Mostly Harmless Team Description

Gerald Steinbauer, Michael Faschinger, Gordon Fraser, Arndt Mühlenfeld,
Stefan Richter, Gernot Wöber, and Jürgen Wolf

Graz University of Technology, Inffeldgasse 16b/II, A-8010 Graz, Austria
robocup@tugraz.at
http://www.robocup.tugraz.at

1 Introduction

In this paper we describe the Mostly Harmless Robotic Soccer Team of Graz University of Technology, that has been formed to participate in RoboCup 2003 as the first Austrian Robotic Soccer Team. The four robots have been developed in cooperation of three faculties of our university. We have designed the robots in a modular way in order to make them as flexible and robust as possible. This module-based design allows for an easy adaptation of our robots for other task beside robotic soccer. Due to the interdisciplinary cooperation our team faced a wide range of research topics like software development, system diagnosis and monitoring, computer vision, machine learning [1], planning, reasoning and robot control.

2 Hardware Architecture

Our main design goal was to build a flexible and robust robotic system which should not only be dedicated to robotic soccer. Therefore, we divide the robot in four independent layers with pre-defined functionalities.

- *Control Layer:* Provides the main control and the communication of the robot.
- *Sensor Layer:* Provides sensory data.
- *Actuator Layer:* Provides manipulation of the environment.
- *Driving Layer:* Provides the movement and the power supply of the robot.

Each of these layers is implemented as a single independent autonomous module. The modules are stacked to build up the robot platform (See Figure 1). Every module is equipped with its own processing unit, a Single Board PC in the *Control Layer* and a C167 micro-controller in the other layers. The communication within the modules is done via a reliable CAN-Bus. All modules are supplied by a single battery pack that is situated in the *Driving Layer*.
Due to the module-based design, the implementation of modules with different functionalities and the adaptation of the robots to other tasks than robot soccer is quite simple. Currently, we are working on a gripper for the *Actuator Layer* to adapt the robot for delivery services within the institutes.

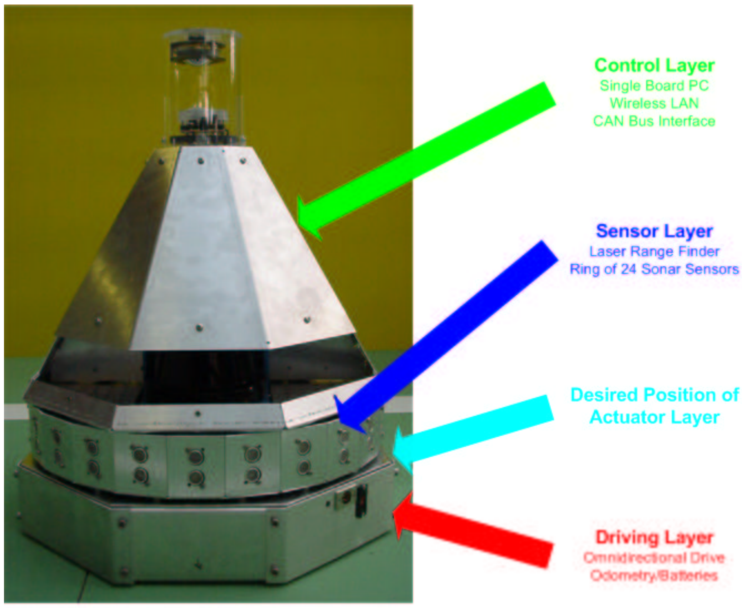

Fig. 1. The modularized robot platform.

2.1 Driving Layer

The *Driving Layer* provides the movement of the robot. In our current design this layer is implemented as an omnidirectional drive. It is built up by four orthogonal crosswise motors each joint to an omni-wheel. By individual control of the speed and the direction of each motor the robot is able to move in any direction and to rotate around its vertical axis simultaneously. This feature prevents the robot from getting stuck in crowded soccer situation as well as in corners or impasses in office environments. This layer also hosts the battery packs to keep the center of gravity of the robot low.

2.2 Actuator Layer

All active interactions with the environment are done by the *Actuator Layer*. This layer is either implemented as a pneumatic kicking device during a soccer game or a manipulator arm with a gripper during service tasks.

2.3 Sensor Layer

The *Sensor Layer* provides the entire sensing of the environment. This layer actually hosts two sensor systems. We use the Sick Laser Range Finder for proximity scans around the robot. This sensor has a high resolution (0.5°) and provides very reliable measurements. A disadvantage of this sensor is the limitation of the scan to 180° around the robot. Therefore, this sensor is supported by another sensor system, a ring of 24 ultrasonic sensors. These sensors have a significantly lower resolution and accuracy but they provide a qualitative scan around the whole robot. Currently we investigate two different implementations of this system (different ultrasonic sensors).

2.4 Control Layer

The more sophisticated information processing is done in the *Control Layer*. This includes more advanced sensory data processing, the decision making process and higher level control. Therefore, this layer is equipped with a powerful processing unit, an 850 MHz Pentium III Single Board PC with 256 MB Ram and a 20 GB hard drive. This layer also provides a communication channel to other robots or computers via a wireless LAN. Due to the field of view and the connection via the Firewire-Interface the omnidirectional camera is mounted on top of this layer. A crucial constraint for the assembly of this layer is the use of standard interfaces instead of proprietary ones (PCI, USB, Firewire, CAN). This eases the exchange of components within this layer.

3 Software Architecture

The Control Architecture is a three tier architecture with different degrees of abstraction and timing granularity (See Figure 2).

- *Hardware Layer:* Provides the interfaces to the sensors and actuators of the robot, i.e. the hardware drivers.
- *Continuous Layer:* Provides processing of the sensory data, image processing, integration of the extracted features in a continuous world model and execution of actions.
- *Abstract Layer:* Provides easy integration of abstract domain knowledge, e.g. RoboCup rules, reasoning about this knowledge and the abstracted world status, longer term planning and strategic decisions.

We use the *Middleware for Cooperative Robotics (MIRO)* [2] as the software-framework for our Control Architecture. MIRO is a CORBA-based framework for robot applications. It provides interfaces to a wide rage of sensors and actuators, mechanisms for simple integration of modules in the framework and a wide range of mechanisms for communication between this modules. This framework allows for a very flexible software design.

3.1 Hardware Layer

The *Hardware Layer* implements the interfaces to the sensors and actuators of the robot. This layer delivers the raw continuous sensory data and performs low level controlling of the actuators. USB for the Laser Range Finder and Firewire for the omnidirectional camera are standard interfaces and already supported by our OS (Linux). The interfaces to modules on the CAN-Bus are implemented as

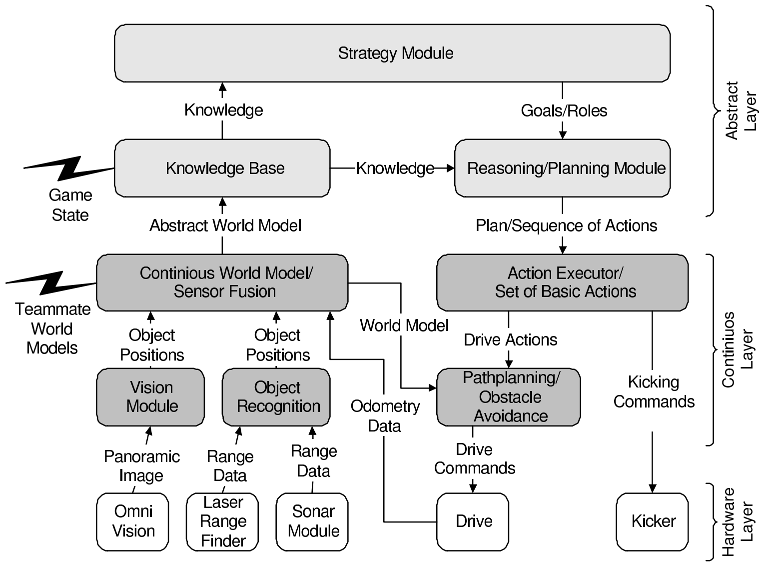

Fig. 2. Software Architecture.

Virtual CAN-Connections. A software module is able to communicate via these connections directly with one dedicated hardware module on the CAN-Bus. This approach is similar to the TCP/IP protocol used in computer networks.

3.2 Continuous Layer

The *Continuous Layer* implements a numeric representation of the sensing and acting of the robot. This layer performs processing of sensory data and image processing. This processing provides possible positions of objects in the environment including the robots own pose. Together with the position information broadcast by other robots and provided by the odometry these positions are fused into a continuous world model by Monte Carlo methods [3] (own pose) and Kalman-Filters (other objects). The world model represents the continuous world by estimating the most likely hypothesis for the position of an object. Furthermore this layer is responsible for the execution of actions. This execution is based on a set of atomic actions implemented as simple control circuits.

3.3 Abstract Layer

The *Abstract Layer* represents the robot's knowledge in a symbolic way and contains a planning module for decision making. The core of this layer is the *Knowledge Base* which incorporates previously collected domain knowledge, an abstracted representation of the continuous world model, and an abstract description of the actions the robot is able to perform. Based on this knowledge the *Strategic Module* chooses the immediate next goal the robot has to achieve in order to fulfill a long term task. The *Planning Module* generates a plan (sequence of basic actions) that satisfies this goal by applying classical AI-planning techniques. This plan is monitored permanently for its validity during execution. It is canceled or repaired if the plan's invariant or other relevant preconditions are no longer valid. The actions contained in the plan are executed by the Action Executor, which communicates with services of the Continuous Layer. The *Abstract Layer* allows easy implementation of desired behaviors by specifying the goals and knowledge as logical sentences.

4 Research Interests

Due to the wide range of disciplines the involved institutes deal with, our team faces many different research topics like software development, system diagnosis and monitoring, computer vision, machine learning, planning, reasoning and robot control. In our first year of RoboCup participation we concentrated on the design and implementation of robust multi-purpose robots and the development of a flexible software framework. Now we are trying to improve the software development process to achieve more robust software and a reuse and an exchange of software among Middle Size teams. MIRO is one step in that direction. By using online diagnosis and reconfiguration of sensors we want to improve the robustness of the robots. Furthermore, we want to improve robot control by application of biologically inspired neural networks (e.g. spiking neurons).

5 Conclusion

We presented the hardware and software design of our new developed robot platform. In both, hardware and software, the platform is very flexible due to its module based design. Therefore, the robot platform easily can be adapted to other tasks beside robot soccer (e.g. service robots).
The described platform was already successfully used for the RoboCup tournament 2003, numerous experiments and extensive research. The same platform will again be used for the RoboCup tournament 2004, and research on system diagnosis, computer vision, machine learning, multi-agent systems, system design and planning will be intensified.

References

1. W. Maass, G. Steinbauer, and R. Koholka. Autonomous fast learning in a mobile robot. In G. D. Hager, H. I. Christensen, H. Bunke, and R. Klein, editors, *Sensor Based Intelligent Robots. International Workshop, Dagstuhl Castle, Germany, October 15-25, 2000, Selected Revised Papers*, volume 2238. Springer, 2002.
2. Hans Utz, Stefan Sablatng, Stefan Enderle, and Gerhard K. Kraetzschmar. Miro – middleware for mobile robot applications. *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures*, 18(4):493–497, August 2002.

3. Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *AAAI/IAAI*, pages 343–349, 1999.