GMD-Robots

Ansgar Bredenfeld, Vlatko Becanovic, Thomas Christaller, Horst Günther, Giovanni Indiveri, Hans-Ulrich Kobialka, Paul-Gerhard Plöger, and Peter Schöll

Fraunhofer Institute for Autonomous Intelligent Systems (AIS)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
`bredenfeld@ais.fraunhofer.de`
`http://www.ais.fhg.de/BE`

1 Introduction

The overall research goal of our 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, we started in 1998 to develop a proprietary fast robot platform and in parallel the development of an integrated design environment.

This team description gives some details on the current state of our hardware platform (September 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 3 degree of freedom platform (2 actuated wheels and a 360 panning camera). 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. 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 two finger ball guidance. Fig. 1 shows two of our robots.

The goalie has a slightly different sensor configuration with respect to the field players; namely the infrared range detectors are all mounted pointing towards the inside of its own goal rather than in the forward direction. In addition, the kicking device of the goalie has no ball guidance but a simple plate to kick the ball.

The computer system of each 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 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.

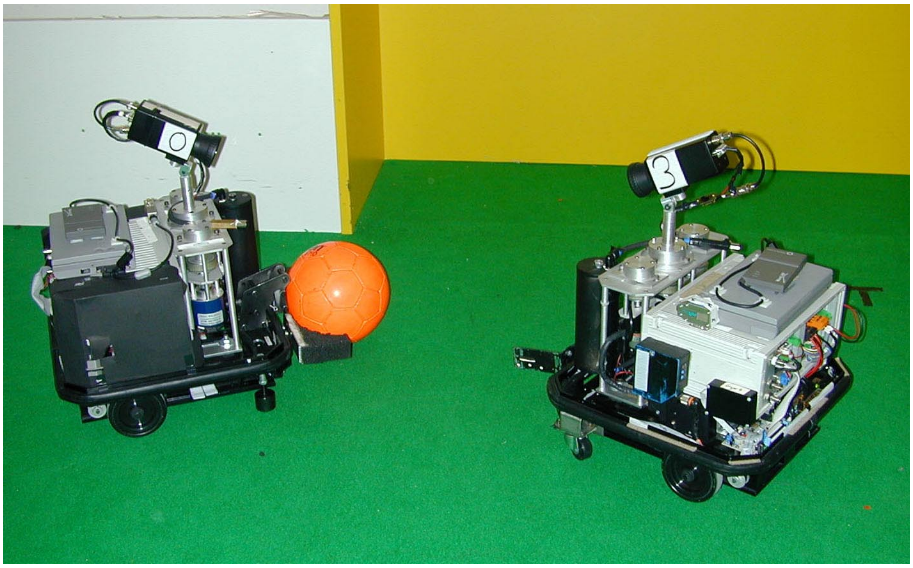

Fig. 1. Two robots of the GMD-Robots team.

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 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 variables shared by 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. Our integrated design environment [2][3] 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 the real robot including the team communication layer and the configuration files necessary to the real-time monitoring and tracing tools.

Specify: The DD-Designer specification and code generation tool 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 have 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 virtual RoboCup field. The sensor equipment of each robot may be different and is flexibly specified by an XML configuration file. The simulation models of the robots are Java classes generated by DD-Designer.

Run: The code for the real robot implements the DD-model in C/C++ language. 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 trace and monitoring tool beTee allows to capture and analyze internal states of a simulated or real robot 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 this 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 re-adjusts accumulated odometry data using the positions of the two goals relative to the robot as estimated by the vision sub-system.

In one of our behavior models, a neural network is used to anticipate whether the ball will be lost in the 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 system is based on a nonlinear control law for the unicycle kinematic model [6]. 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 [7]. 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.

All these behavior systems are developed, simulated and tested using our integrated design environment. Although the 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 architecture [7].

References

1. Jaeger H. and Christaller Th.: Dual dynamics: Designing behavior systems for autonomous robots. Artificial Life and Robotics (1998) 2:108-112
2. Bredenfeld, A., Christaller, Th., Jaeger, H., Kobialka, H.-U., Schöll, P.: Robot Behavior Design Using Dual Dynamics. GMD-Report Nr. 117 (2000)
3. Bredenfeld, A., Christaller, Th., Göhring, W., Günther, H., Jaeger, H., Kobialka, H.-U., Plöger, P., Schöll, P., Siegberg, A., Streit, A., Verbeek, C., Wilberg, J.: 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-Verlag (2000)
4. Bredenfeld, A.: Integration and Evolution of Model-Based Prototypes. In Proc. of the 11th IEEE International Workshop on Rapid System Prototyping (RSP 2000) (2000) 142–147

5. Kobialka, H.-U., Schöll P.: Quality Management for Mobile Robot Development. In Proc. of 6th International Conference on Intelligent Autonomous Systems (IAS-6) (2000) 698–703
6. Indiveri, G.: On the motion control of a nonholonomic soccer playing robot. In this book.
7. Bredenfeld, A., Indiveri G.: Robot Behavior Engineering using DD-Designer. In Proc. of IEEE International Conference on Robotics and Automation (ICRA 2001) (2001) 205–210