International Journal of Humanoid Robotics
Vol. 5, No. 3 (2008) 375–396
© World Scientific Publishing Company

HIERARCHICAL REACTIVE CONTROL FOR HUMANOID SOCCER ROBOTS

SVEN BEHNKE* and JÖRG STÜCKLER†

University of Bonn, Institute for Computer Science VI,
Autonomous Intelligent Systems, Römerstr. 164,
53117 Bonn, Germany
**behnke@ais.uni-bonn.de*
†stueckle@ais.uni-bonn.de

Received 30 July 2007
Accepted 30 January 2008

Humanoid soccer serves as a benchmark problem for artificial intelligence research and robotics. Every year, more teams are competing, for example, in the RoboCup Humanoid League. As the robots improve in the basic skills of walking, kicking, and getting up, teams can focus on soccer skills and team coordination. The complexity of soccer behaviors and team play calls for structured behavior engineering.

In this paper, we describe the design of the behavior control software for the Humanoid League team NimbRo. The software is based on a framework that supports a hierarchy of reactive behaviors. It is structured both as an agent hierarchy (joint–body part–player–team) and as a time hierarchy. The speed of sensors, behaviors, and actuators decreases when moving up in the hierarchy. The lowest levels of this framework contain position control of individual joints and kinematic interfaces for body parts. At the next level, basic skills are implemented. Soccer behaviors like searching for the ball, approaching the ball, avoiding obstacles, and defending the goal reside on the player level. Finally, on the tactical level, the robots communicate via a wireless network to negotiate roles and use allocentric information to configure the soccer behaviors.

Our robots performed very well at RoboCup 2007, which took place in Atlanta, Georgia, USA. They won both the KidSize and the TeenSize humanoid soccer competitions.

Keywords: Reactive control; time hierarchy; agent hierarchy; RoboCup.

1. Introduction

Humanoid soccer serves as a benchmark problem for artificial intelligence research and robotics. The RoboCup Federation and FIRA organize international robotic soccer competitions. The long-term goal of RoboCup is to develop, by the year 2050, a team of humanoid soccer robots that wins against the FIFA world champion.[1] The game of soccer was selected for the competitions because, as opposed to chess, multiple players of one team must cooperate in a dynamic environment. Sensory signals must be interpreted in real time to take appropriate actions. The soccer competitions do not test isolated components, but two systems compete with each other. The number of goals scored is an objective performance measure that allows comparison of systems that implement a large variety of approaches to perception, behavior control, and robot construction. The presence of opponent teams, which continuously improve their system, makes the problem harder every year. Such a challenge problem focuses the effort of many research groups worldwide and facilitates the exchange of ideas.

The RoboCup championships grew to be the most important robotic competition worldwide. In the last RoboCup, which took place in July 2007 in Atlanta, Georgia, USA, 321 teams from 39 countries competed. The total number of participants was about 2000.

In the RoboCup Humanoid League, fully autonomous robots with a human-like body plan compete with each other. The robots must have two legs, two arms, a head, and a trunk. Size restrictions make sure that the center of mass of the robots is not too low, that the feet are not too large, and so on. The robots are grouped into two size classes: KidSize (<60 cm) and TeenSize (80–160 cm).

The RoboCup Humanoid League was established in 2002 and has developed quickly since. It is now the league with the largest number of participants. In Atlanta, 29 teams from 14 countries competed in the Humanoid League. Some of the participating robots are shown in Fig. 1. As the humanoid soccer robots improve in the basic skills of walking, kicking, and getting up, the research teams start to focus on soccer skills and on the coordination of the individual players.

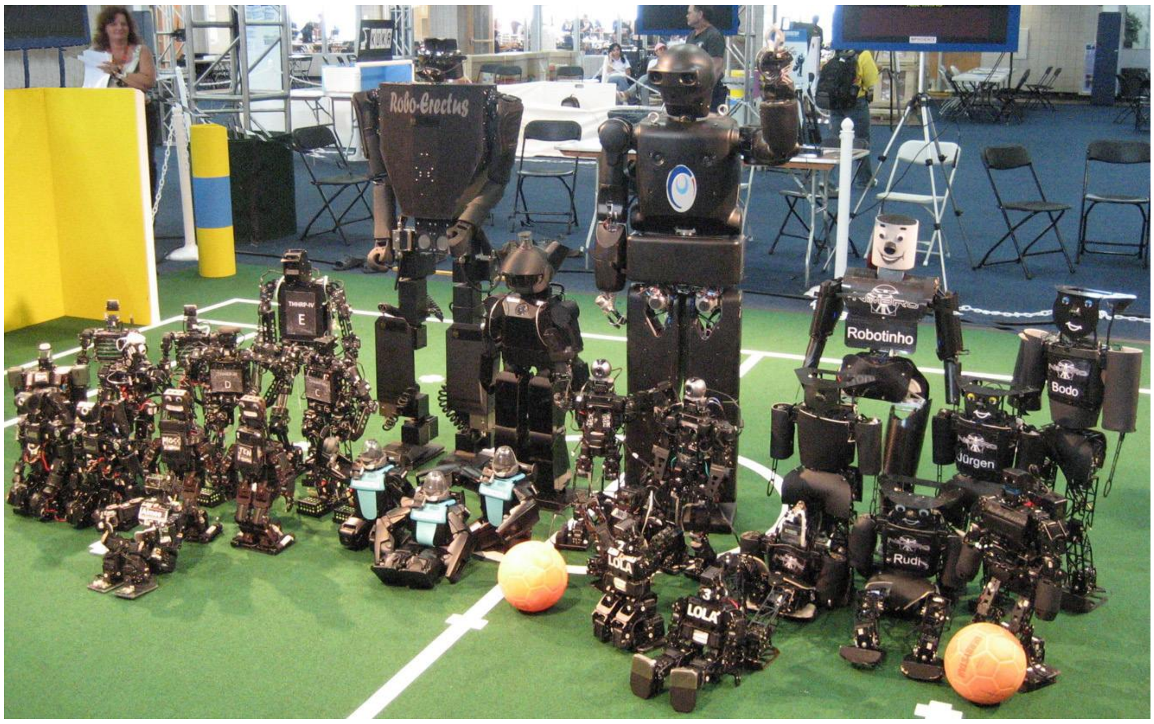


Fig. 1. Some of the humanoid robots that competed at RoboCup 2007.

Playing soccer is not a trivial task. The ball might be at any position on the field and the robots need to search for it if they lose track of its position. The robots must perceive at least two goals and the other players. Higher-level behaviors require self-localization on the field. As two robots play together, there is need for coordination. While some teams use one dedicated goalie and one field player, other teams use two field players. This makes dynamic role assignment necessary. Last but not least, in soccer games robots of the two teams interact physically when going for the ball. This disturbs walking and can lead to falls. Then, the robots need to get up from the ground by themselves in order to continue play. As a result of these difficulties, only a fraction of the participating teams were able to play decent soccer games.

Our approach to coping with these demands was to extend a framework that supports a hierarchy of reactive behaviors[2] in order to implement the behavior control software for the humanoid soccer robots of our team, NimbRo. This framework was originally developed for the FU-Fighters SmallSize robots. It was later adapted to the FU-Fighters MiddleSize robots and also used by CMU in the Four-Legged League.[3] A graphical tool for behavior engineering, which is based on this framework, has been developed at FU Berlin.[4] Such a structured approach to behavior engineering is essential for managing the complexity of soccer games.

We adapted the behavior framework of the FU-Fighters for the control of soccer-playing humanoid robots by extending the agent hierarchy to: joint–body part–player–team. The lowest levels of this hierarchy contain position control of individual joints and kinematic interfaces for body parts. At the next level, basic skills like omnidirectional walking, kicking, and getting-up behaviors are implemented. These are used on the player level by soccer behaviors like searching for the ball, approaching the ball, avoiding obstacles, and defending the goal. Finally, on the tactical level, the robots communicate via a wireless network to negotiate roles and use allocentric information to configure the soccer behaviors. We also introduced master and slave behaviors for the simultaneous activation of behaviors across multiple body parts.

The remainder of this paper is organized as follows. After reviewing some of the related work, we describe the mechanical and electrical design of our KidSize and TeenSize 2007 robots in Sec. 3. Proprioception and visual perception of the game situation are detailed in Sec. 4. Section 5 introduces our behavior control framework and proposes some extensions for its application on a team of humanoid soccer robots. The implementation of basic skills is covered in Sec. 6. The design of our soccer behaviors and their use on the allocentric tactical level are described in Secs. 7 and 8, respectively. Finally, we present the results of using the proposed system at RoboCup 2007.

2. Related Work

The other RoboCupSoccer leagues have been facing the complexity of soccer games for some years now. There, tools for structured behavior engineering have been developed. For example, Jaeger and Christaller proposed the Dual Dynamics architecture, which has been used in the MiddleSize League.[5] The architecture distinguishes elementary behaviors, which implement a target dynamics, and complex behaviors, which control the activation of elementary behaviors. The Dual Dynamics approach has been combined with a planner in the DD&P framework by Hertzberg and Schönherr.[6] The DD-Designer environment, developed at Fraunhofer IAIS, also supports team coordination.[7]

Another tool used in the MiddleSize League is the BAP framework of Utz *et al.*, which allows for specifying hierarchical, event-driven, behavior-based control systems.[8] In the Four-Legged League, the German Team developed XABSL.[9] This allows for XML-based specification of hierarchies of behavior modules that contain state machines for decision-making. State transitions are modeled as decision trees. Parts of the German Team system are used now in the Humanoid League by Darmstadt Dribblers, Humanoid Team Humboldt, B-Human, and DoH!Bots.

Another example of a behavior architecture used in more than one league is the architecture proposed by Laue and Röfer, which combines action selection and potential field motion planning.[10] It was used to control SmallSize and Aibo soccer robots.

A planning-based approach to behavior design that relies on petri nets for synchronization has been proposed by Ziparo and Iocchi.[11] It is used in both the SPQR legged and the rescue teams. Another example of a deliberative system is Readylog, based on Golog, which is used by the team AllemaniACs in several leagues.[12] A system that combines reactive execution with deliberation in a behavior hierachy is the double pass architecture used in the ATHumboldt simulation team.[13]

For the Sony QRIO humanoid robot, a control architecture has been proposed by Fujita *et al.* that executes multiple situated behaviors in parallel and contains memories.[14] Based on this EGO architecture, Chernova and Arkin recently proposed a cognitively inspired action selection mechanism that shifts repeatedly executed routine tasks from a deliberative level to a reactive behavior level.[15]

Several less specialized development environments for autonomous mobile robots are available to support behavior engineering in more general applications than RoboCup. A recent survey has been compiled by Kramer and Scheutz.[16]

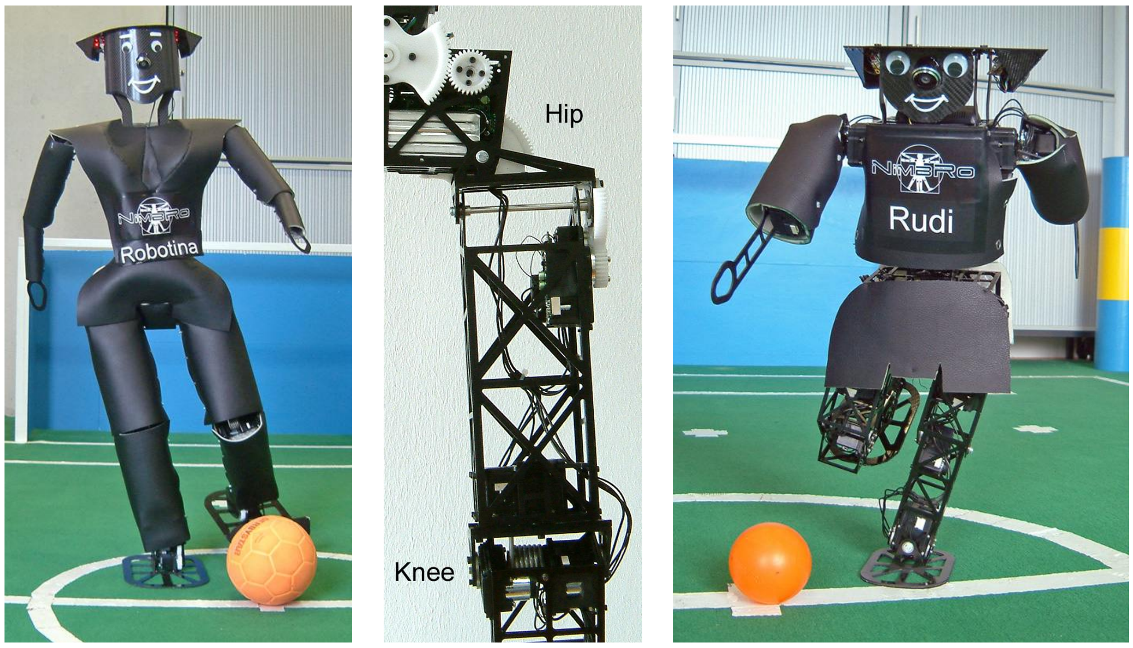
Our architecture for hierarchical reactive behavior control builds on Dual Dynamics, but extends it to multiple levels in the time hierarchy and organizes behaviors in an agent hierarchy. This modular approach facilitates the structured development and maintenance of behaviors. Flexible inhibition mechanisms allow, for example, for smooth transitions between behaviors or for hysteresis of behavior activation. Multiple behaviors are activated concurrently in the different levels and agents of the hierarchy. Even within a behavior layer, multiple behaviors can be active simultaneously.

3. NimbRo 2007 Robots

We constructed five new robots for RoboCup 2007: Rudi, Jürgen, and Lothar play in the KidSize class; Bodo is the TeenSize goalie and Robotina is the TeenSize penalty kick striker. Figure 2 shows Robotina and Rudi. As can be seen, the robots have human-like proportions. Their mechanical design is focused on simplicity, robustness, and weight reduction.

3.1. *Main computer and camera system*

Compared to the NimbRo 2006 robots, which were controlled by a Pocket PC, the 2007 robots have a more powerful main computer and a high-bandwidth vision system that has a 360° field of view. The new robots are controlled by a handheld PC, a Sony Vaio UX, which features an Intel 1.33 GHz ULV Core Solo Processor, 1 GB RAM, 32 GB SSD, a 4.5″ WSVGA touch-sensitive display, 802.11a/b/g WLAN, and a USB2.0 interface. The weight of the UX is only 486 g. Three IDS uEye UI-1226LE industrial USB2.0 cameras provide omnidirectional sight. The cameras feature a 1/3″ WVGA CMOS sensor, a global shutter, and are equipped with ultra-wide angle lenses. Each camera has a 140° × 90° field of view. In order to cover all directions, the cameras are directed to the front (0°) and to the left/right rear (±120°).



(a) (b) (c)

Fig. 2. NimbRo 2007: (a) TeenSize robot Robotina; (b) upper left leg of Robotina without cover; visible are, for example, the external spur gears in the hip and the knee spring; (c) KidSize robot Rudi.

3.2. *Mechanical design*

The NimbRo 2007 robots also have stronger actuators, compared to the NimbRo 2006 robots. The KidSize robots Rudi, Jürgen and Lothar are 60 cm tall and each has a total weight of 4 kg. The robot Bodo, used mainly as TeenSize goalie, is 83 cm tall. These four robots are driven by 20 Dynamixel actuators: six per leg, three in each arm, and two in the trunk. For all leg joints, except the hip yaw, and for the trunk pitch joint, we use large RX-64 actuators (116 g, 64 kg · cm). All other joints are driven by smaller DX-117 actuators (66 g, 37 kg · cm).

The TeenSize robot Robotina is 122 cm tall and has a total weight of about 8.75 kg. Its 21 DOFs are driven by a total of 33 Dynamixel actuators. The additional joint is the roll axis in the trunk. All joints in the legs and the trunk, except for the yaw axes, are driven by two parallel actuators. The actuators are coupled in a master–slave configuration, which doubles the torque and lowers operating temperatures. A master–slave pair of actuators has the same interface as the single actuators used for all other joints. Dynamixel RX-64 actuators are used in the legs and DX-117 actuators are used in the trunk and the arms. The ankle, hip, and trunk yaw/roll axes are reinforced by external 2:1/3:1 spur gears, respectively, resulting in a holding torque of 384 kg · cm (39 Nm) in the ankle and hip roll joints. Figure 2(b) shows parts of Robotina's mechanical structure without cover. The knee is not reduced with an external spur gear, because it needs to move quickly. Instead, a torsional spring is added in parallel to the knee actuators. This spring supports stretching the knee and is designed to compensate for the robot's weight when it is standing with partially bent knees.

The skeleton of all five robots is constructed from aluminum extrusions with a rectangular tube cross section. In order to reduce weight, we removed all material not necessary for stability. The feet, the forearms, and the robot heads are made from sheets of carbon composite material. The upper part of the smaller robots and the entire body of Robotina are protected by a layer of foam and an outer shell of synthetic leather.

3.3. *Electrical design*

Our soccer robots are fully autonomous. They are powered by high-current lithium-polymer rechargeable batteries located in their hip. Five Kokam 1250 mAh cells are used for the smaller robots. Robotina has five Kokam 3200 mAh cells. The batteries last for about 25 min of operation.

The Dynamixel actuators have an RS-485 differential half-duplex interface. Each robot is equipped with a CardS12X microcontroller board, which manages the detailed communication with the Dynamixel actuators. These boards feature the Motorola MC9S12XDP512 chip, a 16-bit controller belonging to the popular HCS12X family. The controller has an I/O coprocessor and many interfaces, including serial lines and A/D converters. The Dynamixel actuators have a flexible interface. Not only are target positions sent to the actuators, but also parameters of the control loop, such as the compliance. In the opposite direction, the current positions, speeds, loads, temperatures, and voltages are read back.

In addition to these joint sensors, the robots are equipped with an attitude sensor, located in the trunk. It consists of a dual-axis accelerometer (ADXL203, ± 1.5 g) and two gyroscopes (ADXRS, $\pm 300°$/s). The four analog sensor signals are digitized with A/D converters of HCS12X and are preprocessed by the microcontroller. The microcontroller communicates with the Dynamixels via RS-485 at 1 MBaud and with a main computer via an RS-232 serial line at 115 KBaud. Every 12 ms, target positions and compliances for the actuators are sent from the main computer to the microcontroller board, which distributes them to the actuators. The microcontroller sends the preprocessed sensor readings back. This allows one to keep track of the robot's state in the main computer.

4. Perception

Our robots need information about themselves and the situation on the soccer field in order to act successfully.

4.1. *Proprioception*

The readings of accelerometers and gyros are fused to estimate the robot's tilt in the roll and pitch direction. The gyro bias is calibrated automatically, and the low-frequency components of the tilt estimated from the accelerometers are combined with the integrated turning rates to yield an estimate of the robot's attitude that is insensitive to short linear accelerations. As described above, joint angles, speeds, and loads are also available. Temperatures and voltages are monitored to notify the user in case of overheating or low batteries.

4.2. *Visual object detection*

The only information sources for our robots about their environment are three cameras. Our computer vision software captures and interprets images with 752×480 pixels at an aggregated frame rate of about 32 fps. The camera's wide field of view allows the robots to see objects close to their own feet and objects above the horizon in all directions at the same time.

Figure 3 shows three camera images that have been captured simultaneously with marked objects. Our computer vision software detects the ball, the goals, the corner poles, and other players based on their color in YUV space. Using a look-up table, the colors of individual pixels are grouped into color classes that are described by ellipsoids in the UV plane. In a multistage process, we discard insignificant colored pixels and detect colored objects. The software also detects the goalposts, the goal area not covered by the goalie, and the white field markers.

We estimate the coordinates of detected objects in an egocentric frame (the distance to the robot and the angle to its orientation), based on the inverted projective

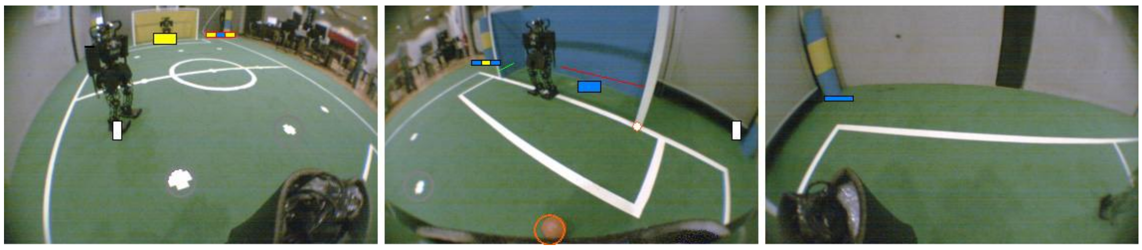


Fig. 3. Images captured by the three cameras. The detected objects are marked: goals (blue and yellow rectangle), ball (orange circle), field markers (gray circles), corner poles (horizontal blue and yellow lines), goalpost (white circle), and obstacles (white vertical rectangles). The free goal area is marked on the left and the right of the goalie with a green and a red line, respectively. Color figures are only available in electronic version.

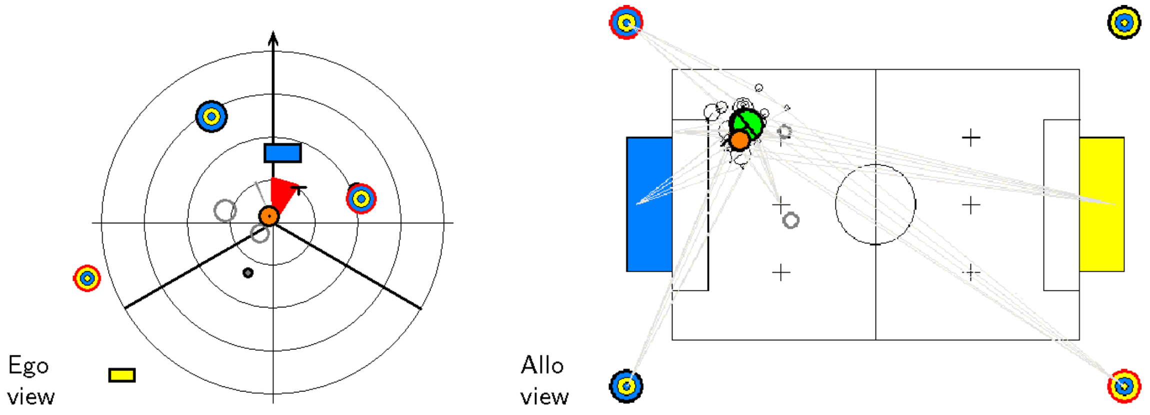


Fig. 4. Egocentric and allocentric representations of the game situation constructed by the computer vision module.

function of the camera. We first correct for the lens distortion and next invert the affine projection from the ground plane to the camera plane. The estimated egocentric coordinates of the key objects are illustrated in the left part of Fig. 4. Here, the objects detected by the three cameras are fused based on their confidence. They are also merged with previous observations, which are adjusted by a motion model, if the robot is moving. This yields a robust egocentric world representation.

4.3. *Self-localization*

The relative coordinates suffice for many relative behaviors, like positioning behind the ball while facing the goal. To keep track of nonvisible goals or to communicate about the ball with other team members, we need the robot coordinates in an allocentric frame [(x, y) position on the field and orientation θ]. We solve self-localization by triangulation over pairs of landmark observations, i.e. detected goals, goalposts, corner poles, and field markers. When we are observing more than two landmarks, the triangulation results are fused based on their confidence. We apply a mean shift procedure to exclude outliers from the final pose estimate. Again, the results of self-localization are integrated over time and a motion model is applied. The right part of Fig. 4 illustrates the resulting allocentric representation.

5. Behavior Architecture

For the behavior control of a team of humanoid soccer robots, we extended a framework that supports a hierarchy of reactive behaviors.[2] This framework was inspired by the Dual Dynamics scheme, developed by Jäger.[5] One of the central ideas of Dual Dynamics is that multiple behaviors are executed concurrently within a layer. An activation dynamics decides whether or not a behavior is allowed to influence the actuators. If a behavior becomes active, its target dynamics decides how it influences the actuators.

Figure 5 illustrates the architecture of our hierarchical framework. In contrast to the original, two-level approach, in our system multiple layers, each containing behaviors of different complexity, run on different time scales. When moving up the hierarchy, the speed of sensors, behaviors, and actuators decreases. At the same time, they become more abstract. On the lowest level, we have few simple and fast behaviors. While the speed of the behaviors decreases when moving up the hierarchy, their number, as well as the number of sensors and actuators, increases. This allows one to model complex systems.

The left column of Fig. 5 shows the sensors, which represent the state of the perceptual dynamics. The boxes shown in the figure are divided into cells, where each cell represents a sensor value that is constant for a time step. The rows correspond to different sensors and the columns show the time advancing from left to right. The sensors have different degrees of abstraction and different time scales. On

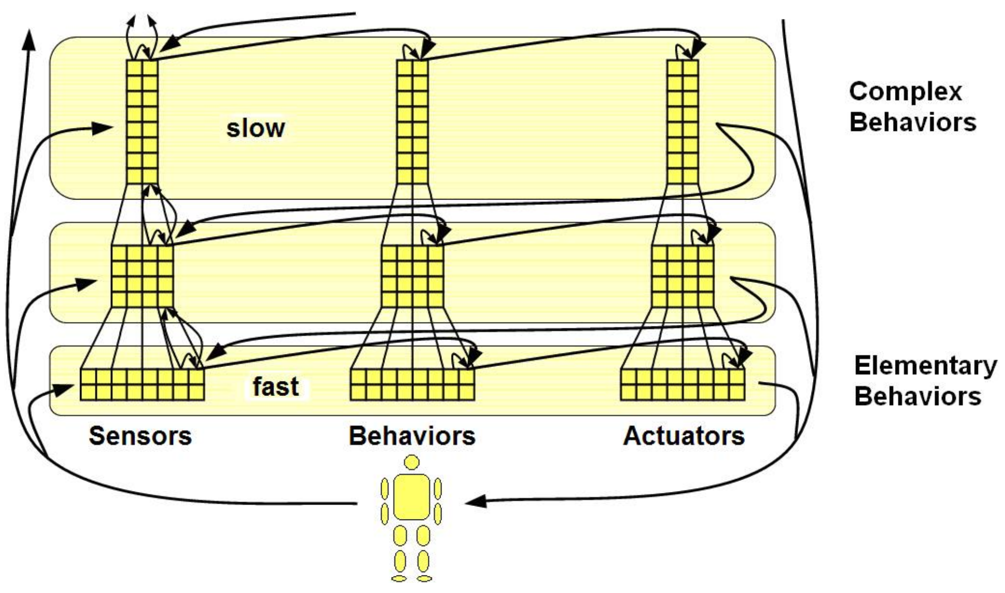


Fig. 5. Sketch of the hierarchical reactive control architecture.

the lowest level, quickly changing physical sensors are connected. On higher levels, sensory information comes from three sources:

- They can be computed from lower-level sensors that are aggregated in time and may become more abstract.
- They can be connected to physical sensors of an appropriate time scale, or
- They mirror higher-level actuators (described on the next page).

The horizontal arrows from one time step to the next indicate that the perceptual dynamics depends on previous sensor values within the same level.

The framework forces the behavior engineers to define abstract sensors that are aggregated from faster, more basic sensors. One example of such an abstract sensor is the robot's attitude which is computed from the readings of accelerometers and gyros. Each physical sensor can be connected to only one level of the hierarchy. One can use the typical rate of the change of a sensor's readings to decide where to connect it. One example of a slowly changing physical sensor is the battery voltage.

The central column of Fig. 5 represents the state of the behavior activation dynamics. Each level contains multiple behaviors. An activation factor between 0 and 1 determines when the corresponding behavior is allowed to influence actuators. Activation depends on the current sensory state within the layer, but also on previous activations and on the activation of conflicting behaviors. This allows one to design various activation patterns, such as hysteresis, smooth transitions, or chaining of behaviors.

The right column of Fig. 5 represents the state of the actuator dynamics. The actuators have different degrees of abstraction. The actuators at the lowest level are connected to quickly changing robot actuators. They determine, for example, joint target positions, speeds, or joint control loop parameters. Higher-level actuators are either connected to slow physical actuators or mirrored at the next lower level as sensors. Such abstract actuators give higher-level behaviors the possibility to configure lower layers in order to eventually influence the state of the world. One example of an abstract actuator is the desired walking speed, which configures the gait engine, described below, implemented at the lower control levels.

Since we use temporal subsampling, we can afford to implement an increasing number of sensors, behaviors, and actuators in the higher layers without an explosion of computational cost.

5.1. *Computation of the dynamics within a layer*

The dynamical system of the sensors, behaviors, and actuators can be specified and analyzed as a set of differential equations. The actual computations, however, are implemented using difference equations. Here, the time runs in discrete steps of $\Delta t^0 = t_i^0 - t_{i-1}^0$ at the lowest level, 0. At the higher levels, the updates are done less frequently: $\Delta t^z = t_i^z - t_{i-1}^z = f \Delta t^{z-1}$, where useful choices of the subsampling factor f could be $2, 4, 8, \ldots$. In Fig. 5, $f = 2$ was used.

A layer z is updated in time step t_i^z as follows:

$\mathbf{s}_i^z$ — *sensor values*:
The N_s^z sensor values $\mathbf{s}_i^z = (s_{i,0}^z, s_{i,1}^z, \ldots, s_{i,N_s^z-1}^z)$ depend on the readings of the N_r^z physical sensors $\mathbf{r}_i^z = (r_{i,0}^z, r_{i,1}^z, \ldots, r_{i,N_r^z-1}^z)$, which are connected to layer z, the previous sensor values $\mathbf{s}_{i-1}^z$, and the previous sensor values from the layer below, $\mathbf{s}_{fi}^{z-1}, \mathbf{s}_{fi-1}^{z-1}, \mathbf{s}_{fi-2}^{z-1}, \ldots$.

In order to avoid the storage of old values at the lower level, the sensor values can be updated from the layer below, e.g. as moving average.

By analyzing the sensor values from the last few time steps, one can also compute predictions for the next few steps that are needed for anticipative behavior. If the predictions in addition take the last few actuator values into account, they can be used to cancel a delay between an action command and the perceived results of that action.

A dedicated set of sensors mirrors the actuator values $\mathbf{a}_{i/f}^{z+1}$ of the next higher layer $(z+1)$ in order to make them accessible for layer z.

α_i^z — *activation factors*:
Within a layer z, behavior activation is computed in two phases. First, the N_α^z behaviors compute a desired activation, $\hat{\boldsymbol{\alpha}}_i^z = (\hat{\alpha}_{i,0}^z, \hat{\alpha}_{i,1}^z, \ldots, \hat{\alpha}_{i,N_\alpha^z-1}^z)$, based on the sensor values $\mathbf{s}_i^z$ and the previous activations α_{i-1}^z. In the second phase, an inhibition mechanism ensures that higher-priority behaviors take precedence over lower-priority ones. Priority is encoded as cyclic inhibition graph within a layer. The desired activation is decreased by the desired activation of higher-priority behaviors to compute the behavior activation: $\alpha_{i,j}^z = \mathsf{argmin}_{k \text{ inhibits } j} \max(0, \hat{\alpha}_{i,j}^z - \hat{\alpha}_{i,k}^z)$. Hence, partially active higher-priority behaviors will only partially inhibit lower-priority behaviors. This facilitates smooth transitions between behaviors.

$\mathbf{G}_i^z$ — *target values*:
Each behavior j can specify for each actuator k within its layer z a target value, $g_{i,j,k}^z = G_{j,k}^z(\mathbf{s}_i^z)$. These target values frequently implement simple control laws.

$\mathbf{a}_i^z$ — *actuator values*:
The more active a behavior j is, the more it can influence the actuator values $\mathbf{a}_i^z = (a_{i,0}^z, a_{i,1}^z, \ldots, a_{i,N_a^z-1}^z)$. The desired change for the actuator value $a_{i,k}^z$ is $u_{i,j,k}^z = \tau_{i,j,k}^z \alpha_{i,j}^z (g_{i,j,k}^z - a_{i-1,k}^z)$, where $\tau_{i,j,k}^z$ is a time constant. If several behaviors want to change the same actuator k, the desired updates are added: $a_{i,k}^z = a_{i-1,k}^z + u_{i,j_0,k}^z + u_{i,j_1,k}^z + u_{i,j_2,k}^z + \ldots$.

Hence, non-conflicting behaviors can access the same actuator simultaneously.

5.2. *Agent hierarchy*

The control hierarchy of our soccer robots is arranged in an agent hierarchy, where

- multiple joints (e.g. left knee) constitute a body part (e.g. left leg),
- multiple body parts constitute a player (e.g. field player), and
- multiple players constitute a team.

The topmost layers of the lower-level agents are connected to the lowest layer of the agent at the next level of this hierarchy. In order to ensure simultaneous activation of lower-level behaviors across multiple body parts, we introduced master and slave behaviors. While the master behavior, e.g. in the trunk body part, computes its activation as described above, slave behaviors in the arms and the legs do not compute their own activation, but use that of their master behavior.

5.3. *Bottom-up design*

Behaviors are constructed in a bottom-up fashion. First, the control loops that should react quickly to fast-changing stimuli are designed. Their critical parameters, e.g. a mode parameter or a target position, are determined. When these fast primitive behaviors work reliably with constant parameters, the next level can be added to the system. For this higher level, more complex behaviors can now be designed which influence the environment — either directly, by moving slow actuators, or indirectly, by changing the critical parameters of the control loops at the lower level.

After the addition of several layers, fairly complex behaviors can be designed that make decisions using abstract sensors based on a long history, and that use powerful abstract actuators to influence the environment. In the following, we describe such a behavior hierarchy for the control of a team of soccer-playing humanoid robots.

6. Basic Skills

Fundamental prerequisites for playing soccer are the ability to walk and to kick. As body contact between the physical agents is unavoidable, the capability of getting up after a fall is also essential. To act as a goalkeeper, the robot must be able to perform special motions. These basic skills are implemented on the body part layer through behaviors which generate target positions for individual joints at a rate of 83.3 Hz. To abstract from the individual joints, we have implemented here a kinematic interface for the body parts.[17] It allows one, for example, to specify the leg angle and the angle of the foot plate, both relative to the trunk, and the leg extension.

In the following, we will give a brief description of each basic skill:

- *Omnidirectional walking.* The ability to move in any direction, irrespective of the orientation, and to control the rotational speed at the same time has advantages in many domains, including RoboCupSoccer. Omnidirectional drives are used by most teams in the wheeled leagues, and omnidirectional walking is heavily used in the Four-legged League. It is much easier to position robots for kicking and to outmaneuver opponents when using omnidirectional locomotion.
  The omnidirectional gait of our humanoid soccer robots is achieved by generating walking patterns online.[17] Translational and rotational walking speeds can be set to values within continuous ranges for each direction. Besides speed limits, there are no restrictions on the combination of walking speeds. The gait target

vector (v_x, v_y, v_θ) can be changed continuously while the robot is walking. The key ingredients of the omnidirectional gait are shifting the weight from one leg to the other, shortening of the leg not needed for support, and leg motion in the walking direction. By controlling the foot angles in dependence of the trunk angular velocity, we were able to enhance the robustness and speed range of the gait significantly. During walking it is also possible to twist the trunk. Higher-level behaviors can specify this rotation through a gaze orientation actuator.

- *Kicking.* Our robots are able to perform a parametrizable kick motion to the front. Figure 6 shows a snapshot sequence of Robotina's kicking motion. An actuator allows behaviors at the upper level to trigger the kick with both the left and the right leg. The kick strength is also configurable through an actuator. To correct for smaller angular misalignments between the robot and the kick target, the kick angle is adjusted by rotating the kicking leg in the yaw direction.
- *Getting up after a fall.* Since in soccer games physical contact between the robots is unavoidable, the walking patterns are disturbed and the robots might fall. Using their attitude sensors, the robots detect a fall, relax their joints before impact, classify the prone or supine posture, and trigger the corresponding getting-up sequence. We designed the getting-up sequences in a physics-based simulator using sinusoidal trajectories.[18] Figure 7 shows Rudi getting up from the supine posture. The getting-up sequences work very reliably. Under normal circumstances, i.e. appropriate battery voltage, the routines worked with 100 successes in 100 tests.
- *Goalkeeper motions.* The goalkeeper is capable of diving in both directions or bending forward with spread arms. Figure 8 shows our TeenSize goalie Bodo diving during the RoboCup 2007 Penalty Kick competition.

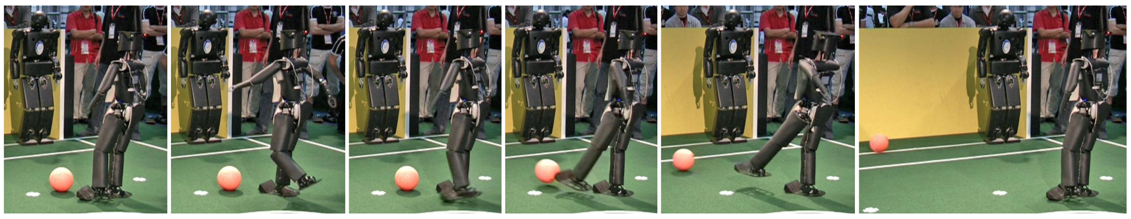


Fig. 6. Kicking motion of Robotina during the RoboCup 2007 Penalty Kick competition.

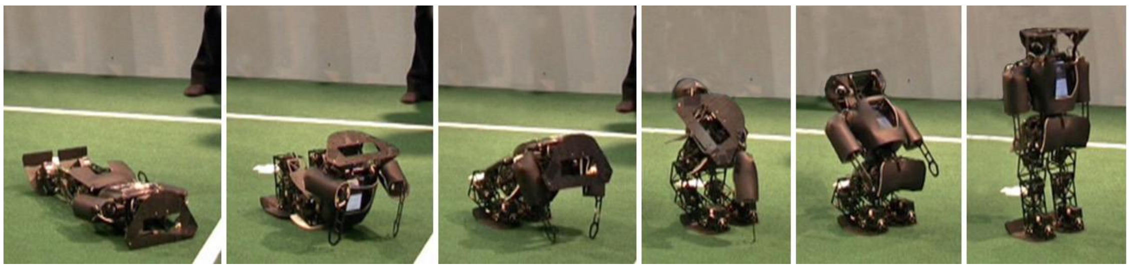


Fig. 7. Rudi getting up from the supine posture during a RoboCup 2007 soccer game.

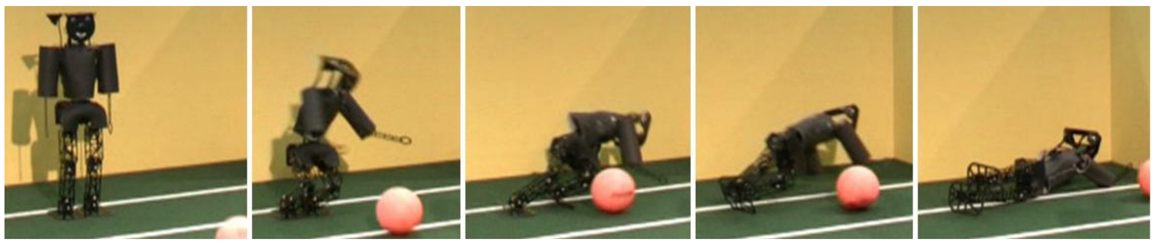


Fig. 8. Diving motion of the goalkeeper Bodo during the RoboCup 2007 Penalty Kick competition.

Some of the basic skills can be configured through actuators from the next higher level in our behavior control system. This makes abstraction from joint trajectory generation possible when one is implementing soccer-specific tasks. Figure 9 illustrates the inhibitory structure of the basic skills and the interface that they provide for the next behavior level.

7. Soccer Behaviors

The next higher level of our behavior control framework contains soccer behaviors which are executed at a rate of 41.7 Hz. They build on the basic skills and have been designed for 2 vs. 2 soccer games.

An essential part of this level is an adequate representation of the current game situation. The visual perception supplies relative distance, angle, and perceptual confidence for the ball, the own goal, the opponent's goal, the center angle of the largest goal area not covered by the goalie, and the nearest obstacle. In the offensive role, the relative position and confidence of the largest free area in the opponent goal is used as the target to kick at (ball-target), if it is perceived confidently. Otherwise, the opponent's goal is used as ball-target. To kick the ball, the robot has to position itself with its kicking leg behind the ball. Thus, the corresponding relative target position of the robot, denoted as the behind-ball position, and the kicking leg are contained in the game state. The decision for the kicking leg is made at every time step, depending on the relative position of the ball and the line from ball to ball-target. If the robot has to approach the ball-to-target line from the right, it kicks with the left leg, and vice versa. To avoid oscillations, decisions are only made as long as the distance of the robot to the ball-to-target line exceeds a threshold.

When playing as a defensive field player, the own goal is used as ball-target, such that the position behind the ball is set to a defensive position between the ball and the own goal. The components of the current game state are provided to the soccer behaviors through sensors. Figure 10 illustrates the positioning of the offensive and the defensive field player with an example.

The robot also maintains hypotheses about the relative ball location that are used for searching the ball. If a kick is triggered, one hypothesis is set to the predicted end position of the ball, as due to limitations in the visibility range the

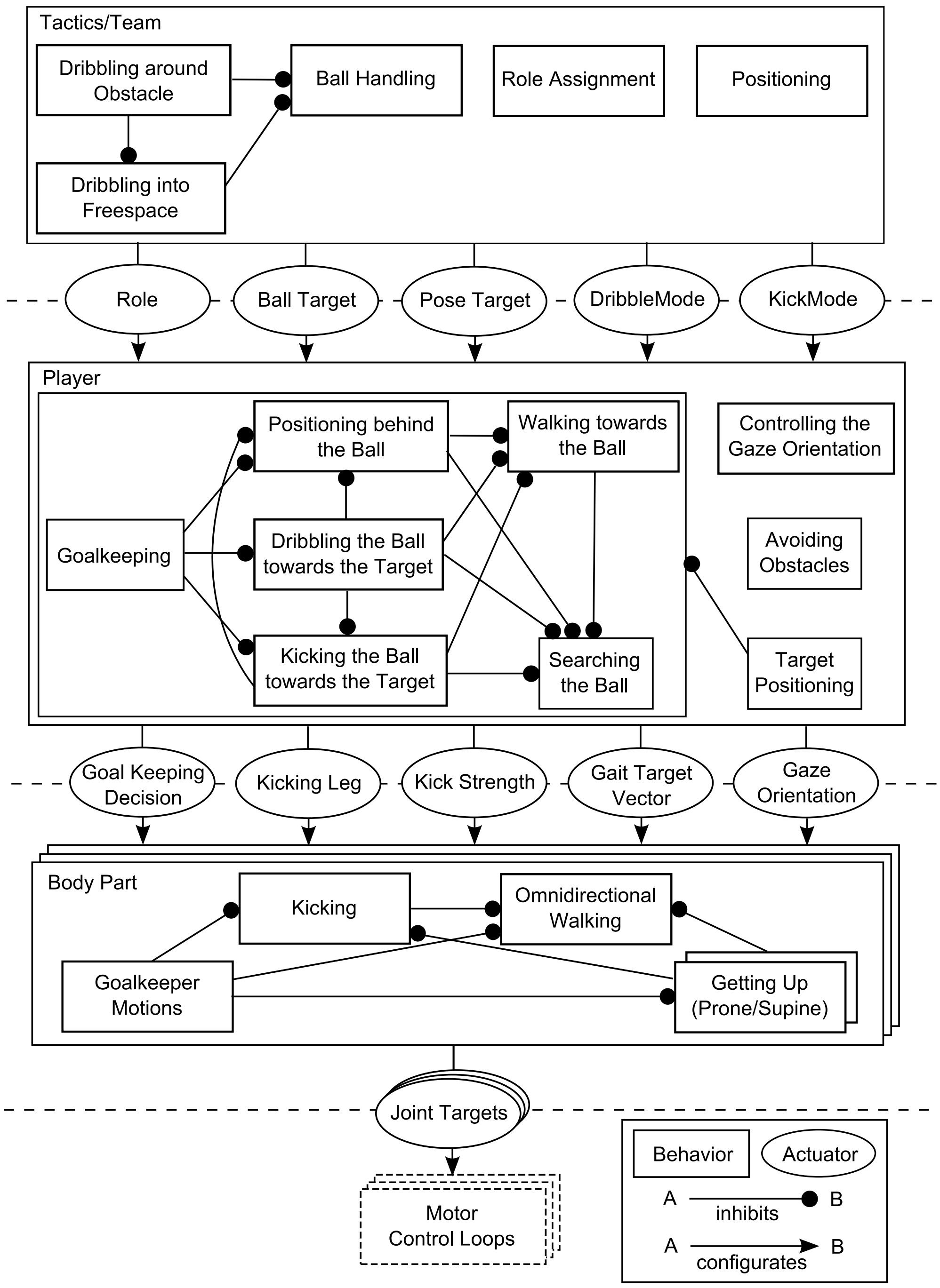


Fig. 9. Sketch of the behavior hierarchy for playing soccer. Complex behaviors at the upper levels are evaluated less often than the behaviors at the lower levels. The complex behaviors configure the behaviors at the next lower level through abstract actuators. They are activated according to abstract sensors and the inhibitory structure within a layer.

visual perception could fail to detect the ball. Additionally, hypotheses are maintained for the perceptions of the ball communicated by other players of the team. The confidences in these hypotheses depend on the self-localization and ball perception confidences of the other players and the self-localization confidence of

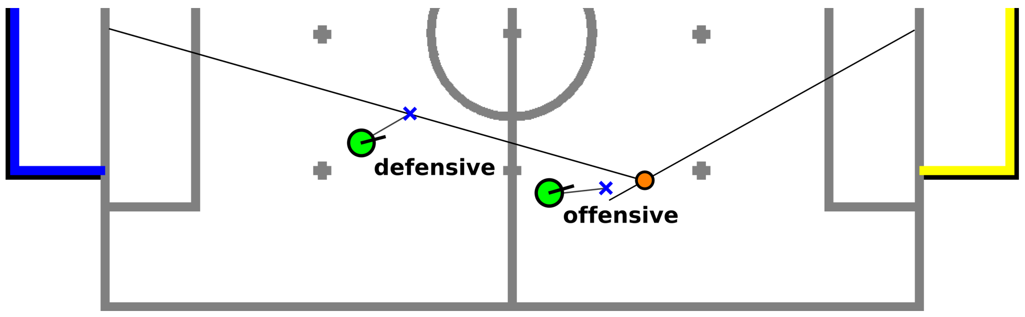


Fig. 10. Positioning of offensive and defensive field players (green circles with orientation indicator). The offensive field player positions itself on the line from ball to target goal (yellow) with sagittal and lateral offsets behind the ball (orange circle) on the blue cross, such that the ball lies in front of the kicking leg. The defensive field player takes a defensive, supporting pose by positioning itself on the direct line from ball to own goal at a certain distance from the ball. Color figure is only available in electronic version.

the robot itself. The relative positions of the hypotheses are altered according to the motion model. Their confidences are discounted by the time since the last update.

The following list describes the implemented soccer behaviors:

- *Searching the ball.* If the ball is not perceived, the robot has to explore the field for it. When a confident ball hypothesis exists, the robot gazes at and walks toward it. If no confident hypotheses exist, the robot first stops walking and sweeps its upper trunk over a range of $\pm\frac{\pi}{2}$, because the motion blur reduces the visibility range of balls during walking. If still no ball is in sight, it walks to the further goal, and then turns to the other goal. After this, it repeats the search process.
- *Walking toward the ball.* As it is possible to perceive the ball, while the opponent's goal is occluded, the robot is walking towards the ball. The behavior is not activated if the robot is close to its own goal.
- *Positioning behind the ball.* If ball and opponent's goal are visible, the robot positions itself behind the ball, as given in the current game state. When the distance to the target position is large, the robot orients itself toward the target position, such that it can approach it by mainly combining forward walking with turning. If it is near the target position, the robot aligns itself toward the ball-target. For intermediate distances, the gait rotation is interpolated linearly between the two alignment targets. The behavior also handles the case where the ball is located between the robot and the behind-ball position. Here, the robot walks around the ball by walking toward the target position but avoiding the ball-to-target line. As a defensive field player, however, the robot orients itself toward the ball at any distance without avoiding the ball-to-target line.
- *Kicking the ball toward the target.* This behavior is activated as soon as the behind-ball position has been reached with a certain precision in the angle to the ball-target and in the distance to the target position. The active behavior triggers a kick with the correct kicking leg. If the ball comes into a kicking position by chance, the behavior initiates a kick with the corresponding leg. As the robot has

to come to a complete stop before the kicking motion can be executed, it can cancel the kick if the ball rolls away in the meantime.

- *Dribbling the ball toward the target.* If positioning behind the ball was not successful for a longer time, or at kick-off, the robot dribbles the ball toward the ball-target for some time. Target-oriented dribbling is achieved by steering toward the ball if it is still far away. If it is close, the robot aligns the ball in the center in front of its feet by lateral motion, and aligns its orientation toward the ball-target. The better this alignment is, the faster the robot moves in the forward direction. At intermediate distances, the alignment targets are interpolated linearly. Dribbling inhibits kicking the ball.
- *Avoiding obstacles.* The visual perception supplies the behavior with the nearest obstacle. If it is detected closely in front of the robot, obstacle avoidance is activated. The avoidance sets the gait target actuator to a constant and a variable part of the direction from obstacle to robot, which is inversely proportional to the obstacle distance. If the ball is between obstacle and robot, the variable avoidance is weakened, such that the robot moves more aggressively behind the ball. A stuck situation is indicated by a resulting gait target vector that is small in length for a longer time. In this case, the robot may sidestep the obstacle if the ball is not between the obstacle in front and the robot and is perceived on one side of the obstacle. The action is canceled if either the preconditions for sidestepping do not hold anymore or a certain amount of time has elapsed since sidestepping was activated.
- *Controlling the gaze orientation.* A gaze control behavior keeps the ball within an angular range of $\pm\frac{\pi}{4}$ in the front camera by rotating the upper trunk in the yaw direction. If the ball is not visible or within range and the robot is localized, it aligns the upper body with the line between the goals to improve the visibility of the landmarks.
- *Goalkeeping.* The goalkeeper's objective is apparently to keep the ball out of the own goal. It positions itself in the own goal on the line from own goal to ball, or, if the ball is not visible, on the line between the goals. Balls close to the robot let it react immediately and trigger a corresponding goalie motion to capture the ball. To achieve fast reaction to an approaching ball, the visual perception supplies an estimate of the ball velocity from two successive frames. The goalkeeper reacts to balls that have a velocity larger than a fixed threshold. The type of goalkeeper motion, i.e. bending down or diving to a side, is determined by the intersection point of the moving ball direction and the goal line.

8. Tactics and Team Behaviors

The soccer behaviors so far make use of egocentric information obtained from visual perception. To implement team play and to act in an allocentric perspective, we introduced a next higher level in our behavior framework that abstracts from the reactive, egocentric soccer behaviors. On the basis of self-localization, tactical behaviors can configure the soccer behaviors, which have been implemented at the level below, by setting ball-target and target pose in allocentric coordinates at a rate of 20.83 Hz. To implement target pose positioning, an additional behavior is necessary at the next lower level that inhibits all other soccer behaviors. Moreover, the tactical behaviors can control when the soccer behaviors are allowed to dribble or kick. For instance, dribbling can be preferred to kicking. In this way, special game tactics can be implemented. Also, the technical challenges of the RoboCup Humanoid League, i.e. obstacle avoidance, slalom dribbling around poles, and passing between two field players, have been implemented with limited effort using this abstract interface.

Another main concept at the tactical level is the role of the player within the soccer team. A player can act as either an offensive field player, a defensive field player, or a goalkeeper. If only one field player is on the field, it plays offensive. When the team consists of more than one field player, the field players negotiate roles by claiming ball control. As long as no player is in control of the ball, all players attack. If one of the players takes control, the other player switches to the defensive role. A player may claim ball control if it is not already claimed, the relative ball position and angle are within a certain threshold, and the player is positioned behind the ball better than the other field player. As soon as the relative ball position and angle exceed a larger threshold or the player detects a fall or performs a kick, it abandons ball control. To assess the positioning quality of the players, a positioning utility is calculated by each player and communicated via WLAN to the other. The positioning utility depends on the deviation from the relative target position behind the ball that is given in the current game situation.

Another application of the role concept is goal clearance by the goalkeeper: the goalkeeper switches its role to field player when the ball gets closer than a certain distance. In this case, it starts negotiating roles with other field players like a standard field player. Thus, the goalie might walk toward the ball in order to kick it across the field.

We implemented the following behaviors at this level, excluding special behaviors for the technical challenges:

- *Role assignment.* The behavior implements default role negotiation and role switching as described above.
- *Positioning.* Positioning on a target pose is used for kick-off and for technical challenges.
- *Ball handling.* This behavior contains default ball-handling skills, i.e. dribbling the ball close to the goal instead of kicking, and dribbling the ball from the opponent's field corner to the center of the opponent's half.
- *Dribble around obstacles.* If the ball lies in front of a close obstacle, the player dribbles the ball around the obstacle onto the side where the angle between goal and obstacle is largest.

- *Dribble into freespace.* If an obstacle is located between the player and the ball-target in the opponent's half, the player dribbles the ball onto the field side that is not occupied by the obstacle.

9. Results

Our robots performed very well at the RoboCup 2007 competitions. In the TeenSize class, a penalty kick round robin was played between all seven teams. Robotina smoothly approached the ball and kicked it hard in one of the goal corners. Our goalkeeper Bodo reacted quickly and dived to keep the goal clear. Consequently, our robots won all six games of the round robin. They scored 27 of 30 possible goals, which corresponds to a 90% success rate, and let in only 10 goals. In the semifinal, they met again the titleholder, Team Osaka. The game ended 4:3 for NimbRo. In the final, our robots met Pal Technology from Spain [see Fig. 11(a)].[19] The Pal robot kicked the ball very hard, but could not dive to keep the goal clear. It adapted a strategy of walking into one of the corners to block it. Robotina was observing the goalkeeper's position and decided to kick the ball into the other (free) corner. The exciting game ended 5:4 for NimbRo.

Our TeenSize robot Bodo also excelled in the technical challenges. It was very quick to walk across the field in the foot race, where it needed only 10.27 s for a distance of more than 4 m. It also walked quickly through six black obstacle poles (16.46 s for more than 3 m) without touching any pole. Bodo dribbled the ball in slalom around three colored poles, completing two-thirds of the dribbling challenge. Only Team Osaka had the same degree of completion in the TeenSize technical challenges.

In the KidSize soccer tournament, the 22 teams were split into four groups. Our robots won all the games of their group. In the quarter final, they met Darmstadt Dribblers from Germany.[20] The exciting game ended 8:6 for NimbRo after extra

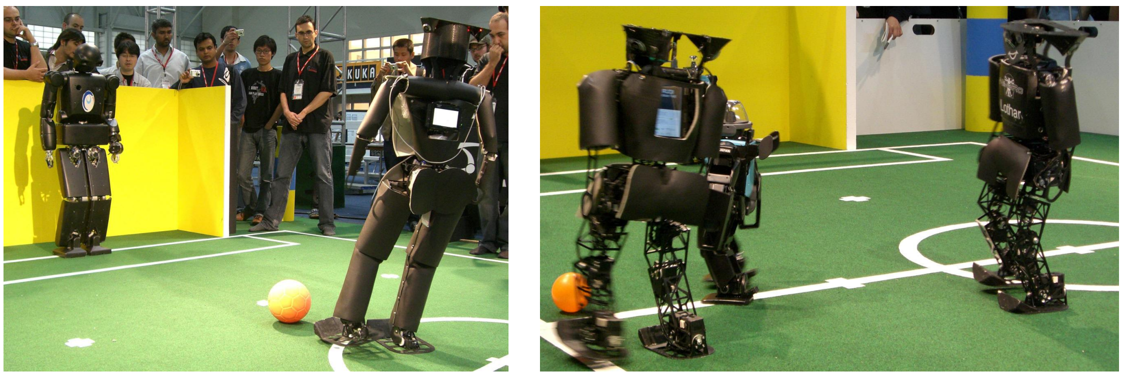

(a) (b)

Fig. 11. Final games of RoboCup 2007. (a) TeenSize penalty kick — NimbRo vs. Pal Technology. NimbRo won 5:4. (b) KidSize 2 vs. 2 soccer: NimbRo vs. Team Osaka. NimbRo won 8:6.

time. Our robots met the FUmanoids from Berlin, Germany, in the semifinal.[21] NimbRo won the game 11:0.

The same two teams that met in the 2005 and 2006 finals met in the 2007 final soccer game again: NimbRo vs. Team Osaka [see Fig. 11(b)].[22] While Team Osaka used a dedicated goalie and only one field player, we decided to use two field players and no goalie. The Osaka robots were very quick to walk behind the ball, positioned themselves for kicking, and kicked it hard across the field. The NimbRo robots excelled in one-on-one fights and in team play. When they perceived that the opponent was positioning itself for a kick, they walked against the ball to keep it moving. Thus, the Osaka robot had to approach the ball again. The second field player stayed in a defensive position, but took over if the primary attacker lost control of the ball. This team play was a big advantage for our team. The exciting game was open until the end. The final score was 8:6 for NimbRo. Videos showing the performance of our robots at RoboCup 2007 can be found at http://www.NimbRo.net.

10. Conclusion

This paper has described the design of the behavior control software for our NimbRo 2007 robots, which won all the humanoid soccer competitions of RoboCup 2007. We implemented the software in a framework that supports a hierarchy of reactive behaviors. A kinematic interface for body parts made it possible to abstract from individual joints when implementing basic skills like omnidirectional walking. These basic skills made it possible to abstract from body parts when implementing more complex soccer behaviors. At this player level, our humanoid robots are very similar to wheeled or four-legged soccer robots. Finally, at the tactical level, the players of our team are coordinated through role negotiation and the soccer behaviors are configured according to the game situation.

Playing soccer with humanoid robots is a complex task, and the development has just started. So far, there has been significant progress in the Humanoid League, which has moved in its few years from remote-controlled robots to soccer games with fully autonomous humanoids. Indeed, the Humanoid League is already the largest RoboCupSoccer league. The 2007 competition has shown that most robots master the basic skills of walking, kicking, and getting up. However, only a few teams are able to play decent soccer games.

We expect to see the rapid progress continue as the number of players is increased to 3 vs. 3 and the field size will be enlarged in 2008. This will widen the possibilities for team play. The increased complexity of soccer games with more players will make structured behavior engineering a key factor for success.

Many research issues, however, must be resolved before the humanoid robots reach the level of play shown in other RoboCupSoccer leagues. For example, the humanoid robots must maintain their balance, even when disturbed. In the next few years, the speed of walking must be increased significantly. The visual perception of the soccer world must become more robust against changes in lighting and other interferences. We continue to work on these issues.

Acknowledgments

This work is funded by the DFG under grant BE 2556/2-2 and 4-1. The authors thank all members of the NimbRo team for their extraordinary effort prior to and during the RoboCup 2007 competition.

References

1. H. Kitano and M. Asada, The RoboCup Humanoid Challenge as the millennium challenge for advanced robotics, *Adv. Robot.* **13**(8) (2000) 723–737.
2. S. Behnke and R. Rojas, A hierarchy of reactive behaviors handles complexity, in *Balancing Reactivity and Social Deliberation in Multi-Agent Systems* (Springer, 2001), pp. 125–136.
3. S. Lenser, J. Bruce and M. Veloso, A modular hierarchical behavior-based architecture, in *RoboCup 2001: Robot Soccer World Cup V*, Lecture Notes in Computer Science, Vol. 2377 (Springer, 2002), pp. 423–428.
4. A. Egorova, MAAT — Multi agent authoring tool for programming autonomous mobile robots, Diploma thesis, 2004.
5. H. Jaeger and T. Christaller, Dual dynamics: Designing behavior systems for autonomous robots, *Artif. Life Robot.* **2**(3) (1998) 108–112.
6. J. Hertzberg and F. Schönherr, Concurrency in the DD&P robot control architecture, in *Proc. Symp. Engineering of Natural and Artificial Intelligent Systems (ENAIS)* (2001).
7. A. Bredenfeld and H.-U. Kobialka, Team cooperation using dual dynamics, in *Balancing Reactivity and Social Deliberation in Multi-Agent Systems* (Springer, 2001), pp. 111–124.
8. H. Utz, G. Kraetzschmar, G. Mayer and G. Palm, Hierarchical behavior organization, in *Proc. 2005 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS 2005)*, Edmonton, Canada (2005), pp. 2598–2605.
9. 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*, Lecture Notes in Computer Science, Vol. 3020 (Springer, 2004), pp. 114–124.
10. T. T. Röfer, A behavior architecture for autonomous mobile robots based on potential fields, in *RoboCup 2004: Robot World Cup VIII*, Lecture Notes in Computer Science, Vol. 3276 (Springer, 2005), pp. 122–133.
11. V. A. Ziparo and L. Iocchi, Petri net plans, in *Proc. Fourth Int. Workshop on Modelling of Objects, Components, and Agents (MOCA'06)* (2006).
12. A. Ferrein, C. Fritz and G. Lakemeyer, Using golog for deliberation and team coordination in robotic soccer, *KI Kuenstliche Intelligenz* **1** (2005) 24–43.
13. R. Berger and H.-D. Burkhard, Modeling complex behavior of autonomous agents in dynamic environments, in *Proc. Concurrency, Specification and Programming (CS&P)* (2006).
14. M. Fujita, Y. Kuroki, T. Ishida and T. T. Doi, Autonomous behavior control architecture of entertainment humanoid robot sdr-4x, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (2003), pp. 960–967.

15. S. Chernova and R. C. Arkin, From deliberative to routine behaviors: A cognitively-inspired action selection mechanism for routine behavior capture, *Adaptive Behavior* **15**(2) (2007) 199–216.
16. J. Kramer and M. Scheutz, Development environments for autonomous mobile robots: A survey, *Autonom. Robots* **22**(2) (2007) 101–132.
17. S. Behnke, Online trajectory generation for omnidirectional biped walking, in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA'06)*, Orlando, Florida (2006), pp. 1597–1603.
18. J. Stückler, J. Schwenk and S. Behnke, Getting back on two feet: Reliable standing-up routines for a humanoid robot, in *Proc. 9th Int. Conf. Intelligent Autonomous Systems (IAS-9)*, Tokyo, Japan (2006), pp. 676–685.
19. D. Faconti, Technical description of REEM-A, in *RoboCup Humanoid League Team Descriptions*, Atlanta, GA (07/2007).
20. M. Friedmann, J. Kiener, S. Petters, D. Thomas and O. von Stryk, Darmstadt dribblers, in *RoboCup Humanoid League Team Descriptions*, Atlanta, GA (07/2007).
21. H. R. Moballegh, R. Guilbourd, G. Hohl, A. Reuschenbach, J. Yu and R. Rojas, Fumanoid team description 2007, in *RoboCup Humanoid League Team Descriptions*, Atlanta, GA (07/2007).
22. R. Matsumura, N. Shibatani, T. Imagawa, T. Maeda, T. Miyashita, T. Takahashi, Y. Akazawa, N. Yamato and H. Ishiguro, TeamOSAKA (Kid Size), in *RoboCup Humanoid League Team Descriptions*, Atlanta, GA (07/2007).

Sven Behnke received his M.S. degree in Computer Science in 1997 from Martin-Luther-University, Halle-Wittenberg, and his Ph.D. degree in 2002 from Freie Universität Berlin. In 2003, he worked as a postdoc at the International Computer Science Institute, Berkeley. From 2004 to 2008, he headed the Humanoid Robots group at Albert-Ludwigs-Universität, Freiburg. Since April 2008, he has been a full professor for Autonomous Intelligent Systems at Rheinische Friedrich-Wilhelms-Universität, Bonn. He represents the Humanoid League on the German National Committee of RoboCup. His research interests include artificial intelligence, robotics, and computer perception.

Jörg Stückler received his M.S. degree in Computer Science in 2007 from Albert-Ludwigs-Universität, Freiburg. Since April 2008, he has been working as a researcher in the Autonomous Intelligent Systems Group of Rheinische Friedrich-Wilhelms-Universität, Bonn. He has been a key member of the NimbRo team since 2004. His research interests include probabilistic robotics and multiagent systems.