NaoTH 2011 - Extended Team Description

The RoboCup NAO Team of Humboldt-Universität zu Berlin

Hans-Dieter Burkhard, Thomas Krause, Heinrich Mellmann, Claas-Norman Ritter, Yuan Xu, Marcus Scheunemann, Martin Schneider, and Florian Holzhauer

Institut für Informatik, LFG Künstliche Intelligenz, Humboldt-Universität zu Berlin, Rudower Chaussee 25, 12489 Berlin, Germany

http://www.naoth.de
nao-team@informatik.hu-berlin.de

1 Introduction

The research group *Nao-Team Humboldt* was founded at the end of 2007 and consists of students and researchers at the Humboldt-Universität zu Berlin, coming from a number of different countries during the last years, in particular from Germany, Italy, Russia, Egypt, China, Iraq, and Iran. The team is a part of the AI research Lab of the Humboldt-Universität which is headed by Prof. Dr. Hans-Dieter Burkhard and is led by Heinrich Mellmann. At the current state we have 2 Phd students and around 10 Master Students in the core team. Additionally we provide courses and seminars where the students solve tasks related to RoboCup. We have a long tradition within the RoboCup by working for the Four Legged league as a part of the GermanTeam in recent years, where we won the competition three times.

The current team members are: Heinrich Mellmann, Yuan Xu, Thomas
Krause, Claas-Norman Ritter, Florian Holzhauer, Marcus Scheunemann, Paul Schütte, Martin Schneider, Kirill Yasinovskiy, Luisa Jahn, Dominik Krienelke, Christian Rekittke.

We started with Naos in May 2008 and achieved the 4. place at the competition in Suzhou 2008. We achieved 3. place in technical challenge at Graz 2009. In 2010, we won the 2. place at RomeCup/Rom, 1. Place in Athens/Greece, and 4. place at German Open 2010. In 2010 we first time participated simultaneously in SPL in Simulation 3D with the same code. In the 3D Simulation we won the German Open and the AutCup competitions and achieved the 2. place at the RoboCup World Championship 2010 in Singapore. In 2011 we won the Iran Open competition in SPL and got 4. place in Simulation 3D, at the German Open we got 4. place as well.

With our efforts in both leagues, we hope to foster the cooperation between the two leagues and to improve both of them. We also cooperate with several other teams and associate projects at different universities, e.g., we regularly make test games with the humanoid team FUManoids[1].

Our general research fields include agent oriented techniques and machine learning with their applications to cognitive robotics. As our record shows, our results are recognized outside RoboCup as well. Our current research focuses among others mainly on the following topics:

- Narrowing the gap between simulated and real robots (section 2)
- Software architecture for an autonomous agent (section 3)
- Dynamic motion control (section 4)
- World modeling (section 5)

We will described them briefly in the next sections, please refer to our recent publications [1–7] for more details.

[1] http://www.fumanoids.de/

2 Simulation and Real Robots

As a common experience, there are big gaps between simulation and reality in robotics, especially with regards to basic physics with consequences for low level skills in motion and perception. There are some researchers who have already tried to narrow this gap, but there are only few successful results so far. We investigate the relationships and possibilities for methods and code transferring. Consequences can lead to better simulation tools, especially in the 3D Simulation League. At the moment, we participate in both, Standard Platform League and 3D Simulation League with the common core of our program, see Fig. 1. As already stated, therewith, we want to foster the cooperation between the two leagues and to improve both of them.

Fig. 1. NAO robots run in Standard Platform League(left) and 3D Simulation League(right).

When compared to real Nao robots, some devices are missing in the SimSpark, as LEDs and sound speakers. On one hand, we extended the standard version of SimSpark by adding missing devices like camera, accelerometer, to simulate the real robot. On the other hand, we can use a virtual vision sensor which is used in 3D simulation league instead of our image processing module. This allows us to perform isolated experiments on low level (e.g., image processing) and also on high level (e.g., team behavior). Also we developed a common architecture [2], and published a simple framework allowing for an easy start in the Simulation 3D league.

Our plan is to analyze data from sensors/actuators in simulation and from real robots at first and then to apply machine learning methods to improve the available model or build a good implicit model from the data of real robot. Particularly, we plan to:

- improve the simulated model of the robot in SimSpark;
- publish the architecture and a version of SimSpark which can be used for simulation in SPL;
- transfer results from simulation to the real robot (e.g., team behavior, navigation with potential field);

So far, we have developed walking gaits through evolutionary techniques in a simulated environment [8, 9]. Reinforcememt Learning was used for the development of dribbling skills in the 2D simulation [10], while Case Based Reasoning was used for strategic behavior [11, 12]. BDI-techniques have been investigated for behavior control, e.g., in [13, 14].

3 Architecture

An appropriate architecture (i.e., framework) is the base of each successful heterogeneous software project. It enables a group of developers to work at the same project and to organize their solutions. From this point of view, the artificial intelligence and/or robotics related research projects are

usually more complicated, since the actual result of the project is often not clear. In particular, a strong organization of the software is necessary if the project is involved in education. Our software architecture is organized with the main focus on modularity, easy usage, transparency and easy testing. Thereby the main parts are:

- platform interface
- module architecture
- debug infrastructure
- communication
- testing

For the implementation we use different programming languages and existing tools and libraries. Here is a rough overview over the used software:

- C++ at the robot
- Java for tools
- premake4 as build system
- glib for communication, etc.
- Google Protocol Buffers (protobuf) for serialization
- Google Test (googletest) and Google Mock (googlemock) for testing

In the following subsections we describe the design and the implementation of the parts mentioned above.

3.1 Platform Interface

In oder to integrate different platforms, our project is divided into two parts: platform independent part and platform specific one. The platform independent part is called the *Core*, which can run in any environment. All the algorithms are implemented here. The platform specific part contains code which is applied to the particular platform. In the Core part, several different modules are implemented under the module based architecture. Both parts are connected by a *platform interface*, see Fig. 2.

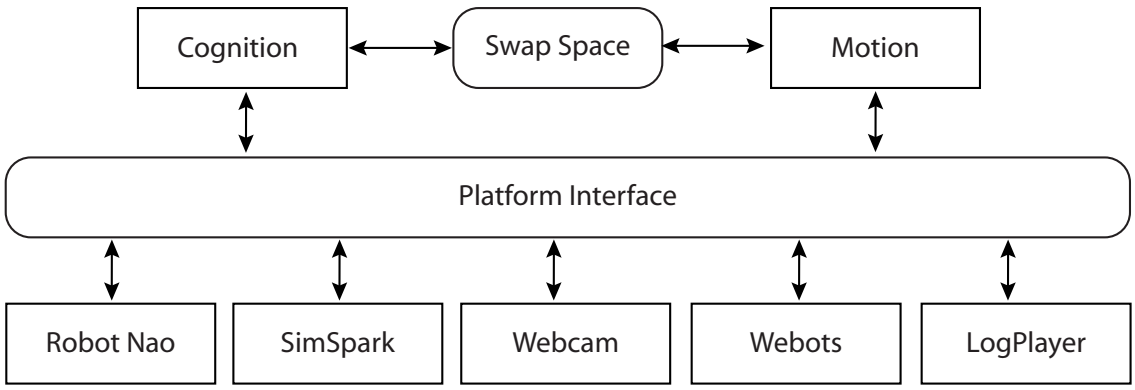

Fig. 2. Platform Interface is responsible for data transferring and execution of the Cognition and Motion.

3.2 Module Infrastructure

Our module framework is based on a *blackboard architecture*. It is used to organize the workflow of the *cognitive/deliberative* part of the program. The framework consists of the following basic components:

Representation objects carrying data, have no complex functionality
Blackboard container (data base) storing representations as information units
Module executable unit, has access to the blackboard (can read and write representations)
Module Manager manage the execution of the modules

A module may *require* a representation, in this case it has a read-only access to it. A module *provides* a representation, if it has a writing access. In our design we consider only sequential execution of the modules, thus the there is no handling for concurrent access to the blackboard necessary, i.e., it can be decided during the compilation time.

3.3 Debug Infrastructure

Our debug infrastructure allow for the debugging code to be switched on/off during runtime. Debug results are transferred over the network and monitored/visualized using our debuging tools. It is designed to analyze and debug a single robot, in order to analyze and develop a team behavior of a robot soccer team we need to connect to all the robots at the same time. For a detailed description refer to the section 7.

3.4 Communication

Communication between the robot and a PC is essential for debugging purposes and controlling tasks. We partitioned our architecture in a way that the user can choose to use different parts of our software but omit others. Thus we can not assume the availability of some given debugging software GUI (like RobotControl, cf. section 7) or the existence of all possible external helper libraries. Even with very few assumptions about the kind of communication and the used software stack we want to give the developers a powerful and flexible communication framework for interacting with the robot.

3.5 Testing

As mentioned in the introduction, the Nao Team Humboldt has a large code base which is being developed by a changing set of team members with various levels of education and knowledge about the project. This leads to various problems with the code, especially when several developers are working on different modules at the same time. Tests can be run by each developer after code changes to ensure that the changes did not result in undesired side-effects or broken code. The tests are implemented on two different granularity levels: *Unit-Tests* — calling a specific function with sample input, and *Integration Tests* — tests of a module to ensure that a module follows the requirements of our module framework and works properly.

We use googletest [15] as a xUnit-based testing library and googlemock [16] as a mocking framework. Upon a commit in our code repository, an automated build system compiles and tests the committed source code. If the build or the tests fail, the committer is notified by email about the error. As a side effect, the tests can be used as examples how to interact with more complex functions or modules, which enables new developers to understand the parts of our framework faster and easier.

4 Dynamic Motion Control

The performance of a soccer robot is highly dependent on its motion ability. Together with the ability to walk, the kicking motion is one of the most important motions in a soccer game. However, at the current state the most common approaches of implementing the kick are based on key frame technique. Such solutions are inflexible and costs a lot of time to adjust robot's position. Moreover, they are hard to integrate into the general motion flow, e.g., for the change between walk and kick the robot has usually to change to a special stand position.

Fixed motions such as keyframe nets perform well in a very restricted way and determinate environments. More flexible motions must be able to adapt to different conditions. There are at least two specifications: Adaption to control demands, e.g., required changes of speed and direction, omnidirectional walk, and adaptation to the environment, e.g., different floors. The adaptation of the kick according to the ball state and fluent change between walk and kick are another examples.

At the current state we have a stable version of an omnidirectional walk control and a dynamic kick which are used in the games. Along with further improvements of the dynamic walk and kick motions our current research focuses in particular on *integration* of the motions, e.g., fluent change between walk and kick.

Adaptation to changing conditions requires feedback from sensors. We experiment with the different sensors of the NAO. Especially, adaptation to the visual data, e.g., seen ball or optical flow, is investigated. Problems arise from sensor noise and delays within the feedback loop. Within a correlated project we also investigate the paradigm of local control loops, e.g., we extended the Nao with additional sensors.

We have investigated different Optimization and Machine Learning approaches, as evolutionary approaches [17, 8] and recurrent Neural Networks [18] for Aibos and humanoid robots, and - as mentioned earlier - Reinforcement Learning (RL) with applications for the 2D simulation league [10]. Recently, another PhD thesis with focus on RL for humanoid robot walking is being written.

4.1 Dynamic Kick

For the implementation of the dynamic kick motion we have to consider following aspects:

- planning of the motion according to the seen ball and the desired kick direction
- reachability space of the motion
- stability while standing on one foot, and moving the other

Together with the desired direction and the strength of the kick this aspects define conditions which have to be satisfied by the motion trajectory in order to perform the kick. In our approach we generate the motion trajectory dynamically according to this conditions, which allows the robot to handle different situations and adapt to changing conditions, e.g., moving ball.

We implement and test our approach in simulation and on the real robot. At the current state the robot is able to kick the ball from any position which is reachable by the foot to any suitable direction. Our current research concerns amongst others, kicking of a moving ball and dynamically change between walk and kick (i.e., without stop).

For detailed description of the implementation, please refer to [5, 6]. Videos showing some experiments performed on the real robot can be found on our homepage.

5 Perception and World Modeling

In order to realize a complex and successful cooperative behavior it is necessary to have a appropriate model of the surrounding world. Thus, one of the main focuses of our current research is the improvement of the perceptional abilities of the robot and its capabilities to build a world model.

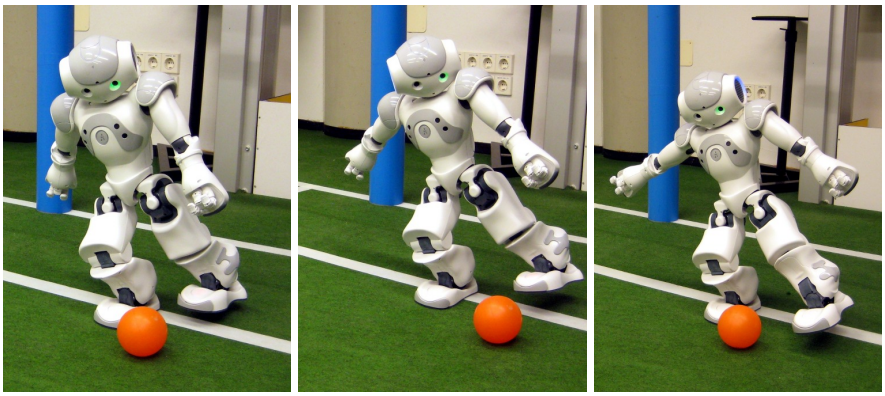

Fig. 3. The robot adapts motion according to changing position of the ball (left and middle), and the changing intended direction of the kick (middle and right).

5.1 SURF based Object Recognition and Tracking

In order to get more independent from color coded image processing we are actively working on using standard CV approaches in the SPL. One of these approaches is the SURF feature detection as provided by the OpenCV library. We integrated OpenCV into our code base and try to learn features to recognize objects, such as the ball, goal posts and robots. Since SURF only needs a gray scale image and is quite tolerant to light changes, this approach minimizes the requirement of calibration before games. One must also find ways to aggregate the detected features into more complex models of objects (like robots). Currently we are able to detect other robots including their rotation using distinctive features on the feet. Future research will focus on classification (learning) and tracking of the features.

We already achieved a processing speed of about 20ms on the real robot. With special pre-filtering and selection of the areas which should be considered for the SURF feature detection we want to improve the runtime of the algorithm even further.

Constraint Selflocalization We investigate the application of Constraint based techniques for navigation, and compare the approach to classical Bayesian approaches like Kalman filters and Monte-Carlo methods. Kalman filters make certain assumptions about the environment (linearity of the model, Gaussian noise), Monte-Carlo methods as particle filters are restricted by high computational demand and thus, low dimensionality. We have investigated these methods under various perspectives in the previous years [19–26] including missing information [27]. Constraint techniques have to handle inconsistent data, but can be advantageous whenever ambiguous data is available [28]. They are computationally cheap using interval arithmetic, and they can be easily communicated allowing for cooperative localization [29–34].

Classical approaches are prone to error propagation (e.g. from camera matrix over self localization to global positions of other objects). Using constraints, all related parameters (e.g. kinematic chain, camera matrix, positions) can be connected by related constraints and the common solution can be calculated by propagation techniques, [35–37].

Constraints are built by image percepts from flags, goals, lines and other objects (even dynamical ones like ball and other players) [31]. They are able to deal with missing information [27] as well. Other sensors such as joint sensors lead to constraints for the camera matrix. Further constraints can use communicated information and past information, e.g., object speed. Each information can be used to constrain the related parameters. The shape of the constraint is determined by the type of information and the expected sensor noise. We use constraint propagation (interval arithmetic) for further restrictions of possible values of the parameters, as for positions [35, 37]. The Figure 4

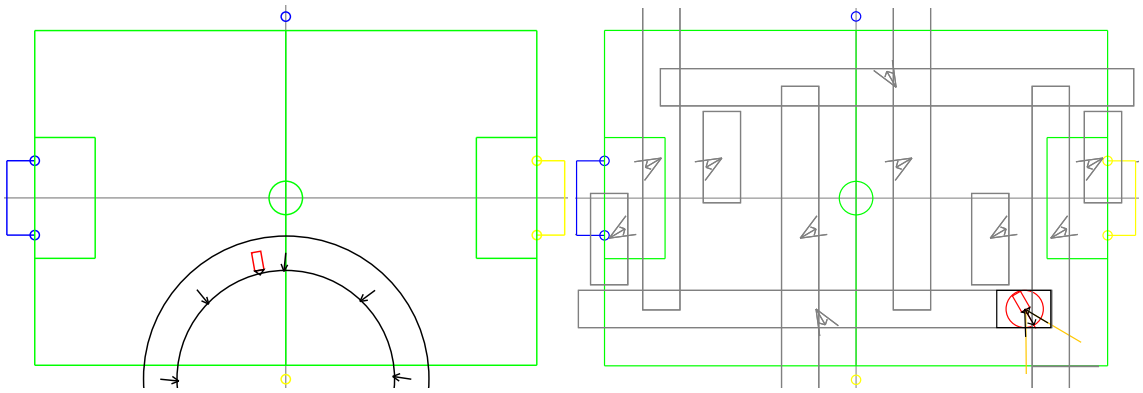

Fig. 4. Both figures show constraints, generated from percepts (pictures show the algorithm on an Aibo field). Left: a circular constraint as generated from flag percepts. Right: a line constraint, generated from line percepts.

illustrates constraints created from visual perception and used to estimate the robots position on the field.

Recent investigations and experiments deal with the treatment of inconsistencies (e.g. by ghost percepts) and further increase of performance. Furthermore we developed and tested a constraint-based player model.

6 Behavior

The Extensible Agent Behavior Specification Language — *XABSL* cf. [38] is a behavior description language for autonomous agents based on hierarchical finite state machines. XABSL is originally developed since 2002 by the *German Team* cf. [39]. Since then it turned out to be very successful and is used by many teams withing the RoboCup community. We use Xabsl to model the behavior of single robots and of the whole team in the Simulation League 3D and also in the SPL.

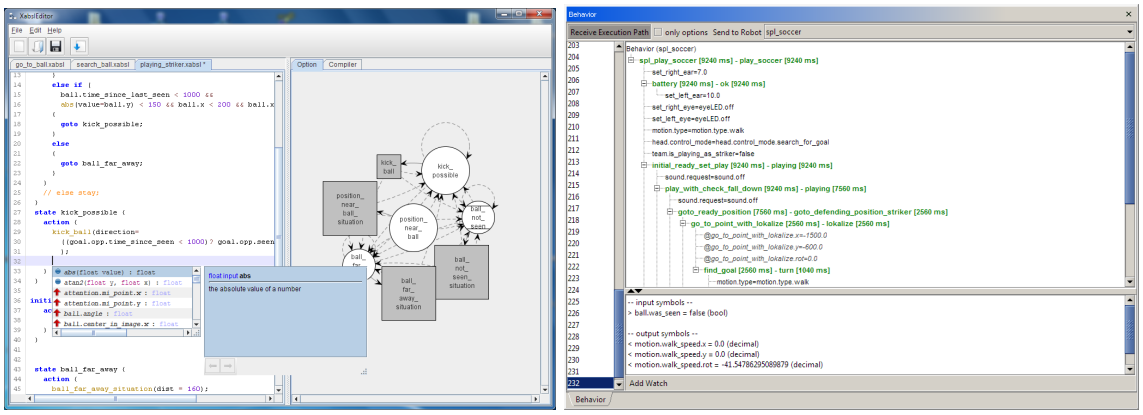

Fig. 5. (left) XabslEditor: On the left side you see the source code of an behavior option. On the right side the state machine of this option is visualized as a graph. (right) In the man frame the execution path is shown as a tree; Bottom some monitored symbols can be seen, the developer can decide which symbols should be monitored; On the left side there is a list of buffered frames, which is very useful to see how the decisions changed in the last time;

In order to be independent from the platform we develop our tools in Java. In particular we are working on a Java based development environment for XABSL named *XabslEditor*. In fact, this tool consists of an full featured editor with syntax highlighting, a graph viewer for visualization of behavior state machines and an integrated compiler. Figure 5 (left) illustrates the *XABSL Editor* which an open behavior file.

Another useful tool we are working on is the visualizer for the XABSL execution tree, which allows to monitor the decisions made by the robot on runtime. At the current state this visualizer is part of our debugging and monitoring tool *RobotControl*. The Figure 5 (right) illustrates the execution tree of the behavior shown within the visualizer.

7 Debuging and Tools

In order to develop a complex software for a mobile robot we need possibilities for high level debugging and monitoring (e.g., visualize the posture of the robot or its position on the field). Since we don't exactly know which kind of algorithms will be debugged, there are two aspects very important: accessibility during the runtime and flexibility. The accessibility of the debug construct is realized based on the our communication framework (see 3.4). Thus, they can be accessed during the runtime either directly by telnet or using a visualization software like RobotControl as shown in the Fig. 6). Further, all debug concepts are realized based on the described command executor concept, thus it is easy to introduce new concepts. In contrast to the other parts, the debug infrastructure is not separated from the code, all concepts are statically available and thus they can be used at any position in the code. Some of the ideas were evolved from the GT-Architecture [40]. The following list illustrates some of the debug concepts:

debug request activate/deactivate code parts
modify allows a modification of a value (in particular local variables)
stopwatch measures the execution time
parameter list allows to monitor and to modify lists of the parameters
drawings allows visualization in 2D/3D, thereby it can be drawn into the image or on the field (2D/3D)

As already mentioned, these concepts can be placed at any position in the code and can be accessed during the runtime. Similar to the module architecture, the debug concepts are hidden by macros to allow simple usage and to be able to deactivate the debug code at the compilation time if necessary.

Additionally, to these individual debugging possibilities there are some general monitoring possibilities: the whole content of the blackboard, the dependencies between the modules and representations and execution times of each single module. The Fig. 6 illustrated some of the visualizations of the debug concepts. In particular a field view, a 3D view, behavior tree, plot and the table of debug requests are shown. The TeamControl shown in the Fig. 6 allows for the visualization of the team behavior. In particular the positions of the robots on the field and their intended motion direction are visualized.

References

1. Cotugno, G., Mellmann, H.: Dynamic motion control: Adaptive bimanual grasping for a humanoid robot. In: Proceedings of the Workshop on Concurrency, Specification, and Programming CS&P 2010. Volume Volume 2., Börnicke (near Berlin), Germany (September 2010)
2. Mellmann, H., Xu, Y., Krause, T., Holzhauer, F.: Naoth software architecture for an autonomous agent. In: International Workshop on Standards and Common Platforms for Robotics (SCPR 2010), Darmstadt (November 2010)
3. Xu, Y., Mellmann, H., Burkhard, H.D.: An approach to close the gap between simulation and real robots. In Ando, Balakirska, Reggiani, von Stryk, eds.: 2nd International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR), Darmstadt (November 2010)

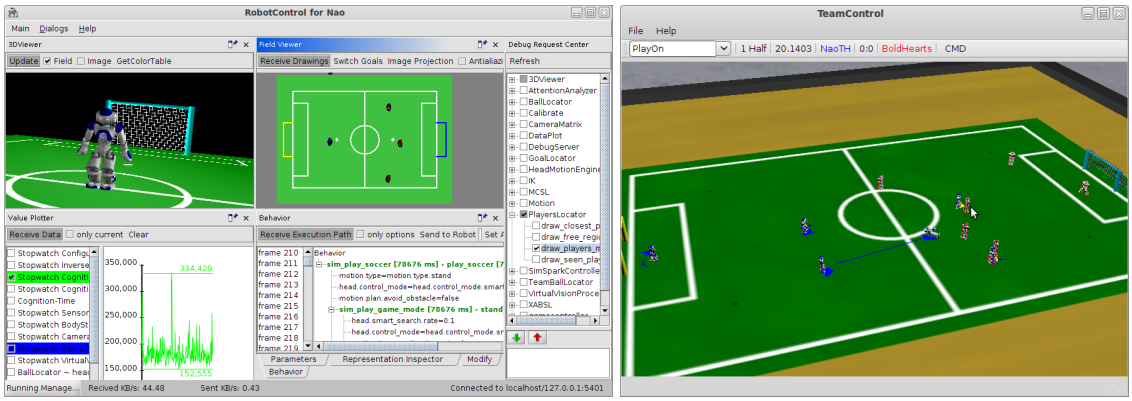

Fig. 6. (left) The RobotControl program contains different dialogs. In this figure, the left top dialog is the 3D viewer, which is used to visualized the current state of robot; the left bottom dialog plots some data; the middle top dialog draws the field view; the middle bottom shows the behavior tree; and the right one is the debug request dialog which can enable/disable debug functionalities. (right) The *TeamControl* is used to monitor team behavior. In the shown screenshot you can see the positions of the robots on the field and their intended motion direction illustrated by arrows.

4. Xu, Y., Burkhard, H.D.: Narrowing reality gap and validation: Improving the simulator for humanoid soccer robot. In: Concurrency, Specification and Programming CS&P'2010, Helenenau, Germany (September 2010)
5. Xu, Y., Mellmann, H.: Adaptive motion control: Dynamic kick for a humanoid robot. In: the Proceedings of the 33rd Annual German Conference on Artificial Intelligence, Karlsruhe, Germany (2010)
6. Mellmann, H., Xu, Y.: Adaptive motion control with visual feedback for a humanoid robot. In: the Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei (2010)
7. Mellmann, H.: Ein anderes Modell der Welt. Alternative Methoden zur Lokalisierung mobiler Roboter. Diploma thesis, Humboldt-Universität zu Berlin, Institut für Informatik (2010)
8. Hein, D., Hild, M., Berger, R.: Evolution of biped walking using neural oscillators and physical simulation. In: RoboCup 2007: Robot Soccer World Cup XI. Lecture Notes in Artificial Intelligence, Springer (2007)
9. Hein, D.: Simloid – evolution of biped walking using physical simulation. Diploma thesis, Humboldt-Universität zu Berlin, Institut für Informatik (2007)
10. Uc-Cetina, V.: Reinforcement Learning in Continuous State and Action Spaces. PhD thesis, Humboldt-Universität zu Berlin (2009)
11. Burkhard, H.D., Berger, R.: Cases in robotic soccer. In Rosina O. Weber, M.M.R., ed.: Case-Based Reasoning Research and Development, Proc. 7th International Conference on Case-Based Reasoning, ICCBR 2007. Lecture Notes in Artificial Intelligence, Springer (2007) 1–15
12. Berger, R., Lämmel, G.: Exploiting Past Experience. Case-Based Decision Support for Soccer Agents. In: Proceedings of the 30th Annual German Conference on Artificial Intelligence (KI'07), Springer (2007)
13. Berger, R.: Die Doppelpass-Architektur. Verhaltenssteuerung autonomer Agenten in dynamischen Umgebungen (in German). Diploma thesis, Humboldt-Universität zu Berlin, Institut für Informatik (2006)
14. Burkhard, H.D.: Programming Bounded Rationality. In: Proceedings of the International Workshop on Monitoring, Security, and Rescue Techniques in Multiagent Systems (MSRAS 2004), Springer (2005) 347–362
15. : googletest - google c++ testing framework
16. : googlemock - google c++ mocking framework
17. Düffert, U.: Vierbeiniges Laufen: Modellierung und Optimierung von Roboterbewegungen (in German) (2004) Diploma thesis.
18. Hild, M.: Neurodynamische Module zur Bewegungssteuerung autonomer mobiler Roboter (in German). PhD thesis, Humboldt-Universität zu Berlin (2007)
19. Hoffmann, J.: Proprioceptive Motion Modeling for Monte Carlo Localization. In: RoboCup 2006: Robot Soccer World Cup X. Lecture Notes in Artificial Intelligence, Springer (2007)

20. Jüngel, M.: Bearing-only localization for mobile robots. In: Proceedings of the 2007 International Conference on Advanced Robotics (ICAR 2007), Jeju, Korea,. (August 2007)
21. Jüngel, M.: Self-localization based on a short-term memory of bearings and odometry. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007), San Diego (October 2007)
22. Jüngel, M., Mellmann, H.: Memory-based state-estimation. Fundamenta Informaticae **85**(Number 1-4) (2008) 297–311
23. Jüngel, M., Mellmann, H., Spranger, M.: Improving vision-based distance measurements using reference objects. In Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup 2007: Robot Soccer World Cup XI. Volume Volume 5001/2008 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2007) 89–100
24. Jüngel, M., Risler, M.: Self-localization using odometry and horizontal bearings to landmarks. In Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup 2007: Robot Soccer World Cup XI. Lecture Notes in Artificial Intelligence, Springer (2007)
25. Mellmann, H., Jüngel, M., Spranger, M.: Using reference objects to improve vision-based bearing measurements. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2008, Acropolis Convention Center, Nice, France, IEEE (22–26 Sept. 2008) 3939–3945
26. Mellmann, H.: Active landmark selection for vision-based self-localization. In: Proceedings of the Workshop on Concurrency, Specification, and Programming CS&P 2009. Volume Volume 2., Kraków-Przegorzaly, Poland (28–30 September 2009) 398–405
27. Hoffmann, J., Spranger, M., Göhring, D., Jüngel, M., Burkhard, H.D.: Further studies on the use of negative information. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA) IEEE. (2006)
28. Göhring, D., Mellmann, H., Gerasymova, K., Burkhard, H.D.: Constraint based world modeling. Fundamenta Informaticae **85**(Number 1-4) (2008) 123–137
29. Göhring, D.: Distributed object modeling using object relations in dynamic environments. In: Proceedings of Concurrency, Specification and Programming (CS&P 2006). (2006)
30. Göhring, D., Hoffmann, J.: Sensor Modeling Using Visual-Object Relation in Multi Robot Object Tracking. In: RoboCup 2006: Robot Soccer World Cup X. Lecture Notes in Artificial Intelligence, Springer (2007)
31. Göhring, D., Burkhard, H.D.: Multi robot object tracking and self localization using visual percept relations. In: Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS), pages 31–36, IEEE (2006)
32. Göhring, D.: Cooperative object localization using line-based percept communication. In Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F., eds.: RoboCup 2007: Robot Soccer World Cup XI. Lecture Notes in Artificial Intelligence, Springer (2007)
33. Göhring, D., Burkhard, H.D.: Cooperative world modeling in dynamic multi-robot environments. Fundamenta Informaticae **75**(1–4) (2007) 281–294
34. Göhring, D.: Constraint Based World Modeling for Multi Agent Systems in Dynamic Environments. PhD thesis, Humboldt-Universität zu Berlin (2009)
35. Göhring, D., Mellmann, H., Burkhard, H.D.: Constraint based belief modeling. In Iocchi, L., Matsubara, H., Weitzenfeld, A., Zhou, C., eds.: RoboCup 2008: Robot Soccer World Cup XII. Lecture Notes in Artificial Intelligence, Springer (2008)
36. Göhring, D., Mellmann, H., Burkhard, H.D.: Constraint based object state modeling. In Herman, B., Libor, P., Miroslav, K., eds.: European Robotics Symposium 2008. Volume Volume 44/2008 of Springer Tracts in Advanced Robotics., Prague, Chech Republic, Springer Berlin / Heidelberg (2008) 63–72 This volume (EUROS 2008).
37. Göhring, D., Mellmann, H., Burkhard, H.D.: Constraint based world modeling in mobile robotics. In: Proc. IEEE International Conference on Robotics and Automation ICRA 2009. (2009) 2538–2543
38. Lötzsch, M., Jüngel, M., Risler, M., Krause, T.: XABSL web site. (2006) http://www.ki.informatik.hu-berlin.de/XABSL.
39. Lötzsch, M., Risler, M., Jüngel, M.: Xabsl - a pragmatic approach to behavior engineering. In: Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS), Beijing, China (October 9-15 2006) 5124–5129
40. Röfer, T., Brose, J., Göhring, D., Jüngel, M., Laue, T., Risler, M.: GermanTeam 2007 - The German national RoboCup team. In: RoboCup 2007: Robot Soccer World Cup XI Preproceedings, RoboCup Federation (2007)