Getting Upright: Migrating Concepts and Software from Four-Legged to Humanoid Soccer Robots

Tim Laue
Department of Mathematics and Computer Science
Universität Bremen
Bremen, Germany
E-Mail: timlaue@informatik.uni-bremen.de

Thomas Röfer
DFKI-Lab Bremen
Safe Cognitive Systems
Bremen, Germany
E-Mail: Thomas.Roefer@dfki.de

***Abstract*— Besides the construction of the robots, the development of control software and tools for simulation and debugging are major topics of interest in the domain of soccer playing humanoid robots. In the RoboCup Four-Legged League, the whole research has—due to the existence of a standard platform—been focused on software development since many years. Therefore, a migration of successful software solutions to humanoid soccer robots is an obvious advancement. In this paper, we point out the similarities between these two domains and describe how to use approved approaches in areas as e.g. vision, localization, or simulation. As a successful example, we describe the migration of GermanTeams's Four-Legged software to the BreDoBrothers humanoid team.**

I. Introduction

The BreDoBrothers [1] are a joint humanoid soccer 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 [2]–[5], which has become world champion in the Four-Legged League twice.

Starting a new humanoid team—which had the aim of participating in RoboCup competitions in its first year—from scratch is a challenging task. Since our team has been consisting of computer scientists and thus has not been primarily interested in robot construction, the usage of a commercially available robot platform (cf. Sect. II) and the transfer of as much code and experiences from the Four-Legged League to the Humanoid League as possible have been an obvious approach. That robot soccer league exists already since 1998 and has since then exclusively focused on software development because of the existence of a standard robot platform. Many different issues as robot vision, localization, object tracking, or robot behavior control have been investigated and solved in a robot soccer scenario. The close similarities to the Humanoid League make many of these solutions directly applicable to this domain and therefore offer an interesting point to start from. In the long view, such a proceeding contributes to the establishment of a higher standard of soccer software for humanoid robots.

This paper is organized as follows: Section II describes the robots used and points out the similarities between the two soccer environments. After that, the implementation of the general software framework (cf. Sect. III) and the application of a dynamic robot simulation (cf. Sect. IV) are presented.

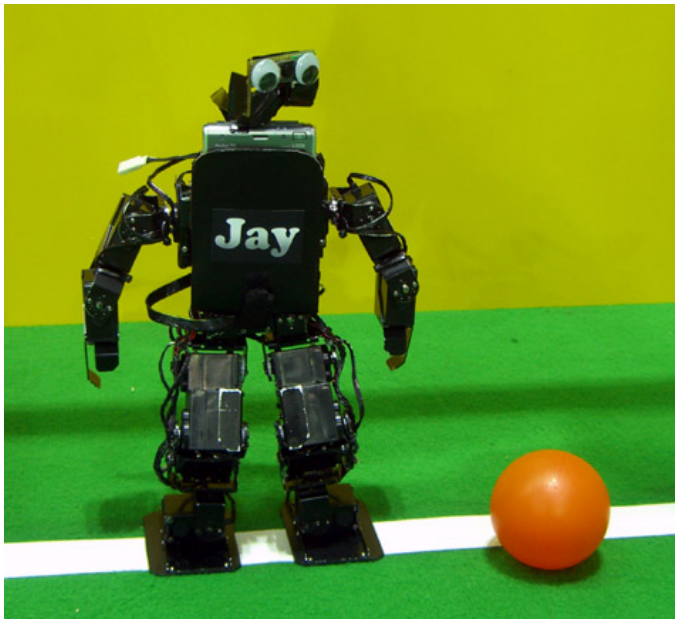


Fig. 1. Jay, one of our KHR-1-based robots, equipped with a PDA in front of its chest and a pan-tilt unit carrying a camera.

The description of the different algorithms is divided into three parts: Vision (cf. Sect. V), World Modeling and Behavior(cf. Sect. VI), and Robot Motion (cf. Sect. VII). The paper ends with an outlook (cf. Sect. VIII) on more techniques from the Four-Legged domain which have not been ported so far but appear to be useful in the near future.

II. The Robots and the Environment

We use the commercially available KHR-1 robot kit from Kondo as a basis for our robots (cf. Fig. 1), 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, a custom-made board with an Atmel ATMega128 controller, accelerometers, and a gyroscope is used. On top of the robot, a pan-tilt unit assembled from small servo motors is attached. This unit allows head motions similar to the ones of the AIBO robot (cf. Sect. VII).

Proceedings of the Workshop on Humanoid Soccer Robots
of the 2006 IEEE-RAS International Conference on Humanoid Robots
Genoa, Italy, December 4, 2006
pp. 32-39
ISBN: 88-900426-2-1

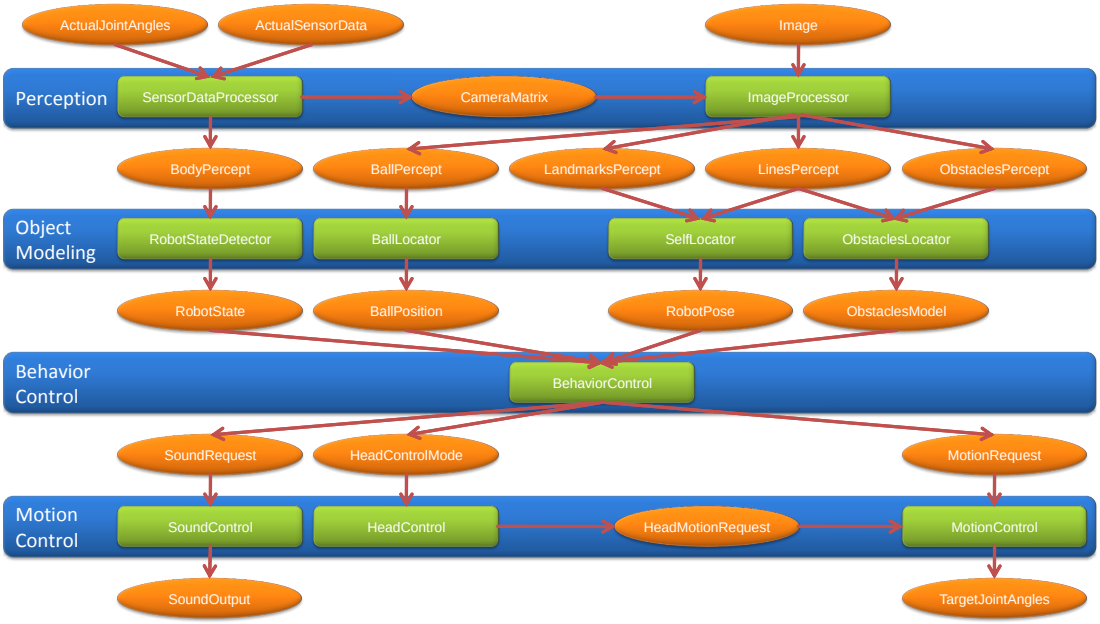


Fig. 2. Tasks and representations used for playing robot soccer.

For all on-board computations, we equipped the robots with standard Pocket LOOX PDAs from Fujitsu Siemens. This widespread approach has been described comprehensively by [6]. This offers computational capabilities comparable with the ones of the AIBO robot. Nevertheless some algorithms run slower due to the missing floating point unit on these devices. The color camera of the PDA—which provides images at a resolution on 320×240 pixels—has been lead through the case and been mounted on the pan-tilt unit. By using the internal camera, framerates of more than 10 images per second (including all processing steps) could be achieved.

The environments of both soccer leagues are quite similar: Legged robots with mainly passive sensors (i. e. cameras and proprioceptive sensors, no laser range findes or ultra sonic sensors) play on a heavily standardized field. All objects are color-coded (the ball is orange, the field is green, the goals and beacons are yellow and blue, the robots have to be mostly black) and at defined places (except the robots and the ball, of course). These colors and places are quite similar in both leagues.

III. Software Framework

The control software of the BreDoBrothers is based on the software framework of the GermanTeam [5]. It is presumably the architecture [7] 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.

The BreDoBrothers use PDAs running Microsoft Windows Mobile to control their robots. 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. IV), and Linux (on the autonomous wheelchair *Rolland* [8] using an embedded PC-104 system). 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.

A. 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. This simplifies the comparison and testing of different implementations for one task, e. g. different approaches of self localization.

It is also 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. IV). 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. IV).

B. Tasks

Figure 2 depicts the tasks and representations enabling the BreDoBrothers to play soccer. They can currently be structured

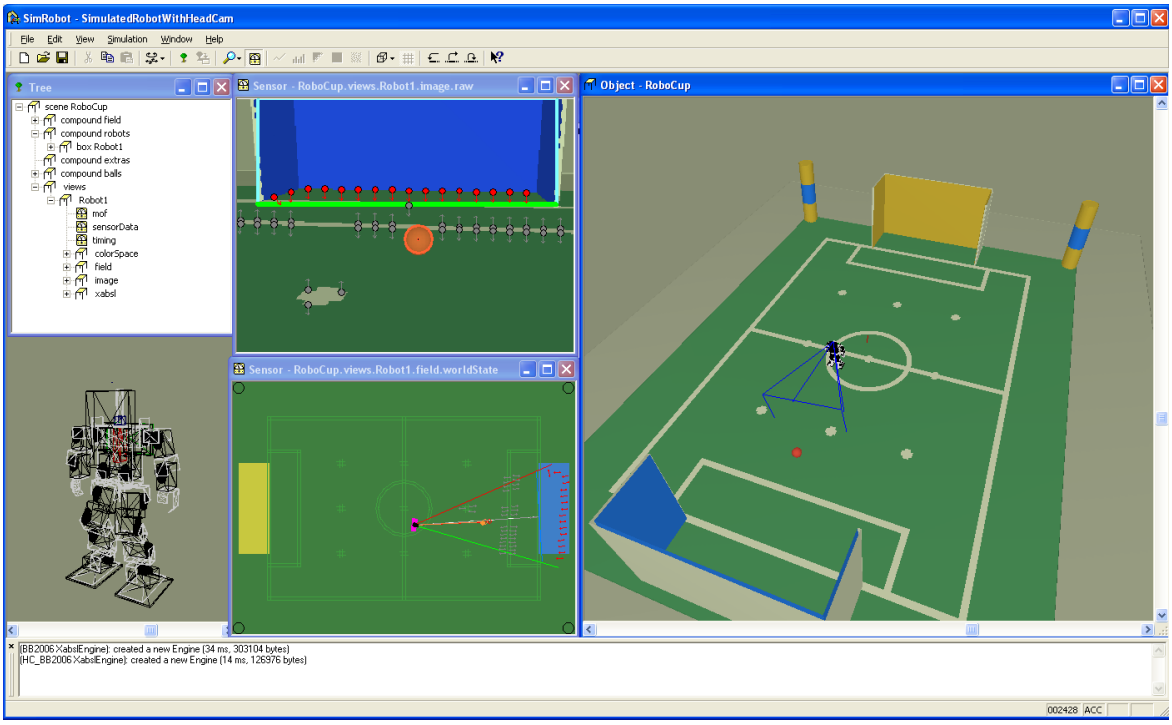


Fig. 3. The user interface of SimRobot while simulating a robot on a KidSize field. The internal frames show (from left to right, top to bottom): a tree of all simulated objects, a simulated image of a camera including several debug drawings which indicate recognized objects, a view of the whole scenario, a close view of the physics of a single robot, a drawing of the robots internal representation of the current world state, and the console which is used to enter interactive commands.

into four general levels[1]:

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, 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.

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 positions of 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, controlling the head's tracking motions, 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.

[1]In general, the architecture does not force any fixed number of levels. Additionally, representations can *skip* levels and may be used at a later position in the processing sequence.

C. 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. 4) 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

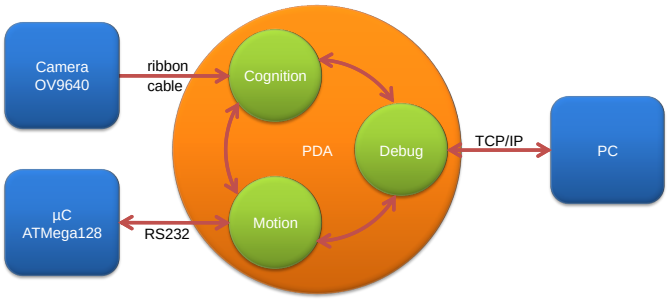
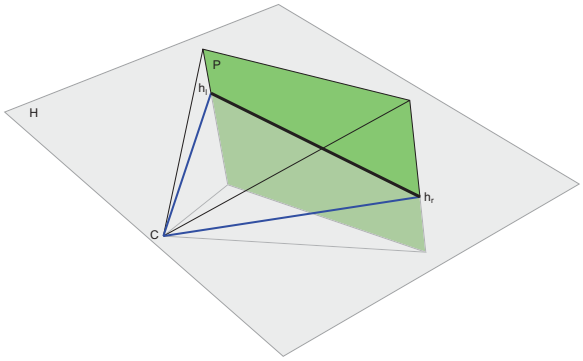
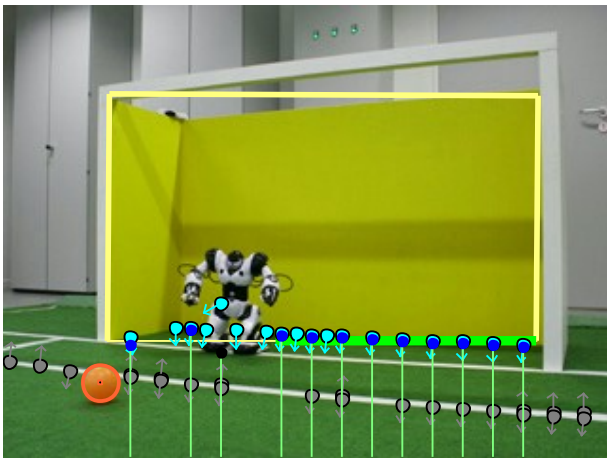


Fig. 4. The processes running on the PDA.



(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).

Fig. 5. Grid-based approach for robot vision.

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.

IV. 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 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 a 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 [10], 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. 3). In the past, this simulator has already been used for the simulation of the Sony AIBO robot and a wheeled Small Size robot [10].

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. These features—together with an interactive, scriptable command line and functions for communicating with real robots—make SimRobot the central tool for our software development and experiments.

V. Vision

Similar to the AIBO, our Kondo 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 [11], which uses a horizon-aligned grid (see Fig. 5(a)) 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. 5(b)) [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.

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 [12].

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 [5] 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. VI), 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 [13], [14]. 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.

Obstacles: A concept different from recognizing robots is detecting general obstacles [15], [16]. Instead of searching specific robot features (which heavily differ between the two leagues) in an image, the unoccupied regions, i. e. the green field including the white lines, are determined. Thus, areas not classified as free space have to be considered to be obstacles. This approach, which has originally designed for implicitly detecting AIBO robots, works with the black robots in the Humanoid League.

VI. World Modeling and Behavior Control

For self-localization, we use a particle filter (cf. Fig. 6) based on the Monte Carlo method [17]. This approach has already been proven to provide quite accurate results in the Four-Legged League's environment [14], [18]. Since the implementation used is not dependant on a specific RoboCup field — the positions and sizes of all objects and lines are separated in a configuration file —, it was directly usable on our humanoid robots. The particle filter approach does not have any problems with the ambiguous color codes on the beacons. Additionally, it is able to deal with the kidnapped robot problem, which often occurs in RoboCup scenarios through replacements of robot by referees or pick-ups by the team.

The tracking of the ball and the estimation of its velocity are realized via a Kalman filter [19], [20].

The robot's behavior is currently programmed using the XABSL engine [21] 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 (cf. Fig. 7a). Single nodes of the graph are modeled by using state machines the transitions of which are controlled by decision trees (cf. Fig. 7b/c). This specification language has already been used in all other robot soccer leagues: in the Middle Size League by the *COPS Stuttgart* [22], in the Small-Size League by *B-Smart* [23], and—of course—by the GermanTeam (and most teams that are based upon it) in the Four-Legged League.

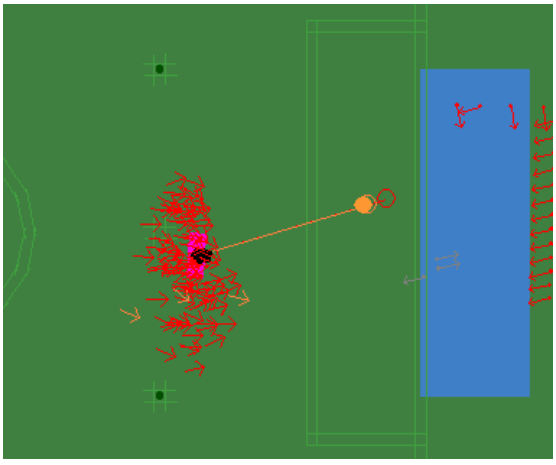


Fig. 6. Visualization of the internal state of the self localization. The large arrows denote the potential poses of the robot. The small arrows near the goal denote perceived goal points. The mismatch between these points and the goal indicates the current localization error.

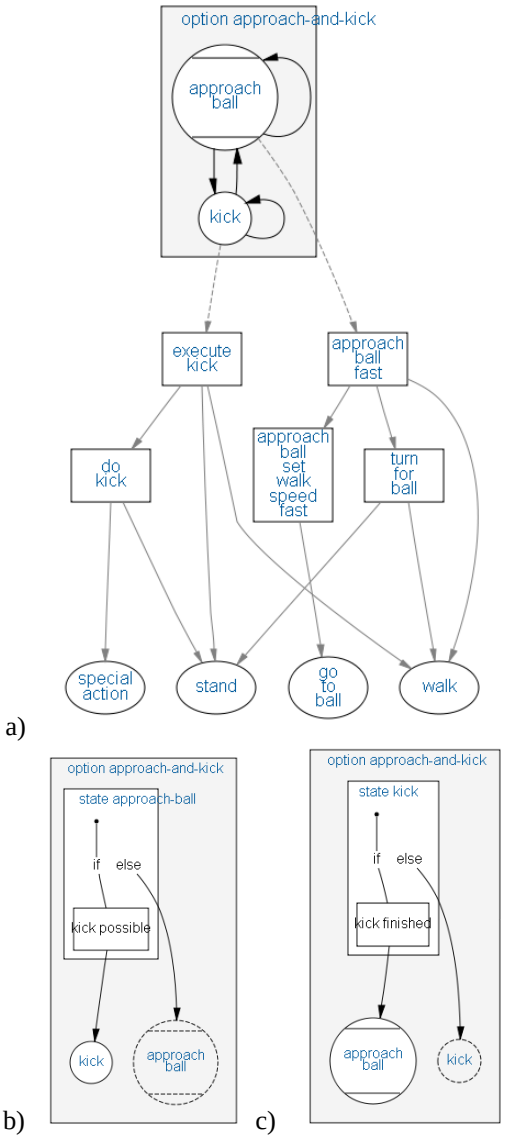


Fig. 7. a) A hierarchy of simple behaviors for playing a ball. b / c) Simple decision trees for determining state transitions inside a behavior.

VII. 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 BreDoBrothers, motions such as kicking and standing up 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 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 BreDoBrothers is similar to the one used by the GermanTeam for the AIBO. As the AIBO, the Kondo has two joints for each leg in the hip (roll, pitch), and one in the knee. Two further joints per leg allow controlling pitch and roll of the feet, i.e. they function as the ankles.

Walking means that the feet move along certain trajectories relative to the center of the body. In the walking engine used by the BreDoBrothers, 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 [24]. 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. 8). 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. Since our Kondo has no rotational joints in the legs, rotational movements must be achieved by moving the legs in different directions with both feet on the ground. This is integrated as additional phases in the normal step cycle before the lift phase and after the lowering phase, i.e. during these times, both feet touch the ground and move in opposite directions.

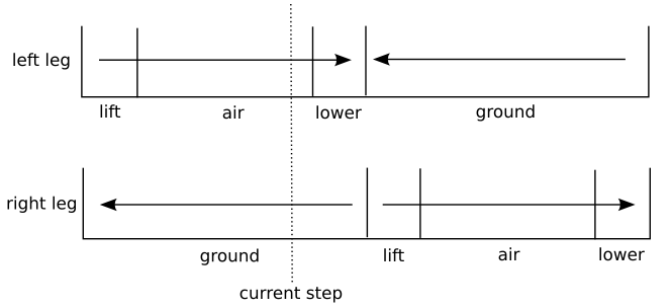


Fig. 8. The dashed line shows the current step of the motion cycle. The right leg uses an offset so that one foot is on the ground at all time. The arrows display the moving direction of the foot during the individual phases.

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 (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.

VIII. Conclusion and Future Work

In this paper, we showed the similarities between the two RoboCup leagues which are based on legged robots and described the possibilities for a useful software migration. Using the BreDoBrothers team as an example, we directly demonstrated the results of the implementation.

One major difference to the Four-Legged League is the unavailability of a complete robot system which does not need to be modified. In our team, this lead to two major problems which could not be solved by the software used: Reproducing a robot's state after a breakdown as well as maintaining an equal state of all robots inside a team of robots. Especially

the latter will become more important the larger the teams will become in the future.

Besides these problems and in addition to the approaches which have already been described in this paper, there are still several methods from the Four-Legged League which have not been ported to our humanoid team but will be useful in future competitions. One issue is the tracking of other robots. This has already been implemented for AIBO robots [25] but not been ported, since we did not implement a reliable perception of other robots, so far. Another big field of work—especially for 3 vs. 3 matches—will be the integration of communication. To arrange tactics (e. g. as done by [9]) as well as to establish a cooperative estimate of the current world state (e. g. as done by [26]) to enable the robots to have information about things which they currently cannot perceive.

Acknowledgments

The authors would like to thank all members of the BreDo-Brothers team as well as all members of the GermanTeam.

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

References

[1] T. Röfer, M. Fritsche, M. Hebbel, T. Kindler, T. Laue, C. Niehaus, W. Nistico, and P. Schober, "BreDoBrothers. Team Description for RoboCup 2006," in *RoboCup 2006: Robot Soccer World Cup X Preproceedings*, G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, Eds. RoboCup Federation; http://www.robocup.org/, 2006.

[2] U. Düffert, M. Jüngel, T. Laue, M. Lötzsch, M. Risler, and T. Röfer, "GermanTeam 2002," in *RoboCup 2002 Robot Soccer World Cup VI, Gal A. Kaminka, Pedro U. Lima, Raul Rojas (Eds.)*, ser. Lecture Notes in Artificial Intelligence, no. 2752. Springer, 2003, more detailed in http://www.tzi.de/kogrob/papers/GermanTeam2002.pdf.

[3] T. Röfer, I. Dahm, U. Düffert, J. Hoffmann, M. Jüngel, M. Kallnik, M. Lötzsch, M. Risler, M. Stelzer, and J. Ziegler, "GermanTeam 2003," in *RoboCup 2003: Robot Soccer World Cup VII*, ser. Lecture Notes in Artificial Intelligence, B. Browning, D. Polani, A. Bonarini, and K. Yoshida, Eds. Springer, 2004.

[4] T. Röfer, R. Brunn, I. Dahm, M. Hebbel, J. Hoffmann, M. Jüngel, T. Laue, M. Lötzsch, W. Nistico, and M. Spranger, "Germanteam 2004," in *RoboCup 2004: Robot World Cup VIII*, ser. Lecture Notes in Artificial Intelligence, no. 3276. Springer, 2005.

[5] T. Röfer, R. Brunn, S. Czarnetzki, M. Dassler, M. Hebbel, M. Jüngel, T. Kerkhof, W. Nistico, T. Oberlies, C. Rohde, M. Spranger, and C. Zarges, "Germanteam 2005," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence.

[6] S. Behnke, J. Müller, and M. Schreiber, "Using Handheld Computers To Control Humanoid Robots," in *Proceedings of 1st International Conference on Dextrous Autonomous Robots and Humanoids (darh2005), Yverdon-les-Bains - Switzerland*, May 2005, p. paper nr. 3.2.

[7] T. Röfer, "An architecture for a national robocup team." in *RoboCup 2002 Robot Soccer World Cup VI, Gal A. Kaminka, Pedro U. Lima, Raul Rojas (Eds.)*, ser. Lecture Notes in Artificial Intelligence, no. 2752. Springer, 2003, pp. 417–425.

[8] C. Mandel, K. Huebner, and T. Vierhuff, "Towards an autonomous wheelchair: Cognitive aspects in service robotics," in *Proceedings of Towards Autonomous Robotic Systems (TAROS 2005)*, 2005, pp. 165–172. [Online]. Available: http://www.taros.org.uk/

[9] T. Röfer, T. Laue, M. Weber, H.-D. Burkhard, M. Jüngel, D. Göhring, J. Hoffmann, B. Altmeyer, T. Krause, M. Spranger, U. Schwiegelshohn, M. Hebbel, W. Nisticó, S. Czarnetzki, T. Kerkhof, M. Meyer, C. Rohde, B. Schmitz, M. Wachter, T. Wegner, C. Zarges, O. von Stryk, R. Brunn, M. Dassler, M. Kunz, T. Oberlies, and M. R. and, "GermanTeam RoboCup 2005," Tech. Rep., 2005, available online: http://www.germanteam.org/GT2005.pdf.

[10] T. Laue, K. Spiess, and T. Röfer, "SimRobot - A General Physical Robot Simulator and Its Application in RoboCup," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence. Springer.

[11] J. Bach and M. Jüngel, "Using pattern matching on a flexible, horizon-aligned grid for robotic vision," *Concurrency, Specification and Programming - CSP'2002*, vol. 1, pp. 11–19, 2002.

[12] T. Röfer, T. Laue, H.-D. Burkhard, J. Hoffmann, M. Jüngel, D. Göhring, M. Lötzsch, U. Düffert, M. Spranger, B. Altmeyer, V. Goetzke, O. v. Stryk, R. Brunn, M. Dassler, M. Kunz, M. Risler, M. Stelzer, D. Thomas, S. Uhrig, U. Schwiegelshohn, I. Dahm, M. Hebbel, W. Nisticó, C. Schumann, and M. Wachter, "GermanTeam RoboCup 2004," 2004, only available online: http://www.robocup.de/germanteam/GT2004.pdf.

[13] T. Röfer and M. Jüngel, "Vision-Based Fast and Reactive Monte-Carlo Localization," *IEEE International Conference on Robotics and Automation*, 2003.

[14] T. Röfer, T. Laue, and D. Thomas, "Particle-filter-based self-localization using landmarks and directed lines," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence. Springer.

[15] S. Lenser and M. Veloso, "Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision," in *Proceedings of IROS'03*, 2003.

[16] J. Hoffmann, M. Jüngel, and M. Lötzsch, "A vision based system for goal-directed obstacle avoidance," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Artificial Intelligence, no. 3276. Springer, 2005.

[17] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *Proc. of the National Conference on Artificial Intelligence*, 1999.

[18] T. Röfer and M. Jüngel, "Vision-based fast and reactive monte-carlo localization," in *IEEE International Conference on Robotics and Automation*. Taipei, Taiwan: IEEE, 2003, pp. 856–861.

[19] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[21] M. Lötzsch, J. Bach, H.-D. Burkhard, and M. Jüngel, "Designing agent behavior with the extensible agent behavior specification language XABSL," in *RoboCup 2003: Robot Soccer World Cup VII*, ser. Lecture Notes in Artificial Intelligence, D. Polani, B. Browning, A. Bonarini, and K. Yoshida, Eds., no. 3020. Padova, Italy: Springer, 2004.

[22] R. Lafrenz, O. Zweigle, U.-P. Käppeler, H. Rajaie, A. Tamke, T. Rühr, M. Oubbati, M. Schanz, F. Schreiber, and P. Levi, "Major Scientific Achievements 2006 - CoPS Stuttgart registering for world champi-onships in Bremen," in *RoboCup 2006: Robot Soccer World Cup X Preproceedings*, G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, Eds. RoboCup Federation; http://www.robocup.org/, 2006.

[23] O. Birbach, A. Burchardt, C. Elfers, T. Laue, F. Penquitt, T. Schindler, and K. Stoye, "B-Smart. Team Description for RoboCup 2006," in *RoboCup 2006: Robot Soccer World Cup X Preproceedings*, G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, Eds. RoboCup Federation; http://www.robocup.org/, 2006.

[24] T. Röfer, T. Laue, M. Weber, H.-D. Burkhard, M. Jüngel, D. Göhring, J. Hoffmann, B. Altmeyer, T. Krause, M. Spranger, O. v. Stryk, R. Brunn, M. Dassler, M. Kunz, T. Oberlies, M. Risler, U. Schwiegelshohn, M. Hebbel, W. Nisticó, S. Czarnetzki, T. Kerkhof, M. Meyer, C. Rohde, B. Schmitz, M. Wachter, T. Wegner, and C. Zarges, "GermanTeam RoboCup 2005," 2005, http://www.germanteam.org/GT2005.pdf.

[25] T. Laue and T. Röfer, "Integrating simple unreliable perceptions for accurate robot modeling in the four-legged league," in *RoboCup 2006: Robot Soccer World Cup X*, ser. Lecture Notes in Artificial Intelligence, G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, Eds. Springer; http://www.springer.de/.

[26] W. Nistico, M. Hebbel, T. Kerkhof, and C. Zarges, "Cooperative visual tracking in a team of autonomous mobile robots," in *RoboCup 2006: Robot Soccer World Cup X*, ser. Lecture Notes in Artificial Intelligence, G. Lakemeyer, E. Sklar, D. Sorrenti, and T. Takahashi, Eds. Springer; http://www.springer.de/.