Preprint of the paper submitted to the International Conference on Robotics and Biomimetics, December 17-20, 2006, Kunming, China, page 613-618

Modular software architecture for teams of cooperating, heterogeneous robots

Martin Friedmann, Jutta Kiener, Sebastian Petters, Dirk Thomas, Oskar von Stryk
Department of Computer Science
Technische Universität Darmstadt
Darmstadt, Germany
{*friedmann, kiener, petters, dthomas, stryk*}*@sim.tu-darmstadt.de*

***Abstract*— For teams of cooperating autonomous lightweight robots with challenging dynamical locomotion properties a platform independent modular software architecture and platform independent modules for sensor data processing, planning and motion control have been developed. The software architecture allows high level communication between modules on different abstraction levels of the control architecture within one robot system as well as communication between different and heterogeneous robots and computers using wireless network. Very different behavior control paradigms may be realized on the basis of the developed architecture. The application to teams of cooperating small and medium size humanoid robots is investigated in this paper. Scenarios for inter robot communication and cooperative task accomplishment are described.**

***Index Terms*— cooperative multi-robot systems, mobile robotics, humanoid robots, robot control software**

I. Introduction

Cooperating systems of multiple autonomous robots are currently investigated in several scenarios, e.g. cooperative transport of a load by flying or driving robots, cooperative monitoring and surveillance in disaster areas or in the highly dynamic environment of autonomous legged robot soccer teams. To allow effective, flexible and robust development of heterogeneous robot teams the software must meet the requirements of modularity to integrate different platforms and competing modules for sensor processing, localization, motion control, and behavior control as well as flexibility to adopt to changing hardware like processors, cameras or locomotion system. For cooperative scenarios the exchange of sensor data, world model information and behavior decisions is imperative for an effective robot team performance. Currently, even communication between heterogeneous robots of different manufacturers or developers is a problem not even mentioning cooperative task achievement. Additionally, autonomous lightweight systems with challenging dynamical locomotion properties like small or medium size humanoid robots or unmanned aerial vehicles must meet strong restrictions on the available CPU power when compared with standard PCs. This must be incorporated in the low- and high-level software design.

One challenge in the development of mobile autonomous robots reacting to fast changing environments is the execution of rapid, goal oriented and situation aware motions considering motion stability and real time constraints. For the solution of the herein presented tasks wheeled and humanoid robots are investigated. A special scenario for humanoid robots in such an environment is given in robot soccer in the RoboCup [2], an annually held world wide competition for different robot types. To meet these challenges a suitable software architecture is required.

Frameworks and architectures for mobile wheeled cooperative robot systems are developed since the last 20 years [3]. Among them, Miro [17] is based on CORBA and optimized for high efficient multi processor systems. In difference to these systems legged robots suffer from a low additional payload restricting power supply and therefore onboard computation capabilities. Thus most humanoid robots are equipped with energy saving and therefore less powerful single or multi processor systems. On these systems the use of CORBA leads to a loss of efficiency as most of the advantages of this middleware cannot be used.

Saphira [9], [14], the software architecture on the commonly used wheeled Pioneer robots, allows the direct development of high level applications. However, as Saphira is not transparent and not available in source code, integration of further sensor or actuator hardware, which is not supported by the manufacturer, is complicated. Additionally, an important feature for sensor data processing, the generation of timestamps for sensor data, is not possible to add.

The architecture ALLIANCE [15] is tailored for wheeled and six legged robot systems. It focuses on fault tolerant cooperation and multi robot learning and uses a fully distributed control architecture.

CLARAty [12] is a framework for generic and reusable robot components, which can be adapted to different platforms, but there is no direct support for cooperating multiple robot systems.

Furthermore, all of these architectures aim towards wheeled platforms and do not consider the special demands of lightweight four and two legged robots with potential instabilities in locomotion and their requirements in software architecture.

Approaches for functional architectures of autonomous humanoid robots were only recently presented, but they do not meet the needs required by the complex task of team cooperation or efficient portability to a wide range of hardware (sensors, actuators, onboard computers).

One example is the Japanese HRP2 humanoid robot system software [13]. This monolithic software does not support the

exchange of single modules, porting to other platforms and cooperation of several robots.

A modular, distributed control architecture for a humanoid torso mounted on wheels used in a human robot cooperation is described in [11], without mentioning the special aspects concerning goal oriented multi robot cooperation and bipedal locomotion.

In recent years significant improvements in mobile robotics with humanoid robots were achieved in different projects for cooperative work with humans and humanoid control [19]. The success of the cooperation for complex task achievement is mainly based on the human part. The robot is primarily used as a teleoperated tool, but not as a stand-alone autonomous system.

The software architecture [16] developed by the GermanTeam (HU Berlin, U Bremen, TU Darmstadt, U Dortmund) in the Four Legged Robot League of the RoboCup meets the demands of cooperative legged robot teams.Despite its modularity this architecture lacks the ability of flexibly changing modules as the modules' interfaces are fixed. This inhibits changing the application's structure, e. g. for testing or exchanging new modules with modified input and output. Furthermore the graphical user interface (GUI) for debugging the application is highly dependent on MS Windows specific graphics libraries and thus only usable on this system.

To enable heterogeneous autonomous robot cooperation in addition to suitable hardware the underlying software must meet the described requirements for mobile robot systems combined with the special needs for lightweight robots. In this paper, the newly developed platform independent framework *RoboFrame* is presented. Together with the corresponding modules for sensor processing it provides efficient mechanisms for data exchange on single or multi processor systems used on wheeled and humanoid robots, an easily usable interface for the exchange of high level software modules typically occurring in an autonomous robot architecture and a flexible and easy adaptable hardware and operation system abstraction layer. Due to its modular design the adaption to different sensors and actuators is simplified and accomplished in short time. Debugging, testing and data validation is supported by an extendable and platform independent GUI.

II. Supported Robot Platforms

The software architecture has so far been successfully installed on several heterogeneous real and simulated robot systems, both legged and wheeled.

A. Humanoid Robots

The investigated prototypes of humanoid robots are based on a similar kinematic design with six non redundant degrees of freedom (DOF) in each leg, three or four DOF in each arm, two DOF in the neck and a slightly different number of DOF in the waist (0, 1, or 2). The robots are equipped with servo motors with different holding torque (20 - 37 kg-cm). Two of the supported humanoid robots are described in more detail: Mr. DD and Bruno (Fig. 1 and Table I).

All robots have been equipped with monocular (Bruno) or stereo camera systems (Mr. DD) built from inexpensive off-the-shelf camera systems mounted on the 2 DOF robot neck.

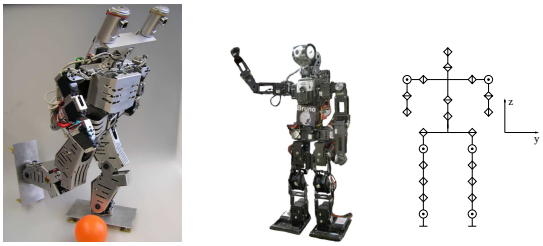

Fig. 1. Two of the supported humanoid robot systems: Mr. DD (left), Bruno (middle) and kinematic structure of Bruno (right).

TABLE I
Technical data of the selected humanoid robots.

Model	Mass	Height	DOF	onboard PC	Processor
Mr. DD	4.8 kg	65 cm	24	embedded System	133 MHz
Bruno	3.5 kg	55 cm	21	PPC Acer n50	523 MHz

In addition to the articulated narrow angle lens camera for higher resolution vision, Bruno is equipped with a 100 degree wide angle lens camera in the chest for peripheral but still directed view. This combination provides the robot with a binocular, variable resolution view of its environment. It may be seen as an embodiment of the inner, focused and the outer, peripheral vision of the human eye using two different cameras. They are connected to the controller board via USB.

Stable walking requires a lightweight upper body with little mass for the onboard PC, batteries and sensors [4], [18]. Due to the low additional payload for bipedal robots, a major problem is to design powerful enough onboard computing components which weigh as less as possible and require as less additional energy supplies as possible. The selected onboard PCs for the humanoid robots are an embedded Linux system and a Pocket PC with Windows CE, because they offer a computing environment close to a regular PC and thus are well suited for algorithms in high level languages. Additionally, the Pocket PC comes with an integrated battery and a compact full featured computer with display and touch screen for possible onboard debugging. Bruno is equipped with an additional 32bit micro controller board with 50 MHz for servo controle and motion generation.

Further information concerning these robots may be found in [7] or on the Darmstadt Dribbler's homepage www.dribblers.de.

B. Wheeled Robot Systems

The customized wheeled robot Pioneer 2dx (Fig. 2, left) is equipped with 2 actuated wheels and a gripper for actuation and sonar, camera, bumper, an 3-axes gyroscope, and a 2-axes accelerometer for sensing. In difference to the other presented robot systems, the Pioneer 2dx comes with a high additional payload of 20 kg, as the system moves on wheels with only little problems concerning stable locomotion.

The robot's actuators and sensors are controlled by a micro controller board with a C166 processor, all higher level computations are performed by a VSBC-6 mainboard (256 MB RAM) with a AMD K6-3 processor (400 MHz) running Windows 2000 [1].

Fig. 2. Supported wheeled robot: Pioneer 2dx .

C. Simulation

To enable software-in-the-loop testing of robot-controle software, a simulation environment has been developed at the authors' group. This program is capable of simulating motion and camera data of biped and wheeled robots. The simulation may be used transparently with the controle software. The main advantage of the usage of a simulated robot is the permanent availability of the device, independent of time, space and expensive hardware. Especially the performance of an individual robot's autonomous behavior as well as team behavior can be tested extensively before being implemented to the real robot system to avoid hardware faults caused by malfunctions in the software but also to validate the correctness and efficiency of different modules of the software architecture like vision, perception, localization and behavior. Fig. 3 shows a picture of the simulation with two humanoid robots on a RoboCup KidSize soccer field.

Fig. 3. Simulated robot in the RoboCup environment. In the left figure the simulated robot is approaching the ball, in the right figure the image generated by the simulated head camera of Bruno is shown.

III. Software Architecture

The object oriented software is developed in C++ and consists of the framework RoboFrame [6] and of modules executed in this framework for different tasks like image processing, object detection, localization, motion and behavior control. For execution of these modules the framework provides transparent communication mechanisms for interaction and data exchange between modules running in one thread of execution, different threads or processes on the same CPU or even running on different CPUs.

Caused by the short development cycles for new versions of improved robot hardware components a modular design and easy exchangeable modules are crucial for rapid software development and maintenance.

To allow a flexible use of the framework in heterogeneous teams of legged and wheeled robots much effort has been spent to develop a hardware and operation system independent core with an easily adjustable system dependent abstraction layer handling synchronization, multi-threading and network access (see Fig. 4). Currently the operating systems Linux (on standard x86 PC and MIPS architecture), BSD Unix, Windows 2000, Windows XP and Windows CE are supported. It was crucial for the framework to have a small memory footprint to allow the usage on systems equipped with few memory. Simple applications can be realized with less than 200kb.

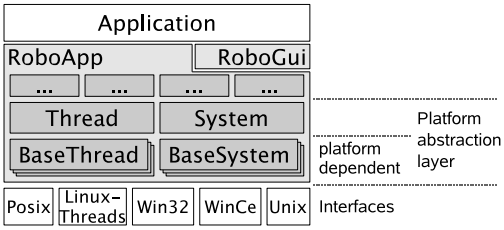

Fig. 4. Platform abstraction is based on encapsulation of operating system calls, thread management and synchronization and network connections. It enables to provide RoboFrame and applications with specific functionality.

To enable time optimized data exchange two different communication mechanisms are implemented: For use with large data structures on one machine, a blackboard architecture using shared memory and mutual exclusion mechanisms is provided.

For smaller data structures, the preferred way of data exchange is using a message based system. The messages can be arbitrary objects, which become serialized, and are transparently transmitted by the framework between the modules. If the modules exchanging messages reside within the same thread, the framework allocates the data exchange buffers in a way which avoids the cost for copying messages thus improving performance. This mechanism enables to exchange even very large messages without any overhead, as long as the modules run in the same thread (see Fig. 5). Messages and shared-memory data are identified by unique keys and are timestamped.

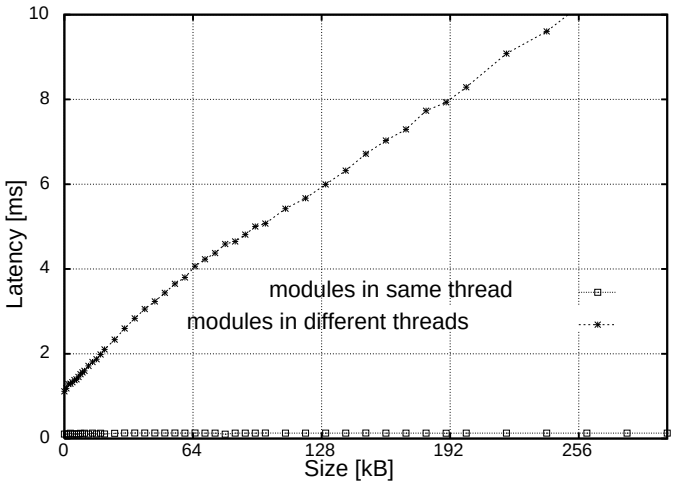

Fig. 5. Latency between sending and receiving a message from one module to another measured on a PocketPC with Intel XScale CPU at 400 MHz. If both sender and receiver reside in the same thread, the buffer allocated by the framework is shared resulting in no delay by copying the message. Otherwise the latency increases with the size of the message.

For communication between different machines the network protocols TCP for reliable, connection oriented and UDP for unreliable, connectionless, datagram oriented communications can be used. For reasons of performance in most application UDP is used to reduce the latency if reliability is not a primary concern.

To create a concrete application within the framework, the required modules are added to threads provided by the framework. The threads can be configured to be executed at a given frequency and / or if they receive new input data for further processing.

Further on the framework provides tools for the rapid development of GUIs. A GUI can be connected to multiple applications build with the framework using a reliable TCP connection. It controls the application and provides a wide range of debugging, monitoring and testing tools.

Any message sent within the application can be requested by the GUI for further investigation. Using the framework dialogs can be developed which display messages in their respective context depending on the module sending the message. Modules also can generate special debugging messages, which will only be sent to the GUI, or receive GUI generated messages.

The GUI also features a special message recording tool to write any selection of messages emitted by a running application to a log file. The log files can be investigated offline for debugging or be replayed to the application. This feature allows an easy comparison of different modules accomplishing the same task by providing them with the same input data and comparing their respective output data.

To ensure the functionality and correctness of RoboFrame, a set of test cases has been implemented. This has been done using the common tool for regression tests CppUnit. For each of the main components, multiple tests provide a nearly full coverage. It is planned to execute the tests after each change in the revision control system to find problems at an early stage. Additionally, the tests can easily be extended to test the modules which normally reside outside the framework.

IV. Implementation for Application Scenarios

Due to the flexibility of the data exchange provided by the framework, the control architecture is not limited to a reactive or a hierarchical-deliberative paradigm. Arbitrary structures can be realized, depending on the individual connections between interchangeable modules for sensor data acquisition, image processing, behavior and motion generation.

A. RoboCup scenario

For use in the RoboCup scenario the application layout shown in Fig. 6 has been developed. Single modules may be replaced within the existing application without affecting other modules, as long as the key and type of their message queues are not changed. Different algorithms accomplishing the same task can be compared easily by implementing them as modules emitting and receiving the same messages and testing them in the same application. Additionally, the recombination of individual modules to new applications is possible.

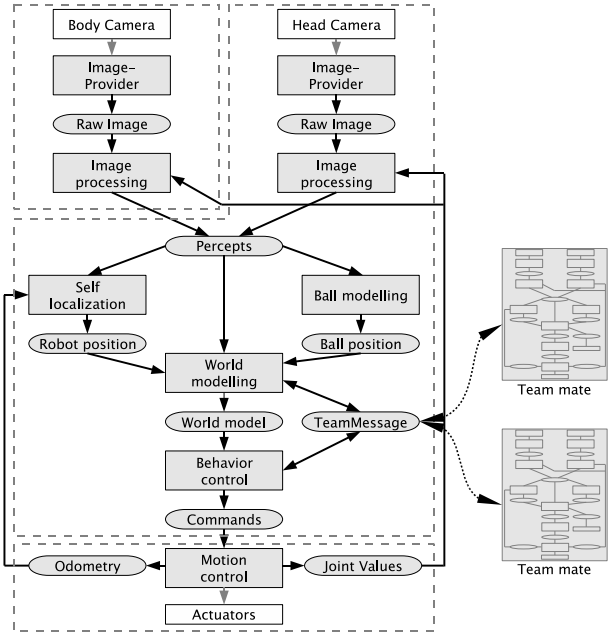

Fig. 6. Overview of modules (rectangles) and exchanged messages (ellipses) of a control architecture for the application robot soccer. White blocks are sensors or actuators, gray blocks are modules executed on the Pocket PC. Modules within the same dashed gray box are running within the same thread.

Overview of modules. The `ImageProvider` is a hardware dependent module providing image data of the camera in use. Based on the images, the `ImageProcessing` module detects objects of interest (in the current scenario e.g. ball, goals, beacons, ...). Information on any of the objects found is provided in robot centric coordinates to the following modules. To perform the coordinate transformation, the current joint values are used to gather information about the camera's current position and orientation.

`SelfLocalization` estimates the robot's current position based on objects detected by the camera and information on the motion currently performed by the robot. Currently Markov localization with particle filtering [5] is used for this task. `BallModeling` tracks the position of the ball using a Kalman filter. The `WorldModeling` module fuses any data generated by the previous modules as well as information generated by the robot's team mates to estimate the current state of the game.

This information is used in the `Behavior` module to generate a decision on the robot's next action. Currently the behavior is modeled discretely by a hierarchical state machine described in the behavior specification language XABSL [10]. This description is interpreted during runtime to allow quick exchange and debugging of the behavior. The behavior is able to communicate with other robots of the same team to generate coordinated cooperative actions.

Actions requested by the behavior module are executed by the `MotionControl` module. This module strongly depends on the specific (legged) robot's kinematics, dynamics and hardware used for locomotion and has to be adopted to it. The statically stable walking motions are generated in real time using parameterized curves for foot and hip trajectories which are transformed into joint angles using inverse kine-

matics. The walking motions have been optimized regarding to stability and walking speed by varying the controlling parameters, e.g. step length and pitch angle in the upper body etc., in a hardware-in-the-loop optimization approach [8]. The resulting forward walking speed of 40cm/s in permanent operation of Bruno was the fastest of all humanoid robots at RoboCup 2006. Besides controlling the robot's motion this module also provides information on the robot's current state of motion (e.g. speed, direction, position of joints).

All modules except the motion generation are executed on the onboard Pocket PC. The motion generation is performed on a micro controller board which is able to satisfy the hard real time constraints.

B. Cooperation and Adaptability

Inter Robot Communication, Dynamic Role and Task Assignment. Exchange of information within a team of robots is provided by a communication module, which can send any data package from one team member to all others by a fast, but possibly unreliable UDP broadcast. Each communication partner is configured with an unique identifier to allow a mapping of an incoming message to a robot. By this means sensor data information are shared between the robots. This enables fusion of information gathered by several robots, thus providing more reliable information on the environment as well as the possibility to detect failures of single sensors.

Furthermore, the behavior modules communicate their current decisions and roles to the other robots of the team and allow a matching dynamic role assignment of the team mates. Based on these information the robots may change their roles dynamically according to the current situation in the scenario. This permits coordinated actions of the robots.

Tasks are assigned in a team of robots via a task allocation module by a Contract-Net based method. The tasks can be executed in a cooperation independently parallel, sequential or synchronously parallel.

Reusability. For use in different scenarios the existing modules may be easily reused, rearranged and if needed combined with new, application specific modules. Hardware dependent modules for the motion of different wheeled and biped platforms have been developed and used in several scenarios described in the results section.

V. Results, Summary and Outlook

A modular and flexible software framework has been developed for teams of cooperating, autonomous robots characterized through heterogeneous hardware, lightweight robot design and possibly challenging dynamical locomotion properties.

The presented hardware and software was tested successfully at several competitions in robot soccer (RoboCup 2005, 2006; GermanOpen 2005). The robots showed penalty kicks, goalie behavior and cooperative 2 vs. 2 soccer games.

Architecture and portability. The rapid porting of the software components to other robot platforms has been demonstrated for several different robot types. Similar constructed humanoid robots could be controlled within some hours, the communication with a four legged robot was provided within two workdays, and the wheeled robot Pioneer 2dx using a different operating system was integrated at the time of a few workdays. All these new features where designed to be enhancements instead of modifications to be able to use them simultaneously in the same code base.

Team communication between differently driven robot systems. In early experiments a four legged robot and a humanoid robot exchange position information of a seen ball. The ball is visible only for one of the robots, the other estimates the position based on the communicated ball messages. If the transmission is successful, the client robot walks towards the ball resp. executes a predefined motion related to the position of the ball (see Fig. 7).

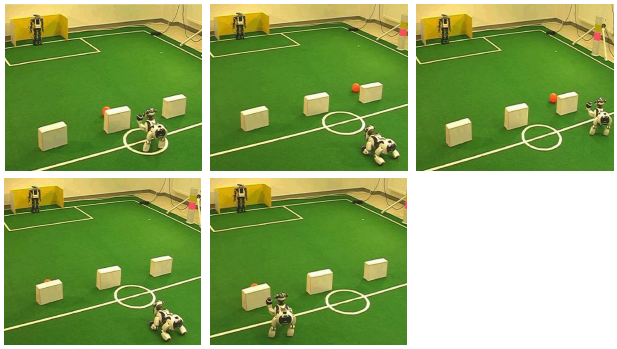

Fig. 7. Cooperative ball search: The humanoid robot communicates the seen ball position to the four legged robot. During communication, the four legged robots waits in the middle (Images 2, 4), when the ball is communicated, the four legged robots walks to the communicated position and moves its leg (Images 1, 3, 5; images numbered from upper left to bottom right).

Inter robot message exchange is used to improve failure safety of the world modeling. Missing data on one system can be compensated by the corresponding communicated messages from another robot system. As the messages are labeled with their respective sender, the receiving modules can decide, whether the contained information was acquired by the robot itself or by another system. In case of transmission failure the modules are designed not to be dependent on the external messages.

During RoboCup 2005 two humanoid robots took part in heterogeneous robot soccer as goalie and field player. The task to dribble a ball into the opposing goal was solved by a robot based on the world model information. As part of this process one robot's information is enhanced by the communicated ball position of the team mate's world model. This data exchange offers a more flexible and robust behavior as more information are available for data processing.

Cooperative behavior between heterogeneous humanoid robots.

In a scenario with humanoid robots kicking a ball the robots exchange data on two different levels: Data generated by the world modeling module, including the position of the ball and of team mates, is shared by all robots. By fusing this data a more reliable and failsafe system is created, as errors in the perception of one robot may be compensated by others. Decisions made in the behavior level are based on this merged world model. For a dynamic role assignment the roles of the other robots, generated in the behavior level,

are communicated to the team mates and taken into account by them. This strategy avoids situations, where several robots approach the ball at the same time. Instead of hampering each other only the robot closest to the ball approaches it, while the other robot clears the area. Both communication patterns were applied successfully during the RoboCup 2006 to achieve an improved world model and to realize a dynamic role assignment for a striker and supporter behavior, see Fig. 8.

Based on this cooperative behavior the robots won all games except one and came in third in soccer in a group of 16 participating teams.

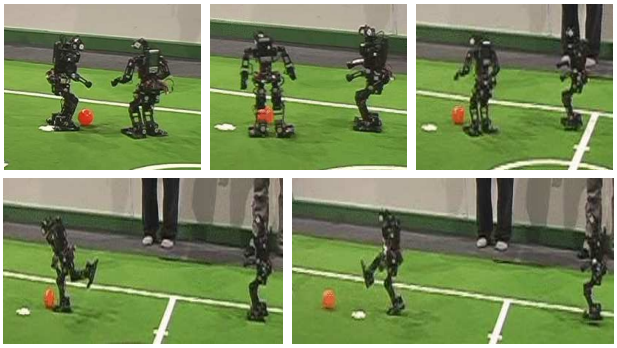

Fig. 8. Cooperative team play in humanoid robot soccer: Both robots detect the ball (Image 1), the robot closer to the ball approaches, the other robot is communicated to sidestep (Image 2). When the first robot claims the role "striker", the other robot obtains the role "supporter" and walks in its own half (Image 3), until the kick is executed (Images 4, 5).

Currently first tests of synchronously executed tasks in a cooperation of a humanoid robots, equipped with a camera, and a wheeled robot system are being undertaken. The task is to kick a ball into a goal after following it for a long distance. In this scenario only the humanoid robot is able to kick the ball. As the humanoid robot is slower in locomotion than the wheeled robot and the wheeled robot has a high additional payload, the wheeled robot serves as a platform for transportation for the humanoid robot. The perception task is executed by the humanoid robot, which communicates the object position to the wheeled robot. This robot executes the locomotion task based on the communicated object position. The cooperation must be done synchronously, otherwise the wheeled robot will fail in locomotion. The task is implemented with the presented humanoid robot Bruno and the wheeled robot Pioneer 2dx in simulation, see Fig. V.

Future work. The presented development provides a basis for more complex scenarios with heterogeneous robot systems.

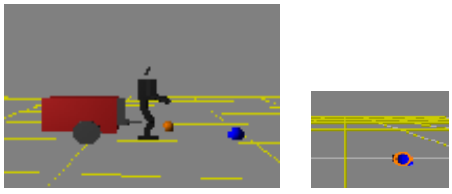

Fig. 9. Simulated Following Scenario: The humanoid robot is transported by the wheeled robot (left) and perceives the ball (right).

The cooperative behavior used in the RoboCup can be enhanced to communicate the prospective kick of the ball towards team mates so that the other robot can prepare for receiving the ball.

Currently under investigation is the extension and application of the software architecture to outdoor robots as lightweight marine, off-road ground and aerial vehicles. In the long run envisioned applications for teams of such heterogeneous robots include cooperative monitoring, search and rescue in a catastrophe.

REFERENCES

[1] *Pioneer 2, Mobile Robots, Operational Manual for Pioneer 2dx*, www.mobilerobots.com.

[2] "The RoboCup federation," www.robocup.org.

[3] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14 – 23, March 1986.

[4] M. Buss, M. Hardt, J. Kiener, M. Sobotka, M. Stelzer, O. von Stryk, and D. Wollherr, "Towards an autonomous, humanoid, and dynamically walking robot: modeling, optimal trajectory planning, hardware architecture, and experiments," in *Third IEEE International Conference on Humanoid Robots*, vol. 3, 2002, pp. 2491 – 2496.

[5] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo localization: Efficient position estimation for mobile robots," in *Proc. of the Nat. Conf. on Artificial Intelligence*, Orlando, USA, 1999, pp. 343–349.

[6] M. Friedmann, J. Kiener, S. Petters, D. Thomas, and O. von Stryk, "Reusable architecture and tools for teams of lightweight heterogeneous robots," in *Proc. 1st IFAC-Symposium on Multivehicle Systems*, Salvador, Brazil, 2 - 3 Oct 2006, p. to appear.

[7] M. Friedmann, J. Kiener, R. Kratz, S. Petters, H. Sakamoto, M. Stelzer, D. Thomas, and O. von Stryk, "Darmstadt Dribblers & Hajime Team (KidSize) and Darmstadt Dribblers (TeenSize): Team description paper," 2006. [Online]. Available: www.dribblers.de

[8] T. Hemker, H. Sakamoto, M. Stelzer, and O. von Stryk, "Hardware-in-the-loop optimization of the walking speed of a humanoid robot," in *CLAWAR: Int. Conf. on Climbing and Walking Robots*, Brussels, Belgium, 12 - 14 Sep 2006, pp. 614 – 623.

[9] K. Konolige, "Saphira robot control architecture," SRI International, Tech. Rep., 2002.

[10] M. Lötzsch, M. Risler, and M. Jüngel, "Xabsl - a pragmatic approach to behavior engineering," in *IEEE International Conference on Intelligent Robots and Systems (IROS), submitted*, 2006.

[11] D. Ly, K. Regenstein, T. Asfour, and R. Dillmann, "A modular and distributed embedded control architecture for humanoid robots," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2775 – 2780.

[12] I. Nesnas, A. Wright, M. Bajracharya, R. Simmons, T. Estlin, and W. S. Kim, "CLARAty: An architecture for reusable robotic software." in *SPIE Aerosense Conference*, Orlando, Florida, April 2003.

[13] K. Okada, T. Ogura, A. Haneda, D. Kousaka, H. Nakai, M. Inaba, and H. Inoue, "Integrated system software for HRP2 humanoid," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, New Orleans, LA, USA, 26 Apr - 1 May 2004, pp. 3207 – 3212.

[14] A. Orebäck and H. I. Christensen, "Evaluation of architectures for mobile robotics." *Autonomous Robots*, vol. 14, no. 1, pp. 33–49, 2003.

[15] L. E. Parker, "Alliance: An architecture for fault-tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[16] T. Röfer *et al.*, "German Team RoboCup 2005 - Team Report," TU Darmstadt, Uni Bremen, HU Berlin, Uni Dortmund, Tech. Rep., 2005. [Online]. Available: www.germanteam.org

[17] H. Utz, S. Sablatnög, S. Enderle, and G. K. Kraetzschmar, "Miro – middleware for mobile robot applications," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 4, pp. 493–497, August 2002.

[18] D. Wollherr, M. Hardt, M. Buss, and O. von Stryk, "Actuator selection and hardware realization of a small and fast-moving autonomous humanoid robot," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2002, pp. 2491 – 2496.

[19] K. Yokoyama, H. Handa, T. Isozumi, Y. Fukase, K. Kaneko, F. Kanehiro, Y. Kawai, F. Tomita, and H. Hirukawa, "Cooperative works by a human and a humanoid robot." in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 3, 14 - 19 Sep 2003, pp. 2985–2991.