GMD-Robots

Ansgar Bredenfeld, Vlatko Becanovic, Thomas Christaller,
Horst Günther, Giovanni Indiveri, Hans-Ulrich Kobialka,
Paul-Gerhard Plöger, Peter Schöll
GMD - Institute for Autonomous intelligent Systems (AiS)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

e-mail: bredenfeld@gmd.de
web: http://ais.gmd.de/BE/robocup

1 Introduction

The overall research goal of GMD´s RoboCup middle-size league team is to increase both, the controlled speed of mobile robots acting as a team in dynamic environments and the speed of the development process for robot control systems. Therefore, GMD started in 1998 to develop a proprietary fast robot platform and in parallel the development of the integrated Dual Dynamics Design Environment.

This team description gives some details on the current state of our hardware platform (February 2001), the control software architecture we use and the design environment we constructed to specify, simulate, run and test our robots. In addition, we point our some specific skills of our robots.

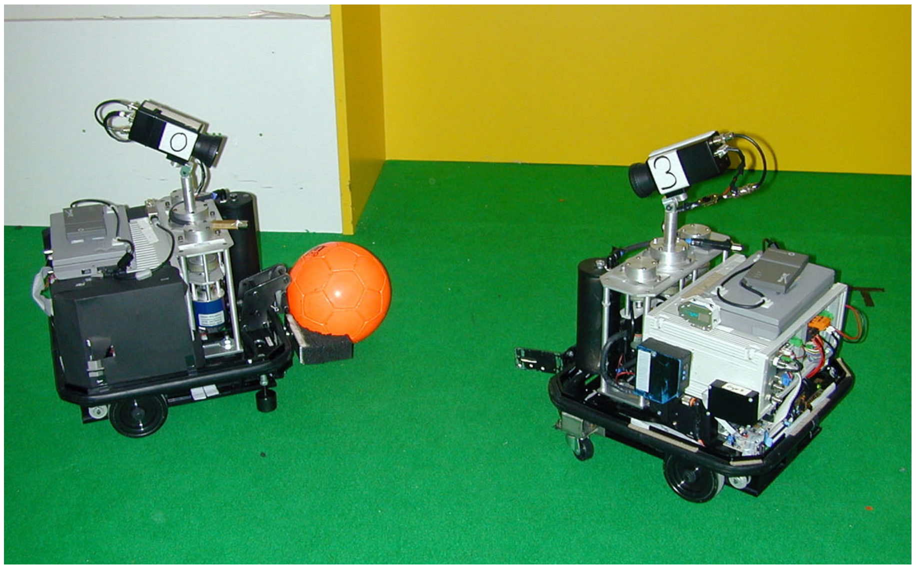
2 Hardware Platform

Our robot hardware is a custom-built 2 degree of freedom platform. We use two 20 Watt, high-quality Maxon motors that are mounted on a very solid, mill-cut aluminium frame. A piezo-gyroscope senses relative changes of heading direction. Obstacle avoidance is supported by four infrared range detectors and a surrounding ring of switches integrated into protective bumpers. Our robots kick the ball with a pneumatic device which is integrated in a ball guidance.

The goalie has a slightly different sensor configuration. In contrast to the field players, all sensors are mounted 90 degree turned. Its infrared range detectors point to the back of the goal. In addition, the kicking device of the goalie has no ball guidance but a simple plate to kick the ball.

The computer system of the robot consists of a Pentium PC notebook connected to two C167 micro controller subsystems for sensor interfaces and actuator drivers. The communication between the PC and the micro-controllers is via CAN bus. The PC communicates with other robots or a central PC via wireless LAN.

Our vision system relies on the well-known Newton Lab's Cognachrome system for ball and goal detection. Since it is mounted on a 360 degree panning unit, we are able to perform “radar-like” scans of the robots surrounding. The angle encoder of the panning unit delivers a precise relative angle of each camera picture.



3 Software Architecture

Our approach to robot programming is based on Dual Dynamics (DD) [1], a mathematical model for robot behaviors which we developed. It integrates central aspects of a behavior-based approach with a dynamical systems representation of actions and goals. Robot behaviors are specified through differential equations, forming a global dynamical system made of behavior subsystems which interact through specific coupling and bifurcation-induction mechanisms. Behaviors are organized in levels where higher levels have a larger time scale than lower levels. Since the activation of behaviors (activation dynamics) is separated from their actuator control laws (target dynamics), we named our approach “Dual Dynamics”. An important feature of DD is that it allows for robust and smooth changes between different behavior modes, which results in very reactive, fast and natural motions of the robots. Through the distribution of the overall control task on several simple controllers, we obtain a very robust overall system behavior.

We couple the behavior systems of the robots by a team communication mechanism. This allows to establish real-time point-to-point connections between all robots of a team. Since the behavior systems of the robots are specified as an interrelated set of DD-models, we are able to identify in advance the data communication flow occurring during run-time. This allows to synthesize a dedicated communication layer which efficiently distributes shared variables of several behavior systems between the robots without any unnecessary protocol overhead. The variables shared between different behavior systems are simply selected in the graphical specification of a DD-model (see below).

4 Design Environment

The successful design of robot software requires means to specify, implement and simulate as well as to run and debug the robot software in real-time on a team of physical robots. The integrated Dual Dynamics design environment [2][3] we develop and utilize allows to specify behavior systems as DD-models on a high-level of abstraction and to synthesize all code artifacts required to make these models operative in practice: a simulation model, a control program for a real robot including the team communication layer and set-up parameters for real-time monitoring and tracing.

Specify. The specification and code generation tool *DD-Designer* comprises a graphical editor to enter the specification of a DD-model in terms of sensors, actors, sensor filters and behaviors. Sensor filters and behaviors of a model are further detailed using the equation editor of DD-Designer. Since the robots of our team operate different behavior systems, DD-Designer supports concurrent development of a set of behavior systems. This includes the specification of team communication between these different models. We use a multi-target code generator to refine DD-models to a hyperlinked, indexed HTML documentation and all implementation code required by the simulator *DDSim*, the robots and the real-time monitoring tool *beTee*. DD-Designer is based on a framework generated from a high-level object-oriented meta-model specification [4].

Simulate. The Java simulator *DDSim* is specifically tailored to simulate a team of robots with different behavior systems on a RoboCup field. The sensor equipment of each robots may be different and is flexibly specified by a configuration file (using XML format). The simulation models of the different robots are Java classes which are completely generated by DD-Designer.

Run. The code for the real robot implements the DD-model in C/C++ code. This code is again directly derived from the high-level specification edited in the DD-Designer. Since both artifacts, simulation model and robot control program, are derived from the same specification, we avoid all problems that occur if a migration from a mathematical simulation model to a robot control program has to be performed manually.

Test/Analyze. The real-time trace tool *beTee* allows to capture and analyze internal variable states of an implemented DD-model in real-time [5].

5 Behavior Skills

Dual dynamics behavior systems use symbolic sensors to represent the percepted environment of the robot. We do not maintain a global world model. Self localization is performed based on odometry and gyroscope data. Since these data is noisy and subject to be disturbed by bumping robots or slipping wheels, we compensate odometry errors by improving the self-localization of our robots using *weighted* Monte Carlo sampling [2]. This approach readjusts accumulated odometry data using heading and distance of the goals as obtained by our vision sub-system and the angle encoder of the panning unit.

In one of our designed behavior models, a neural network is used to anticipate whether the ball will be lost in near future. Kicking is then triggered by the behavior system dependent on the pose of the robot and the activation of its behaviors.

A second behavior is based on a nonlinear control law for the unicycle kinematic model. Such law is designed to steer the vehicle on a static or dynamic target (the ball) along a specified direction (the opponents goal). If the target is static (still ball) the control signals are smooth in their arguments and the solution guarantees exponential convergence of the distance and orientation errors to zero. The major advantages of this approach are that there is no need for path planning and, in principle, there is no need for global self-localization either.

The behavior system of the goalie consists essentially of a two-dimensional controller, which tries to maintain a fixed distance to the back of the goal and a certain angle to the visible ball [6]. If the robot should be hit by opponent robots thus losing its position in front of the goal, a homing behavior is triggered in order to recover the correct position and the keep goal behavior is thus restarted.

All these behavior systems are developed, simulated and tested using the DD-Design Environment. Although our environment was originally targeted to design control systems using the Dual Dynamics architecture, it proved to be flexible enough to specify and simulate “classical” controllers and to integrate them seamlessly into a behavior-based Dual Dynamics architecture [6].

References

[1] H. Jaeger and Th. Christaller. ’Dual dynamics: Designing behavior systems for autonomous robots’, *Artificial Life and Robotics*, 2:108-112, 1998

[2] A. Bredenfeld, T. Christaller, H. Jaeger, H.-U. Kobialka, P. Schöll, ’Robot Behavior Design Using Dual Dynamics’, *GMD-Report Nr. 117*, 2000

[3] A. Bredenfeld, T. Christaller, W. Göhring, H. Günther; H. Jaeger, H.-U. Kobialka, P. Plöeger, P. Schöll, A. Siegberg, A. Streit, C. Verbeek, J. Wilberg, ’Behavior engineering with "dual dynamics" models and design tools’, in *RoboCup-99: Robot Soccer World Cup III* / M. Veloso, E. Pagello, H. Kitano (Editors), LNCS 1856, Springer, 2000

[4] A. Bredenfeld, ’Integration and Evolution of Model-Based Prototypes’, *Proc. of the 11th IEEE International Workshop on Rapid System Prototyping (RSP 2000)*, Paris, France, June 21-23, 2000

[5] H.-U. Kobialka, P. Schoell, ’Quality Management for Mobile Robot Development’, *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, Venice, Italy, July 25-27, 2000

[6] A. Bredenfeld, G. Indiveri, ’Robot Behavior Engineering using DD-Designer’, *IEEE International Conference on Robotics and Automation (ICRA 2001)*, Seoul, May 23-26, 2001