C-like behavior specification language. The overall robot behavior is split up into a set of simple behaviors which are interdependently arranged as nodes in an acyclic, directed graph (cf. Fig. 7a). Single nodes of the graph are modeled by using state machines whose transitions are controlled by decision trees (cf. Fig. 7b/c).

Continuous robot motion planning (e. g. for obstacle avoidance) is realized via the integration of the potential field implementation of [19].

8 Conclusion

The BreDoBrothers are a new team that intends to participate in the Humanoid League. Due to a Four-Legged League background of most team members, we

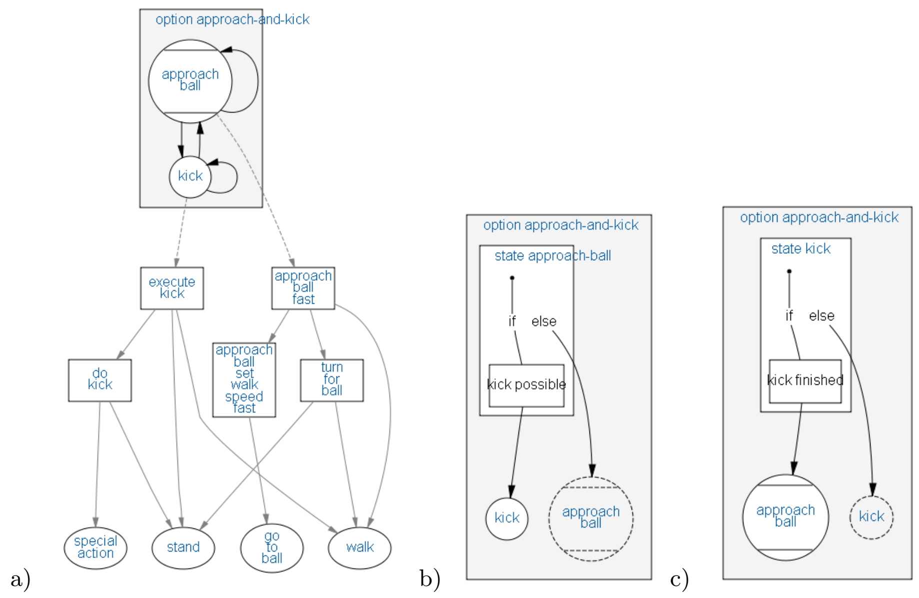


Fig. 7. a) A hierarchy of simple behaviors for playing a ball. b / c) Simple decision trees for determining state transitions inside a behavior.

have a strong focus on software development instead of robot construction. Despite being a newcomer equipped with standard robots, we hope to play soccer with a reasonable performance by reusing as much software from another league as possible. In this paper, we showed the similarities between the two legged leagues and the possibilities for software migration.

References

1. Düffert, U., Jüngel, M., Laue, T., Lötzsch, M., Risler, M., Röfer, T.: GermanTeam 2002. In: RoboCup 2002 Robot Soccer World Cup VI, Gal A. Kaminka, Pedro U. Lima, Raul Rojas (Eds.). Number 2752 in Lecture Notes in Artificial Intelligence, Springer (2003) More detailed in http://www.tzi.de/kogrob/papers/GermanTeam2002.pdf.
2. Röfer, T., Dahm, I., Düffert, U., Hoffmann, J., Jüngel, M., Kallnik, M., Lötzsch, M., Risler, M., Stelzer, M., Ziegler, J.: GermanTeam 2003. In Browning, B., Polani, D., Bonarini, A., Yoshida, K., eds.: RoboCup 2003: Robot Soccer World Cup VII. Lecture Notes in Artificial Intelligence, Springer (2004) to appear.
3. Röfer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Jüngel, M., Laue, T., Lötzsch, M., Nistico, W., Spranger, M.: Germanteam 2004. In: RoboCup 2004: Robot World Cup VIII. Number 3276 in Lecture Notes in Artificial Intelligence, Springer (2005)
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)
5. Behnke, S., Müller, J., Schreiber, M.: Using Handheld Computers To Control Humanoid Robots. In: Proceedings of 1st International Conference on Dextrous Autonomous Robots and Humanoids (darh2005), Yverdon-les-Bains - Switzerland. (2005) paper nr. 3.2
6. Röfer, T.: An architecture for a national robocup team. In: RoboCup 2002 Robot Soccer World Cup VI, Gal A. Kaminka, Pedro U. Lima, Raul Rojas (Eds.). Number 2752 in Lecture Notes in Artificial Intelligence, Springer (2003) 417–425
7. Laue, T., Spiess, K., Röfer, T.: SimRobot - A General Physical Robot Simulator and Its Application in RoboCup. In: RoboCup 2005: Robot Soccer World Cup IX. Lecture Notes in Artificial Intelligence, (Springer)
8. Röfer, T.: Evolutionary gait-optimization using a fitness function based on proprioception. In: RoboCup 2004: Robot Soccer World Cup VIII. Number 3276 in Lecture Notes in Artificial Intelligence, Springer (2005) 310–322
9. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks. (1995)
10. Bach, J., Jüngel, M.: Using pattern matching on a flexible, horizon-aligned grid for robotic vision. Concurrency, Specification and Programming - CSP'2002 **1** (2002) 11–19
11. Röfer, T., Laue, T., Weber, M., Burkhard, H.D., Jüngel, M., Göhring, D., Hoffmann, J., Altmeyer, B., Krause, T., Spranger, M., Schwiegelshohn, U., Hebbel, M., Nisticó, W., Czarnetzki, S., Kerkhof, T., Meyer, M., Rohde, C., Schmitz, B., Wachter, M., Wegner, T., Zarges, C., von Stryk, O., Brunn, R., Dassler, M., Kunz, M., Oberlies, T., and, M.R.: GermanTeam RoboCup 2005. Technical report (2005) Available online: http://www.germanteam.org/GT2005.pdf.
12. Svoboda, T.: Central Panoramic Cameras - Design, Geometry, Egomotion. PhD thesis, Czech Technical University (1999)
13. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In: Proc. of the National Conference on Artificial Intelligence. (1999)
14. Röfer, T., Laue, T., Thomas, D.: Particle-filter-based self-localization using landmarks and directed lines. In: RoboCup 2005: Robot Soccer World Cup IX. Lecture Notes in Artificial Intelligence, (Springer)
15. Röfer, T., Jüngel, M.: Vision-based fast and reactive monte-carlo localization. In: IEEE International Conference on Robotics and Automation, Taipei, Taiwan, IEEE (2003) 856–861
16. Kalman, R.E.: A new approach to linear filtering and prediction problems. Transactions of the ASME–Journal of Basic Engineering **82** (1960) 35–45
17. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005)
18. Lötzsch, M., Bach, J., Burkhard, H.D., Jüngel, M.: Designing agent behavior with the extensible agent behavior specification language XABSL. In Polani, D., Browning, B., Bonarini, A., Yoshida, K., eds.: RoboCup 2003: Robot Soccer World Cup VII. Number 3020 in Lecture Notes in Artificial Intelligence, Padova, Italy, Springer (2004)
19. Laue, T., Röfer, T.: A behavior architecture for autonomous mobile robots based on potential fields. In: RoboCup 2004: Robot Soccer World Cup VIII. Number 3276 in Lecture Notes in Artificial Intelligence, Springer (2005) 122–133