Team Description of AIS-Musashi 2003

A. Bredenfeld[1], V. Becanovic[1], Th. Christaller[1 2], I. Godler[5], M. Hülse[1], G. Indiveri[4], K. Ishii[3], J. Ji[1], H-U. Kobialka[1], N. Mayer[2], B. Mahn[1], H. Miyamoto[3], A.F.F. Nassiraei[2], S. Olufs[1], F. Pasemann[1], P.-G. Plöger[1], P. Schöll[1], M. Shimizu[3], K. Zahedi[1]

[1] Fraunhofer Institute AIS, St. Augustin, Germany,
[2] GMD-Japan Research Laboratory, Kitakyushu, Japan
[3] Kyushu Institute of Technology, Kitakyushu, Japan
[4] University of Lecce, Italy
[5] Kitakyushu University, Kitakyushu, Japan
ansgar.bredenfeld@ais.fraunhofer.de
http://www.ais.fraunhofer.de/BE/robocup/

Abstract. This paper describes the Middle-Size-League Team AIS-Musashi. The team description gives updated details on the state of the current robot platforms, the behavior control software and the design environment we use to specify, simulate, run and test our co-operating robot team. In addition, we report on first experiments with evolved neuro controllers on the new modular robot platform Volksbot.

1 Introduction

AIS-Musashi is a joint RoboCup team with partners from Japan, Italy and Germany. It has its root in the GMD-Robots team which participated in international and national RoboCup tournaments since 1998. This paper gives an overview of the status of the team with respect to new hardware modifications of the robots, i. e. a new kicking device, new software features added to the development environment and a more elaborated description of experiments with evolved neuro-controllers performed on the newly developped modular robot platform Volksbot.

2 Software architecture

Our approach to robot programming is based on Dual Dynamics [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 Dual Dynamics 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 [6].

3 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 basic concept of our integrated design environment [2] it to specify the behavior systems of the robots on a very high-level of abstraction without looking at implementation details in a specific programming language. This approach allows to automatically derive all code artifacts which are required to simulate, to run and to test a team of robots. Only the high-level specification is changed.

DD-Designer - Specify and generate. The first step in our design flow is the specification of the behavior system for the robots. We specify a robot behavior not as an isolated behavior but also consider the communication of behavior systems running on the different robots of a team. The specification and code generation tool DD-Designer comprises a graphical editor to enter the specification of a behavior model in terms of sensors, actors, sensor filters and behaviors. This includes the specification of team communication [3] between robots operating possibly different behavior models. We use DD-Designer to generate all implementations of the behavior model. This includes a simulation model for our simulator DDSim, the control program for each robot in the team and the configuration of the real-time monitoring and tracing tool beTee.

DDSim - Simulate. The robot simulator and behavior debugger DDSim is specifically tailored to simulate a team of robots with operate different behavior systems. The sensor equipment of each robot may be different and is flexibly specified in an XML-configuration file. Beside this the environment where the robot operates in can be specified which allows to use the simulator for completely different applications, e.g., in a production environment [4] or in an office environment [5].

beTee - Test and Analyze. The monitoring and test tool beTee allows to visualize the state of all internal variables of a behavior program running on a robot. This can be done on line while connecting to the simulator or the real robot, or off line while reading from a file recorded by a robot. The tool offers the functionality to observe and monitor variables as time series and spatially related to the physical environment of the robot.

4 Behavior skills

The Dual Dynamics behavior system switches smoothly between different behavior modes based on the perceived sensor information of the robot, the internal modes of other behaviors being active on the robot or on state information of other robots in the team, e.g. their perceived distance to the ball. Obstacle avoidance is based on a modified potential field approach and superimposed depending on the actual collision danger [7].

We do not maintain a global world model. Instead, we perform self-localization of our robots using weighted Monte Carlo sampling. Our approach re-adjusts accumulated odometry data using heading of the goals as obtained by our vision system.

A ball prediction that bases on regression filters has been developed. While the movement model (linear movement) is very simple, the main problem is noise filtering. While this has been mastered for the goalie (yielding a significant performance improvement), we did not succeed for the field players yet. This was because the field robots have a turning camera which together with the movement of the robot creates much noise.

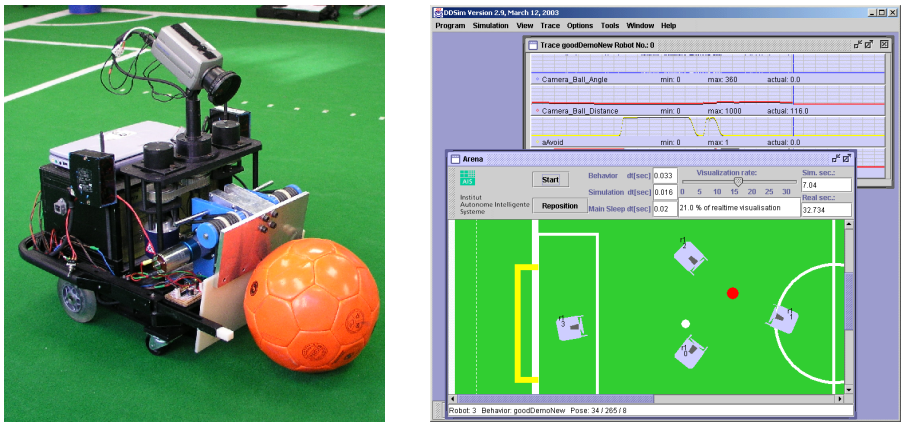

Fig. 1. Left: AIS-Musashi hardware platform (with the new kicking device). Right: Screen shot of DDSim, our simulator and behavior debugger for robot teams.

5 The VolksBot

This section introduces the new robot platform Volksbot [8] which will replace the current robot platform in future. The robot Volksbot is a modular and robust mobile robot platform developped to be used as research robot platform, for education and as basis system for industrial robotics applications. The robot

is equipped with an omni-directional vision system and, at present, with a differential drive. The modular concepts allows to easily replace the differential drive by an omni-directional drive by simply changing the lower part of the robot platform (see Figure 5).

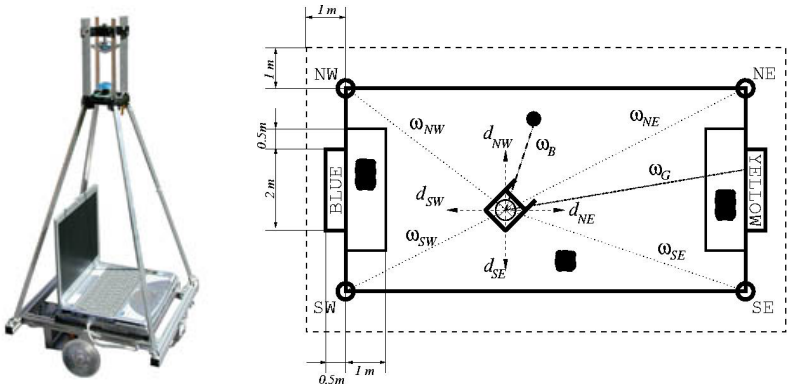

Fig. 2. The robot platform Volksbot and a sketch of the provided sensor qualities.

The main sensor of the Volksbot is an omni-directional vision system which is interfaced using Firewire to an off-the-shelf notebook. Vison processing is done on the notebook using the vision library AISVision [9]. In the following experiments with evolved neuro controllers, we only use sensor information extracted from the omni-directional view of the Volksbot. These sensor information is listed in the table 1. The standard sensor values are mapped onto intervals according to the neuron model (sec 6). Figure 5 depicts the semantic of these qualities.

sensor	**symbol**	**mapped onto**
angle of the opposite goal	ω_G	$(-0.5, +0.5] \cup -1.0$
corner posts	$\omega_{NW}, \omega_{NE}, \omega_{SW}, \omega_{SE}$	$(-0.5, +0.5] \cup -1.0$
distance to an obstacle	$d_{NW}, d_{NE}.d_{SW}, d_{SE}$	$(-1.0, +1.0)$
ball angle	ω_B	$(-0.5, +0.5] \cup -1.0$
ball distance	d_B	$(0, +1.0)$
actuators	**symbol**	**range**
left wheel	O_L	(-1.0, +1.0)
right wheel	O_R	(-1.0, +1.0)

Table 1. Sensor qualities, which are provided by the Volksbot robot.

Sensors In the following we will refer to distance sensors, ball sensors, etc. This are virtual sensors, as all information is extracted from the omni-directional view with the help of the vision library AISVision.

For obstacle avoidance we used four distance sensors. Each sensor covers 90° so that a full coverage is guaranteed. This is done in such a way that the controller can react to obstacles when driving forward or backwards. The distance sensors have a maximal range of 2 meters. The sensor qualities are mapped linearly onto the interval $[-1, 1]$, where -1 denotes no obstacle, and 1 denotes an obstacle in minimal distance to the robot. The sensors will be referred to as $d_{NW}, \ldots d_{SW}$, d_{NW} and d_{NE} (see fig. 5).

Each sensor d_i is calculated using three equal subregions $d_{i_0}, \ldots d_{i_2}$. The distance value is the mean value of the subregions $d_i = \frac{1}{3} \sum_{j=0}^{2} d_{i_j}$.

The ball distance d_B is mapped linearly onto the interval $[0, 1]$, where 0 denotes no ball seen, and 1 denotes a minimal distance to the ball. The maximal sensor range is 7 meters.

In the case that the ball is not seen, the sensor value can be mapped onto the maximal distance value, as this does not make a noticeable difference for the controller. In the case of angles this can not be done. In the case of the relative angle to the opponents goal a non visible goal would have the same sensor quality as a goal in maximal or minimal angle, that is $-\pi$ or π. In this case we must distinguish between non visible and maximal sensor qualities. Therefore the sensor quality for the opponents goal angle ω_G is mapped differently. The range of $[-\pi, \pi]$ is mapped linearly to the interval $[-0.5, 0.5]$. In the case that the angle can not be determined the value is -1.

The relative angle to the corner posts $\omega_{NW}, \omega_{NE}, \omega_{SW}, \omega_{SE}$ are mapped analogously to ω_G.

As mentioned before the Volskbot has a differential drive. The DC-motors for the left and right wheel of the Volksbot are controlled by the TMC200 motor controller [8]. The maximal speed of the Volksbot was limited to $\pm 2.5 \frac{m}{s}$.

The controller has two outputs, each controlling the velocity of one wheel. The output neurons deliver values in the interval $[-1, 1]$, where 1 denotes maximal speed forward, and -1 denotes maximal speed backwards. The output neurons are denoted by O_L, O_R, where subscripts L/R refer to the left/right wheel.

6 Evolution of the Neuro-Controllers for the Volksbot

For the controller we used a recurrent neural network, generated by our evolutionary software environment [11]. We use the standard additive neuron model with sigmoidal transfer function, in this case the hyperbolic tangent.

All evolution experiments started with an empty neural network, i.e. only the input and output neurons are defined. No synapses or hidden neurons are present. The structure of the recurrent neural network is then determined by the evolutionary process and varies constantly during this process, i.e. synapses and neurons are added and deleted, and values such as the synaptic strength or bias terms may change stochastically [14].

For the evolution of the recurrent neural networks we used a physical simulation based on the ODE library [10].

With respect to the complexity of this task we began to evolve two neuro-controllers for the subtasks "ball seeking" and "approach opponents goal". Whereby collisions with arbitrary objects which were placed randomly in the simulation have to be avoided. The resulting networks of this evolution are shown in figure 3.

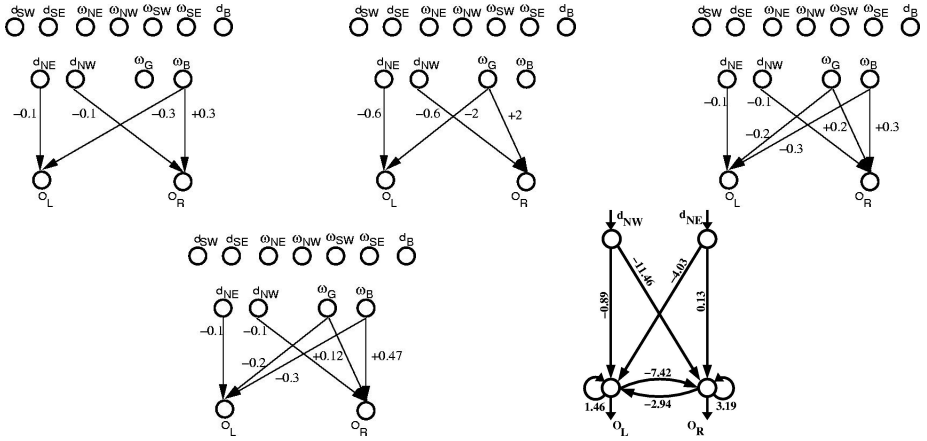

Fig. 3. Resulting networks. Upper left, resulting ball seeking network, upper center, resulting goal seeking network, upper right, hand designed resulting network for both tasks, lower left, optimized neuro-controller, using evolution on the hand designed network, lower right; MRC exploration controller.

According to the available sensor qualities and actuators the neuro-controller has 11 input neurons and 2 output neurons. Interestingly evolution came up with a neuro-controller that did not use all the presented sensor information. But of course during the evolution many controllers appeared with a larger set of used sensors and also recurrent connections. Using additional techniques we were able to reduce the neural structure to the effective minimal controllers presented here.

The structure of the two resulting networks, given by the two defined subtasks was simple enough to understand its function. The next step was to merge the two behavior so that a resulting controller would perform both tasks, ball seeking and goal seeking. The resulting hand designed network is shown in figure 3. In the resulting networks, the absolute values of the synapses from the ball angle input neuron ω_B were chosen to be larger than the synapse strength from the goal angle input neuron ω_G. This way we achieved a higher priority for the ball seeking task. The resulting controller first tries to approach the ball. If the ball angle gets smaller, i.e. the ball is in the center of the vision field of the robot, the influence of the corresponding input neuron ω_B decreases. Therefore

the influence of the goal angle input neuron is larger, so that a priority switch occurs, if the robot has performed the ball seeking task successfully. The robot now approaches the goal with the ball.

The resulting hand designed neuro-controller for both tasks was then optimized using the same evolutionary environment. Insertion of synapses and neurons was prohibited, so that only the synaptic strength of the fixed connections was modified during the process.

Having a close look at the resulting neural network, it can been seen, that a non visible ball $\omega_B = -1$ results in a spinning movement of the robot. Therefore we decided to introduce a switching mechanism. In case that the ball is not seen the controller is switched to the MRC [12][13], which performs an explorative behavior. As soon as a ball is visible again, the controller will switch back to the ball and goal seeking neuro-controller.

The evolved neuro-controllers were tested on a VolksBot which participates in games of the AIS-Musashi team. The evolved behaviors showed ball seeking behavior and approached the opponent goal with the ball. Nevertheless, this is not yet a full-fledged soccer playing behavior but gave us promising feed-back to continue our work and to tackle more complex behavior systems. In the challenge competition, the AIS-Musashi team presented the neuro controller in the free challenge part. The AIS-Musashi team achieved the 4th rank in the technical challenge competition.

References

1. Jaeger H., Christaller, Th.: Dual Dynamics: Designing a behavior system for autonomous robots, Artificial Life and Robotics (1998), 2: 108-112
2. Bredenfeld, A., Indiveri, G.: Robot Behavior Engineering using DD-Designer, in Proc. of IEEE International Conference on Robotics and Automation (ICRA 2001), Seoul (2001)
3. Bredenfeld, A., Kobialka, H.-U., Team cooperation using Dual Dynamics, In Balancing reactivity and social deliberation in multi-agent systems / Hannebauer, Markus[Eds.] (Lecture notes in computer science), (2001)
4. Baum, W., Bredenfeld, A., Hans, M., Hertzberg, J., Ritter, A., Schnherr, F., Christaller, Th., Schraft, R.D., Integrating Heterogeneous Robot and Software Components by Agent Technology, Robotik 2002, VDI-Berichte, pp. 655-660, (2002)
5. Schönherr, F., Cistelecan, M., Hertzberg, J., Christaller, Th., Extracting situation facts from activation value histories in behavior-based robots, In: KI 2001: (Lecture notes in computer science), (2001), pp. 305 - 319
6. Kobialka, H.U., Jaeger, H., Experiences Using the Dynamical System Paradigm for Programming RoboCup Robots, AMiRE 2003, February 2003
7. Kobialka, H.U., Becanovic, V., Speed-Dependent Obstacle Avoidance by Dynamic Active Regions, RoboCup 2003, Lecture Notes in Computer Science.
8. Wisspeintner, T., Mobile Roboterplattform, www.volksbot.de, 28.4.2003.
9. Olufs S. Realtime Color-Segmentaion of fast moving objects. Dipl. Thesis University of Applied Sciences Bonn Rhein Sieg, Department of Applied Computer Science, 2002

10. Smith, R., ODE - open dynamics engine; http://opende.sourceforge.net/, 7.5.2003.
11. Zahedi, K., Hülse, M., ISEE - Intergated Structure Evolution Environment, www.ais.fraunhofer.de/INDY/applications/ISEE/isee.html, 13.5.2003.
12. Hülse, M., Pasemann, F. Dynamical Neural Schmitt Trigger for Robot Control, J. R. Dorronsoro(Ed.): ICANN 2002, LNCS 2415, Springer Verlag Berlin Heidelberg New York, pp. 783–788, 2002.
13. Hülse, M., Zahedi, K., Pasemann, F. Representing Robot-Environment Interactions by Dynamical Features of Neuro-Controllers. Anticipatory Behavior in Adaptive Learning Systems, LNAI, Springer.
14. Pasemann, F., Steinmetz, U., Hülse, M., and Lara, B., Robot Control and the Evolution of Modular Neurodynamics, Theory in Biosciences, 120, pp. 311–326, 2001.