B-Human

Team Description for RoboCup 2007

Thomas Röfer[1], Christoph Budelmann[1], Martin Fritsche[3], Tim Laue[3], Judith Müller[3], Cord Niehaus[3], and Florian Penquitt[3]

[1] Deutsches Forschungsinstitut für Künstliche Intelligenz GmbH,
Sichere Kognitive Systeme, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
E-Mail: {thomas.roefer,christoph.budelmann}@dfki.de
[2] Fachbereich 3 - Mathematik und Informatik, Universität Bremen,
Postfach 330 440, 28334 Bremen, Germany
E-Mail: {fritsche,timlaue,judy,cniehaus,penquitt}@tzi.de

1 Introduction

B-Human consists of several Computer Science researchers and students from the Universität Bremen and additionally one electrical engineering student from Universität Darmstadt. Due to this distribution of expertise our team is not primarily interested in robot construction, but rather concentrates on the software components.

Our current robot platform is build from a commercial standard robot construction kit (cf. Sect. 2) carrying a low-level controller board for actuator control or sensor interpretation and a standard PDA for high-level computations.

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 [1, 2]. Thus the implementation of the general architecture (cf. Sect. 3) and the simulation environment (cf. Sect. 4) as well as several algorithms (cf. Sect. 6) have been adopted unmodified for the most parts.

2 The Robots

The commercially available Bioloid robot kit from Robotis is the basis for our robots (cf. Fig. 1)), since this kit has already been used in RoboCup competitions by different other teams. We upgraded this platform with an improved version of our own controller board which has already been a part of our 2006 Kondo platform. The new version of this controller board is again constructed around an ATmega128 micro controller but includes a lot of new sensors and features, i. e. a 3-axis gyroscope and acceleration sensor, a magnet-compass or pressure sensors in both feet.

For all high-level on-board computations, we equipped the robots with standard PDAs from Fujitsu Siemens running Microsoft Windows Mobile. The PDAs also have an integrated 1.3 mega pixel camera which provides us with about 10 frames per second at a 320x240 resolution. This widespread approach has already been described comprehensively by [3].

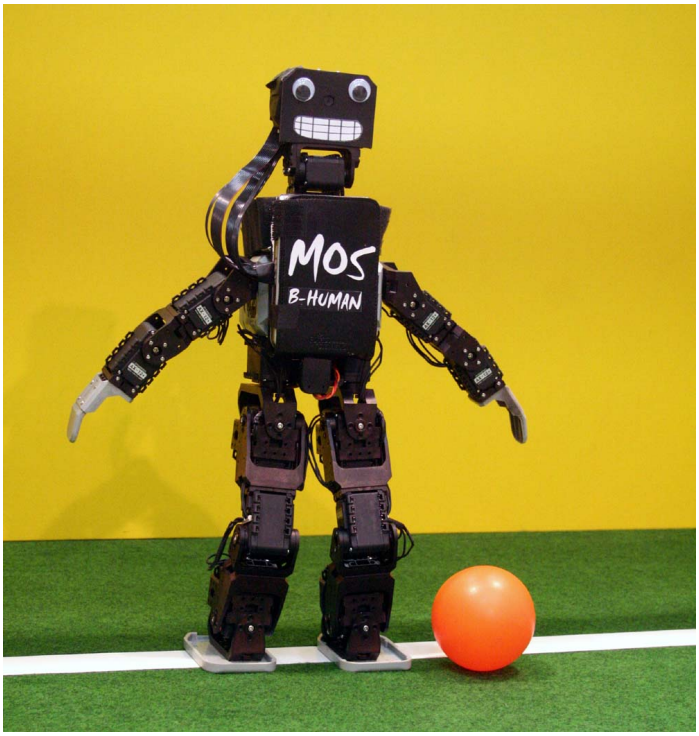

Fig. 1. Mos – one of our robots.

3 Software Framework

The B-Human control software is based on the software framework of the GermanTeam [4]. It is possibly the architecture [5] used most often for controlling real robots in RoboCup. Currently, more than one third 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 [6] 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 B-Human software runs under Microsoft Windows Mobile 2003 SE. Thereby this platform is the fifth one supported by the framework besides the Sony AIBO, Sony's Open-R Emulator running under Cygwin, Microsoft Windows (under the Simulator SimRobot, cf. Sect. 4), and Linux (on the autonomous wheelchair *Rolland* [7] using an embedded PC-104 system). In 2006, the software used several threads for concurrent execution. However, since Windows Mobile limits each process to access only 32 MB of RAM, this year, parallel processes are employed to give access to the full memory size of the PDA.

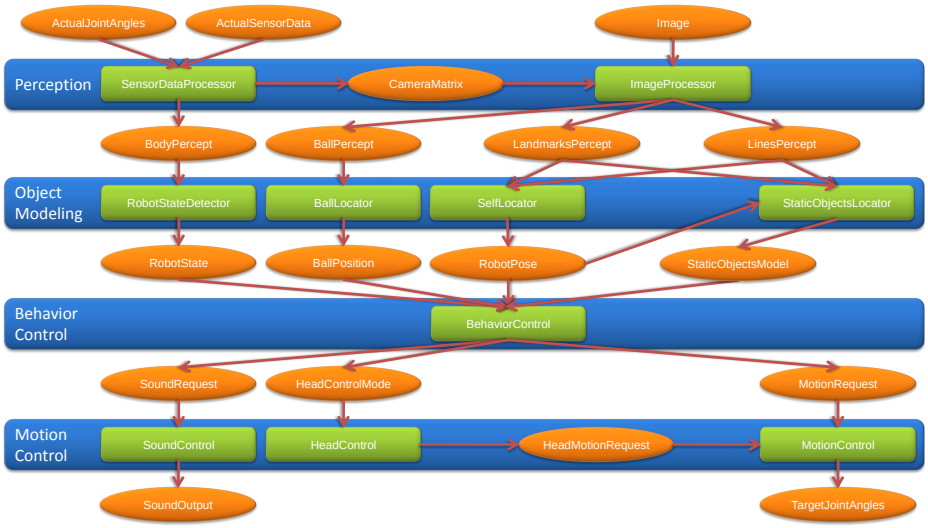

Fig. 2. Tasks and representations used for playing robot soccer.

In addition to controlling the robots using on-board PDAs, the framework running under the desktop version of Windows can also directly control the 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 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 B-Human robots 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, and the ball. 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.

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, and the position of the ball (opponent players are currently ignored). 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

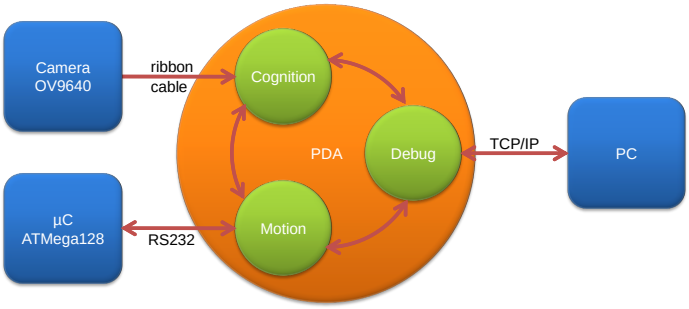

Fig. 3. The processes running on the PDA

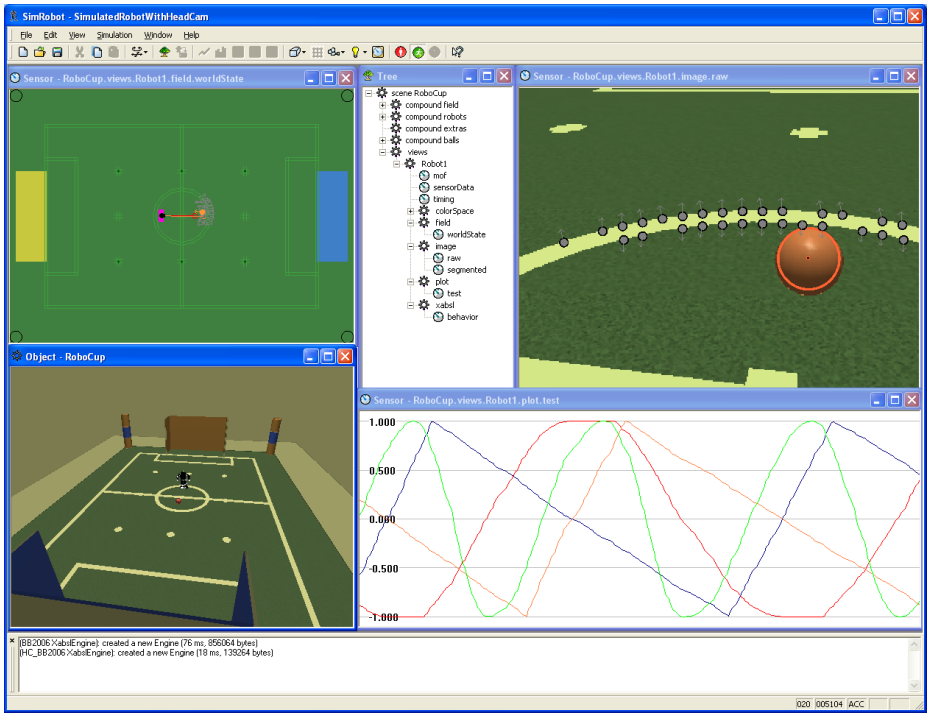

Fig. 4. The user interface of SimRobot while simulating a robot on a KidSize field. The internal frames show (from left to right, top to bottom): the robot's world state estimate, a tree of all simulated objects, a simulated image of the camera on a pan-tilt unit, a view of the whole scenario, a plot of the robot's gait trajectories, 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-

a) b)

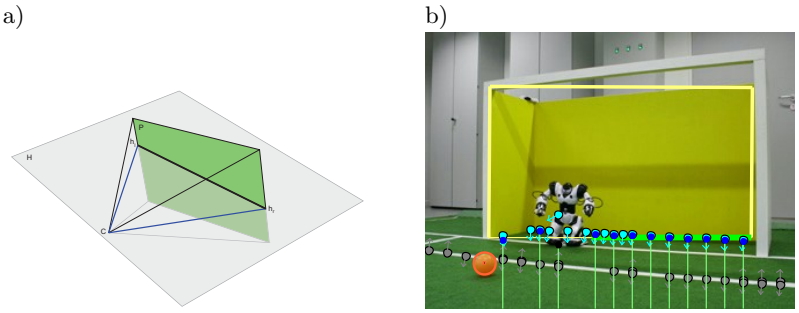

Fig. 5. Grid-based approach for robot vision. a) Construction of a horizon-aligned grid (image taken from [9]). b) An image from the robot's perspective. Recognized features on the field are labeled by different dots (edges of significant field lines), a circle (the ball), and lines (the goal and its free part).

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.

We use SimRobot [8], 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 humanoid robot which is playing soccer (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 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 Vision

Our Bioloid robots use a directed vision system, which consist of a color camera which is mounted on a pan-tilt unit. For this application, the approach of [10], which uses a horizon-aligned grid (see Fig. 5a) for analyzing only parts of an image, has been well proven. The parameters of the horizon results from the position and orientation of the camera, which may be computed from the current states of the robot's joints on both platforms.

Each grid line is scanned pixel by pixel. During the scan, each pixel is classified by color. A characteristic series of colors or a pattern of colors is an indication of an object of interest which has to been analyzed in more detail. Recognition algorithms for the most important features (e. g. lines, landmarks, and the ball) are already part of the GermanTeam's vision system (cf. Fig. 5b) [9] and may be used—in most cases—without any modifications, despite the adaption to a different image size and the byte order of pixels, of course.

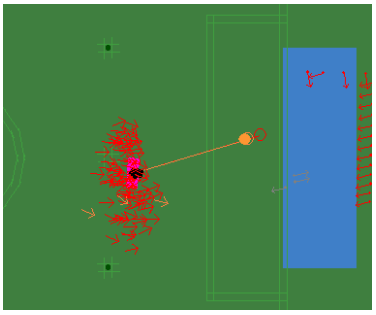

Fig. 6. Visualization of the internal state of the self localization. The large arrows denote the potential poses of the robot. The small arrows near the goal denote perceived goal points. The mismatch between these points and the goal indicates the current localization error.

The Ball. Since the KidSize class and the Four-Legged League use the same balls, the ball recognition approach did not need to be changed at all. Currently, a Levenberg-Marquardt fitting of a circle, given a set of points lying on the edges of the ball, is applied [11].

Landmarks. The most significant types of landmarks are the the two goals and the four beacons. While the goals do not differ in color and shape from the ones in the Four-Legged League, the approach described in [4] works fine, especially due to the extraordinary size of the goals. However, the recognition of beacons demands some additional work. While the ambiguousness of the color sequences does not cause any significant problems (cf. Sect. 6), the lack of pink (which is always included in the AIBO beacons) and the possibility to take the lower part of a humanoid beacon for a goal, are problematic issues. These can be solved by using the standard goal recognition algorithm for detecting these lower parts, too, and distinguishing between goals and beacon parts by the height of their upper border.

Field Lines. To improve the position estimate or to bridge phases without any perceptions of major landmarks (e. g. while tracking a close ball), perceptions of field lines are quite valuable and easy to compute [12, 13]. In the Humanoid League, it is even easier to detect lines than in the Four-Legged League, since the lines are twice as wide and are seen from a higher perspective.

6 World Modeling and Behavior Control

For self-localization, B-Human will use a particle filter based on the Monte Carlo method [14]. This approach has already been proven to provide accurate results in a similar environment [13, 15]. Additionally, it is able to deal with the kidnapped robot problem, which often occurs in RoboCup scenarios.

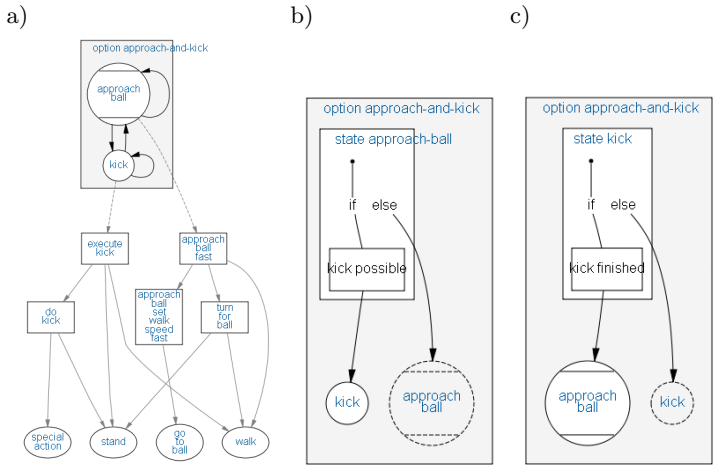

Fig. 7. a) A hierarchy of simple behaviors for playing a ball. b / c) Simple decision trees for determining state transitions inside a behavior.

The estimation of the ball's position and velocity as well as the relative positions of static objects (i. e. the opponent goal and the last seen beacon) is also realized via particle filters.

The robot's behavior is currently programmed using the XABSL engine [6] in combination with YABSL [9], 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. Single nodes of the graph are modeled by using state machines whose transitions are controlled by decision trees.

Continuous robot motion planning is realized via the integration of the potential field implementation of [18].

7 Robot Motion

Motion is the part of robot control in which humanoid robots differ the most from their four-legged counterparts. Standing and walking on four feet is much more stable than walking on two, i. e. while quadruped motion on a plane surface can be performed without any feedback at all, bipedal motion typically requires to keep the balance. However, since the feet of the robots in the humanoid league are still allowed to be quite large, at least parts of the motions can be performed without sensory feedback. In case of the B-Human, motions such as kicking and standing up (cf. Fig. 8) are represented by so called special actions. They consist of 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 are either interpolated to allow fluid movements or they are simply set, ignoring

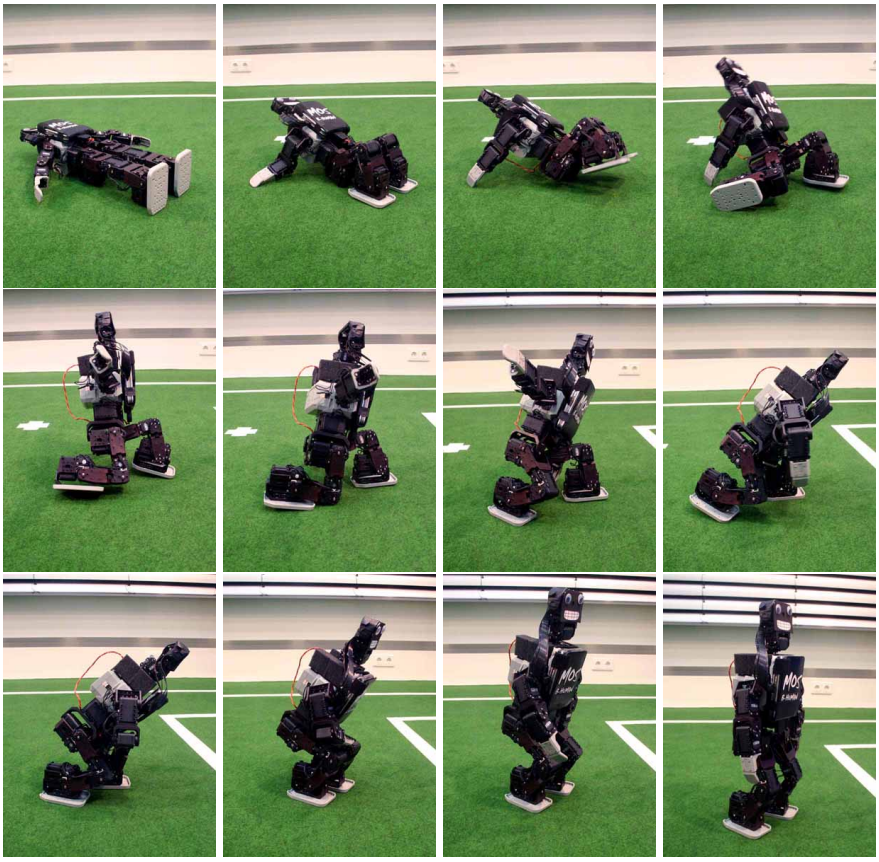

Fig. 8. A robot standing up.

the previous values of the servos. A transition network defines the prerequisites for each motion, e. g. that the robot has to stand before it performs a certain kick.

While special actions are static in nature, walking is not. In the soccer scenario, it is desirable to be able to move in any combination of forward, sideward, and rotational motion, i. e. omni-directional. The maximum speeds that can be reached are limited by the lengths of the robot's legs (more precisely: the distance between hip and ankle) and the step frequency. In general, the gait of the B-Human is similar to the one used by the GermanTeam for the AIBO. As the AIBO, the Bioloid has two joints for each leg in the hip (roll, pitch), and one in the knee. In contrast to the AIBO, two further joints per leg allow controlling pitch and roll of the feet, i. e. they function as the ankles, and each leg has an additional joint in the hip that allows for rotating the whole leg around the vertical axis, enabling the robot to turn.

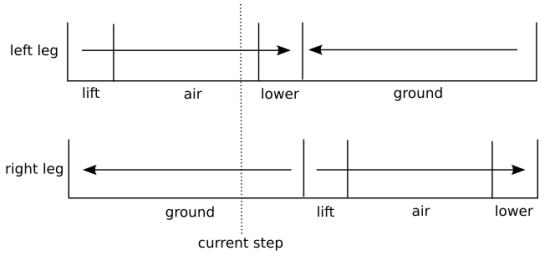

Fig. 9. The dashed line shows the current step of the motion cycle. The right leg uses an offset so that one foot is on the ground at all time. The arrows display the moving direction of the foot during the individual phases.

Walking means that the feet move along certain trajectories relative to the center of the body. In the walking engine used by the B-Human, these trajectories are calculated in Cartesian space, and then they are transformed into joint angles by inverse kinematics. The approach for controlling the upper three joints per leg is very similar to the one used by the GermanTeam for the AIBO, and is described in detail for the AIBO in [19]. The feet, controlled by the remaining two joints in each leg, typically stay parallel to the ground, i. e. the angles of the two ankle joints just compensate for the pitches and rolls resulting from the states of the other three joints of each leg. The trajectories can have different shapes that are controlled by a vast number of parameters, such as foot origin in (x, y, z), step height, step shape (e. g. rectangle, ellipse, half-ellipse, etc.), maximum forward/sideward step size, etc. Depending on the shape of the trajectory of a step, the walk cycle runs through different phases, e. g. for a rectangular shape: ground phase, lifting phase, swinging phase, and lowering phase (cf. Fig. 9). The phases of the two legs are shifted by half a phase. In addition to moving the legs, the robot also has to shift its weight to avoid falling down when one leg is lifted. Effectively this is done by continuously moving the feet's origins from left to right and back according to the walking phase. In addition, it is also possible to tilt the body, swing the arms, and tilt and roll the feet (in addition to their keeping parallel to the ground) based on the current walking phase. However, the best results were achieved when these additional motions were not used.

Although gaits using this approach can be quite stable, sensor-based balancing increases that stability a lot. Using the measurements of the three acceleration sensors (acc_x, acc_y, acc_z) of the robot, the body tilt and amplitude of the body's sideward swinging is controlled. The body tilt is simply derived from the measured body pitch $(\text{atan2}(acc_x, acc_z))$, while the sideward swinging amplitude is determined from the averaged difference between the measurements of the sideward acceleration acc_y and the second derivative of the originally desired sideward motion that is defined as a parameter of the gait.

For the control of the pan-tilt unit, the XABSL engine has been used, too. It allows an easy specification of different state-based tracking and searching

behaviors for the robot's head. In general, the implementation does not differ from the one for the AIBO ERS-7. In detail, this model has an additional second tilt joint. That joint only used for some special motions (e. g. catching the ball) which are of no use for a humanoid robot.

8 Conclusion and Future Plans

The upcoming German Open 2007 will be an important event to test our new hardware platform under real tournament conditions. There are several further improvements that shall be finished until the beginning of this event, e. g. a new fully customized torso segment to hold and protect the PDAs and batteries. The gait parameters will be optimized using Particle Swarm Optimization (PSO) [21], based on our experiences made with the Kondo KHR-1.

Acknowledgments

This work has been funded by the Deutsche Forschungsgemeinschaft in context of the Schwerpunktprogramm 1125 (*Kooperierende Teams mobiler Roboter in dynamischen Umgebungen*).

References

1. Laue, T., Röfer, T.: Getting Upright: Migrating Concepts and Software from Four-Legged to Humanoid Soccer Robots. In: Proceedings of the Workshop on Humanoid Soccer Robots in conjunction with the 2006 IEEE International Conference on Humanoid Robots. (2006)
2. Röfer, T., Fritsche, M., Hebbel, M., Kindler, T., Laue, T., Niehaus, C., Nistico, W., Schober, P.: BreDoBrothers - Team Description for RoboCup 2006. In: RoboCup 2006: Robot Soccer World Cup X. Lecture Notes in Artificial Intelligence, Springer (2006)
3. 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
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. 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
6. 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)

7. Mandel, C., Huebner, K., Vierhuff, T.: Towards an autonomous wheelchair: Cognitive aspects in service robotics. In: Proceedings of Towards Autonomous Robotic Systems (TAROS 2005). (2005) 165–172
8. 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)
9. 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.
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., Burkhard, H.D., Hoffmann, J., Jüngel, M., Göhring, D., Lötzsch, M., Düffert, U., Spranger, M., Altmeyer, B., Goetzke, V., v. Stryk, O., Brunn, R., Dassler, M., Kunz, M., Risler, M., Stelzer, M., Thomas, D., Uhrig, S., Schwiegelshohn, U., Dahm, I., Hebbel, M., Nisticó, W., Schumann, C., Wachter, M.: GermanTeam RoboCup 2004 (2004) Only available online: http://www.robocup.de/germanteam/GT2004.pdf.
12. Röfer, T., Jüngel, M.: Vision-Based Fast and Reactive Monte-Carlo Localization. IEEE International Conference on Robotics and Automation (2003)
13. 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)
14. 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)
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. 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
19. Röfer, T., Laue, T., Weber, M., Burkhard, H.D., Jüngel, M., Göhring, D., Hoffmann, J., Altmeyer, B., Krause, T., Spranger, M., v. Stryk, O., Brunn, R., Dassler, M., Kunz, M., Oberlies, T., Risler, M., etc.: GermanTeam RoboCup 2005 (2005) http://www.germanteam.org/GT2005.pdf.
20. 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
21. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks. (1995)