THE DISTRIBUTED ARCHITECTURE FOR LARGE NEURAL NETWORKS (DISTAL) OF THE HUMANOID ROBOT MYON

Manfred Hild, Christian Thiele, and Christian Benckendorff
Neurorobotics Research Laboratory, Department of Computer Science
Humboldt-Universität zu Berlin, Germany
{hild, thiele, benckend}@informatik.hu-berlin.de

Keywords: Neural Network Architectures; Distributed Systems; Modular Implementations; Humanoid Robotics

Abstract: Humanoid robots are complex systems that require considerable processing power. This applies both for low-level sensorimotor loops, as well as for image processing and higher level deliberative algorithms. We present the distributed architecture DISTAL which is able to provide the processing power of large neural networks without relying on a central processor. The architecture successfully copes with runtime-metamorphoses of modular robots, such as the humanoid robot MYON, the body parts of which can be detached and reattached during runtime. We detail the implementation of DISTAL on 32-bit ARM RISC processors, describe the underlying neural byte-code (NBC) of neurons and synapses, and also depict the graphical application software BRAINDESIGNER which releases the user from program coding.

1 INTRODUCTION

The most sophisticated autonomous robot can only be as useful as its underlying control architecture, which conducts everything from low-level sensorimotor loops to visual processing and highest level behavioral decision making. Usually, hybrid architectures are used to cope with the different constraints, e.g. simple, but highly reactive reflex loops for motion control, versus non-time-critical processing of complex and large decision trees.

There is a long history of architectures for robot control and abstract reasoning alike. A well-known early architecture, e.g., is SOAR, as described in (Laird et al., 1987). SOAR can be considered a set of principles and constraints on processing, for the construction of cognitive models. It is focusing rather on problem solving and learning than on highly reactive robot control. Ten years later then came, amongst others, AuRA (Arkin and Balch, 1997) and SAPHIRA (Konolige and Myers, 1998), the latter already being especially designed for autonomous mobile robots. Some architectures have been application oriented, like BERRA for service robots, as described in (Lindstrom et al., 2000). An evaluative survey of architectures for mobile robots up to the year 2003 is given in (Orebäck and Christensen, 2003).

Along with the continuous increase of processing power, versatile approaches appeared which today can be run on various robot platforms, even though their underlying processor hardware differs considerably. A widely-used open-source framework is URBI (Baillie, 2005), which can be equally well used to control Sony's AIBO, Aldebaran's NAO, or LEGO's Mindstorm NXT robots – just to name a few. Unfortunately, even today, URBI still depends on the presence of a single processor on-board a robot, since URBI always only outputs a single executable. Architectures which are state-of-the-art (Amoretti and Reggiani, 2010; Heintz et al., 2010; Hawes and Wyatt, 2010; Balkenius et al., 2010; Mitchinson et al., 2010; Martínez-Barberá and Herrero-Pérez, 2010) mostly support and encourage distributed processing. An up-to-date survey is given in (Hülse and Hild, 2010).

In the paper at hand, we introduce the distributed architecture for large neural networks DISTAL, which goes beyond present architectures since it supports runtime-metamorphoses of the robot on which it is run. As has been shown in (Hild, 2007), purely neural control of a complex humanoid robot does not necessarily result in limited behavioral capabilities. In the rest of the paper we will illustrate the advantages of DISTAL regarding its ease of use and implementation. We first present the experimental infrastructure and explain the system architecture of the modular humanoid robot MYON. Next, we detail the implementation of DISTAL and the corresponding application software. Finally, a whole-system example is given.

2 EXPERIMENTAL INFRASTRUCTURE

When designing a control architecture, one has to respect not only the computational paradigm, but also the potential target platforms and use cases. DISTAL has been devoted to large neural networks that are meant to be run on distributed processing nodes in varying scientific experimental contexts.

2.1 The Modular Robot MYON

Although DISTAL has successfully been run on different robot platforms, we will focus on the robot MYON in what follows, since this robot has been designed with the DISTAL architecture in mind and therefore reinforces the main concepts.

The robot MYON is shown in Figure 2. All in all, it is 1.25 m tall, weighs 15 kg, and consists of six body parts (head, torso, arms, and legs) which are fully autonomous in terms of energy supply and processing power. DISTAL respects this architecture and supports runtime-metamorphoses, i.e., reconfigurations of the overall body plan. The robot exhibits 32 degrees of freedom and 48 actuators. Joints which need a large amount of torque, e.g. the knee, are driven by several actuators in parallel, using series elasticities. Besides the camera, there are over 200 sensor values of the following types: joint angle, motor angle, motor current, motor temperature, acceleration force, contact force, battery voltage. The mechanical construction, energy supply concept and other hardware aspects are detailed in (Hild et al., 2011b).

Figure 1: All body parts of the robot MYON can be detached and reattached during runtime. Here, the head has been replaced by the robot's left arm.

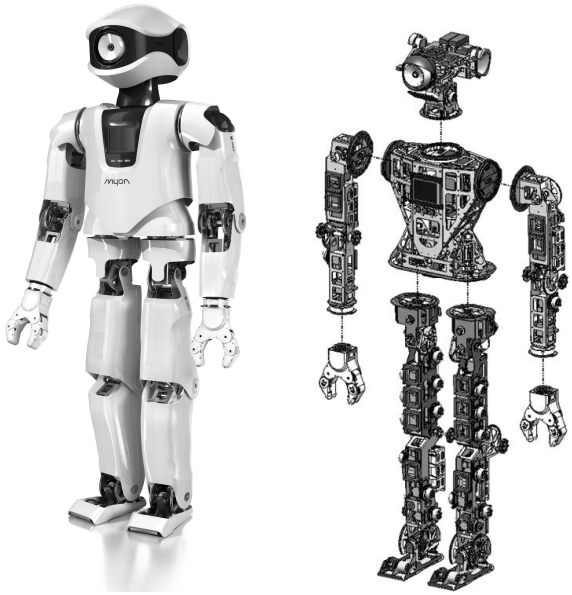

Figure 2: The modular humanoid robot MYON. Left: Image of the functional robot including the exoskeleton shells. Right: Overview of the robot's detachable body parts.

2.2 Use Cases and Operational Modes

Aggregating the experiences with former robotic platforms over the years (Spranger et al., 2010), an assortment of typical use cases could be identified and taken into account, both for the design of the robot MYON and the DISTAL architecture.

The most predominant experimental infrastructure obviously consists of a fully or partly assembled robot, which is connected to a PC. Highest flexibility is achieved when the PC is within the sensorimotor loop, so structural changes, as well as parameter changes, can be realized on the fly. Since the robot's processing power is not used, the corresponding operating mode is called *transparent mode*. The application software will have to cope with unforeseen robot morphologies, e.g. with the one shown in Figure 1.

When experimenting with self-explorative algorithms cables may hinder free movements. Thus, one needs to be able to deploy the neural network at hand permanently to the robot's processing nodes. This process we call *deployment*. After deployment, it should still be possible to monitor and log sensorimotor data as well as internal behavioral states (called *sniffing*). Also helpful, especially during presentations, are standard audio-visual signals which are provided by the robot in stand-alone scenarios, i.e. without any PC. Surely, this also has to be supported by DISTAL. Often, program-debug cycles hinder experimenting, so a graphical network editor is wishful.

3 SYSTEM ARCHITECTURE

The humanoid robot MYON exhibits several unique architectural characteristics. Here, we just give a brief summary of the processing nodes and the communication bus between them. An overall diagram of the system architecture is given in Figure 3. All processing nodes are connected using the so-called SPINALCORD, which is a multi-core bus that transfers energy, sensorimotor data at a rate of 4.5 MBaud, and a control signal which is used to switch the robot on and off.

3.1 Processing Node ACCELBOARD3D

Data processing is predominantly done by 25 processing nodes, which are distributed all over the robot's body. They are called ACCELBOARD3D, since they also possess a 3-axis acceleration sensor, despite the Cortex-M3 ARM RISC processor running at 72 MHz.

Up to four actuators are connected to each ACCELBOARD3D. The actuators are all of the type Robotis RX-28. Whenever several actuators drive the same joint, all of them are connected to the same ACCELBOARD3D. Also, the corresponding sensory data (angular sensors of the joint and all motors; motor current sensors) is sent to the same processing node, so local processing of antagonistic control paradigms can easily be realized. Those situations are automatically detected by the application software BRAINDESIGNER during the deployment process.

Each ACCELBOARD3D also exhibits a mode button and two status LEDs. This is extremely helpful for diagnosis, inspection of internal states which would otherwise be hidden to the user, and switching of operational modes like start, stop and the like.

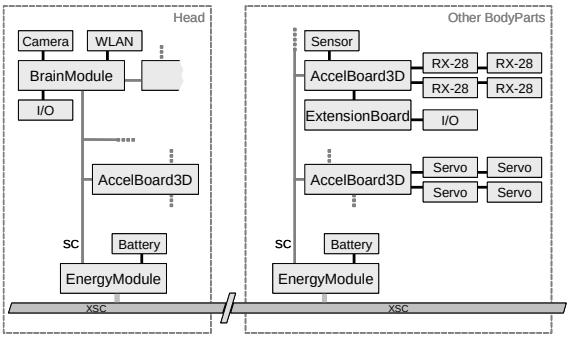

Figure 3: System architecture of the robot MYON. Components within each body part are connected via the so-called SPINALCORD (SC), whereas the body parts are connected by the EXTENDEDSPINALCORD (XSC) which includes lines for energy transfer.

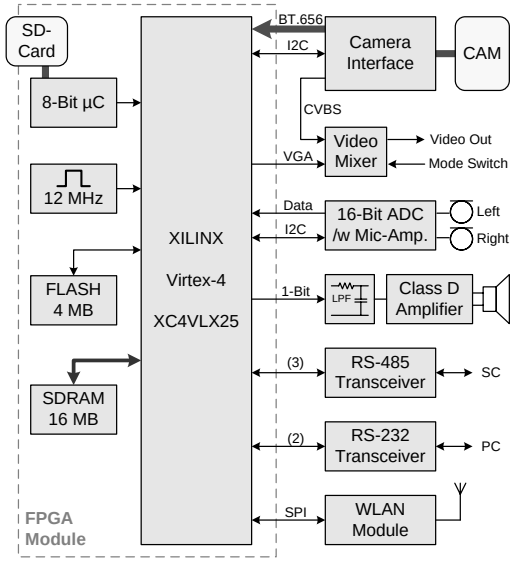

Figure 4: On the one hand the BRAINMODULE is just another processing node of the DISTAL architecture, but on the other hand it possesses considerably more processing power than an ACCELBOARD3D. This is needed for the audio-visual processing inside the robot's head.

3.2 Processing Node BRAINMODULE

As the name already indicates, the BRAINMODULE is a special processing node inside the robot's head. When sniffing the SPINALCORD, the BRAINMODULE is indistinguishable from an ACCELBOARD3D, but as can be seen in Figure 4, the BRAINMODULE possesses enough processing power to do serious audio-visual processing, e.g., a Hough-Transform.

Along with the digital camera interface, there is a special analog video mixer which allows for video keying and overlaying. This is helpful not only during presentations, but also for standard lab situations, where one wants to see the original camera image with the processed visual data superimposed. A simple overlay, e.g., shows a cross hair which indicates the object that the robot is currently investigating. Since this is all done fully synchronously, the researcher can detect the slightest deviation from the expected behavior. When using the wireless interface to monitor all data on a PC, the resultant quality and reactivity is by far lower, due to the restricted bandwidth.

Configuration of the XILINX Virtex-4 field programmable gate logic (FPGA) is done by an 8-bit microcontroller via a standard MiniSD-Card that contains the necessary FPGA bitfile. Future implementations may also use the MiniSD-Card to log sensorimotor and visual data during autonomous stand-alone scenarios without the use of a PC.

4 IMPLEMENTATION

The DISTAL architecture is a realtime framework in the hard sense, i.e., at any time data processing is bound within prescribed time limits. This can easily be monitored using a standard oscilloscope which is hooked to the SPINALCORD. In order to achieve highest performance, we introduced a neural bytecode (NBC) which almost directly translates into compact machine code for the 32-bit ARM RISC processor of the ACCELBOARD3D. In the following, we address these two main concepts which constitute DISTAL.

4.1 SPINALCORD

All processing nodes communicate with each other one hundred times a second using the SPINALCORD. Therefor, each participant has a designated time slot, during which it sends its data. For the rest of the communication time, it receives the data from all the other connected participants. The starting time of a slot is relative to the starting time of the participant with the lowest ID, which has the role of a master and triggers the 10 ms pattern. The whole timing is shown in Figure 5.

The communication on the robot MYON lasts 3.36 ms, which leaves 6.64 ms for the calculation of neural networks and the acquisition of sensor values before the next slot starts. Up to 32 participants are intended, whereof six are the energy modules of the six body parts, which have a shorter time slot than the others, because they only disclose the charge status of the batteries. The slots of all other participants last 125 μs each, during which they send 27 words (16-bit values). The first word is reserved for a synchronization value (0x5555), and five bits of the second word contain the ID of the participant.

As already mentioned before, the morphology of the robot can change, and therefore new participants can join during runtime. A new participant initially listens some hundred milliseconds and then joins the communication at the correct time. It is even possible that the new ID is lower than the ID of the current master, which leads to a new master. The old one automatically becomes a slave when it receives data from a new master before its own slot. If the master is removed, the second lowest ID will recognize this situation, become the master and the communication continues seamlessly. If the BRAINMODULE is connected to the SPINALCORD, it is automatically the master because it has the lowest possible ID, namely zero. It gradually synchronizes the SPINALCORD to the 50 Hz signal of the camera, leading to time-consistent sensory data (regarding SPINALCORD and camera data). It is possible to shift the communication time by nearly 125 μs per 10 ms slot by starting the communication later, near the end of the slot. Because of a 2.5 μs dead time at the beginning of each slot, moving backwards is possible, too.

The 25 words after the synchronization word and the ID contain sensory data and designated fields for motor control voltages, as well as free slots, which can be used by neural networks for the communication between different processing nodes.

4.2 Neural Byte-Code (NBC)

A neural network executed by the ACCELBOARD3Ds consists of several *calculation units*. On the one hand code representing the network topology, and on the other hand the code of the calculation units, is compiled for the ARM processor technology used on the processing nodes of MYON.

For each unit, a so-called preamble is compiled, which fills the registers of the processor with values according to the network topology. After that, a branch is taken to the compiled code of the unit (a *snippet*). The code for each snippet uses the given values to calculate new output values. In addition to these values, two free registers are available for temporary calculations. The commands of the NBC are similar to those available in the ARM instruction set, e.g., a command for signed saturation exists. A sample code for a weighted synapse reads as follows

```
mul   V0, Input, w
write Output, V0
```

where the first line multiplies the input value with a parameter w and puts the result into the register V0 (which is R8 on the ARM processor), whereas the second line writes this register value to the output.

Each of the calculation units consists of *inputs* and *outputs*, *parameters* (constant for each instance) and *internals* (non-volatile values, which are not accessible from outside). *Outputs* and *internals* together are called *locals* and are represented as a large array in the RAM of the processor (see Figure 6); *parameters* are put directly into the unit's preamble.

The calculation order of the units is important in time-discrete neural networks. If a neuron would be calculated before the output value of a synapse is updated, one would get a different behavior than vice versa. Therefore all snippets are executed in a given order. This is simply done by ranking: Snippets starting with "200:" are calculated before snippets starting with "300:". The calculation order of snippets with the same number is undefined. A calculation module can contain multiple snippets executed at different times. Snippets using a time indicator lower than 100 are executed only once at the beginning, thus enabling initialization code.

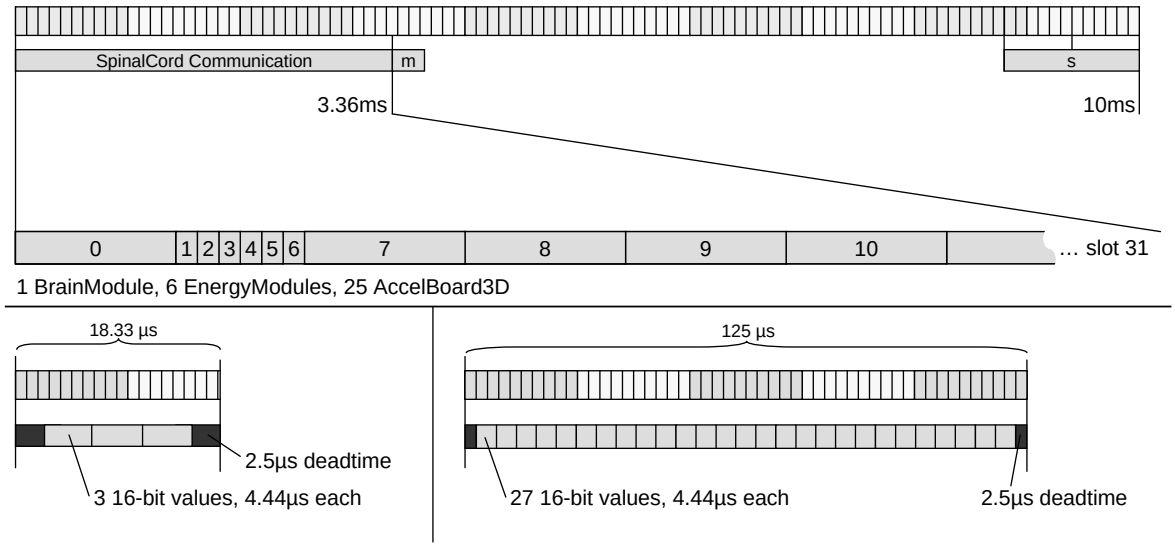

Figure 5: Timing layout of the robot MYON for a 10 ms time slot: During the first 3.36 ms all data is communicated between up to 32 processing nodes (*SpinalCord Communication*), than there are almost 6 ms for motor control and calculation of the neural network. At the end of the time slot, immediately before communication takes place again, new sensor values are acquired (*s*). The *SpinalCord Communication* of MYON consists of three different kinds of participants. Every data chunk which is communicated by the BRAINMODULE or an ACCELBOARD3D is 27 words long and needs 125 µs to be transfered over the SPINALCORD, whereas the data chunks of the ENERGYMODULES are only three words long and therefore need only 18.33 µs to be transfered.

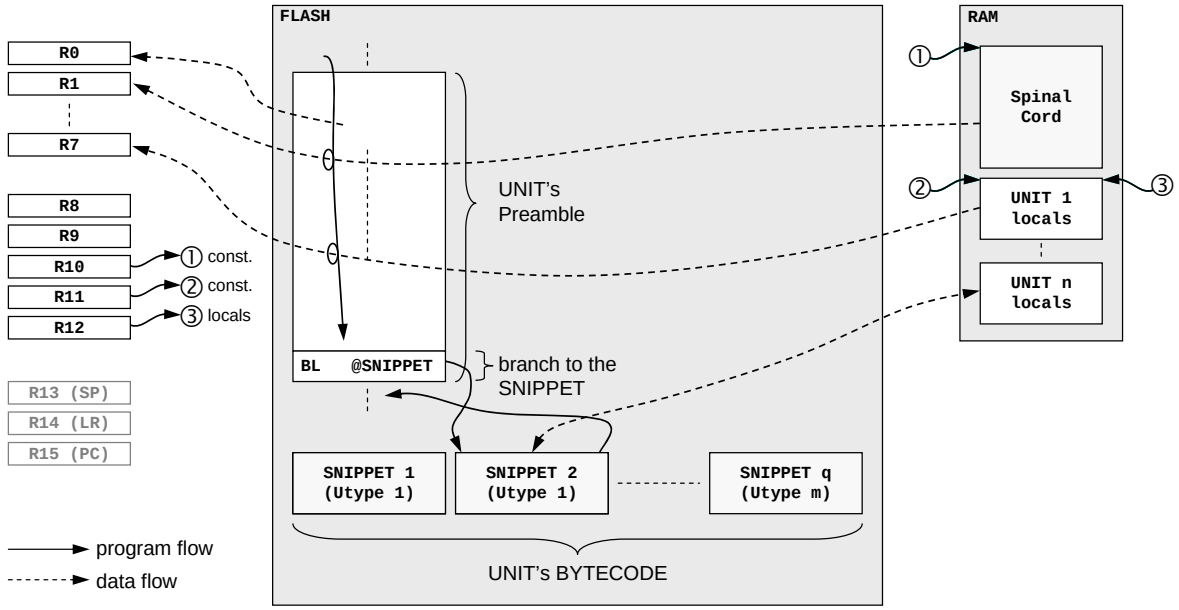

Figure 6: Concept of the execution of the neural bytecode on an ARM processor. On the left side the registers are shown, in the middle the flash memory, and on the right side the RAM. Every instance of a calculation unit has it's own preamble code (in flash), after which a branch to the corresponding snippet is taken.

5 APPLICATION SOFTWARE AND EXAMPLE

Every control architecture has to stand the test in real-world scenarios. DISTAL was used extensively on the robot MYON. The software BRAINDESIGNER was developed to create artificial neural networks for DISTAL using a graphical interface. Besides other tasks, hand-eye-coordination was successfully implemented using BRAINDESIGNER and DISTAL.

5.1 BRAINDESIGNER

The software BRAINDESIGNER offers a graphical interface for assembling neural networks on a PC, using the mouse. Several types of nodes (*neurons*) and directed edges (*synapses*) are available to assemble a network. New types of neurons and synapses can be created, which contain executable code (*Neural ByteCode*) that allows for the implementation of any kind of neural calculation or local learning process. Synapses are independent from neurons – they are called *units*, too. Assembled networks can be encapsulated and included into other networks, enabling the user to create cascaded network hierarchies.

By using special input and output nodes within the software BRAINDESIGNER, it is possible to read and write values to and from fields in the SPINALCORD.

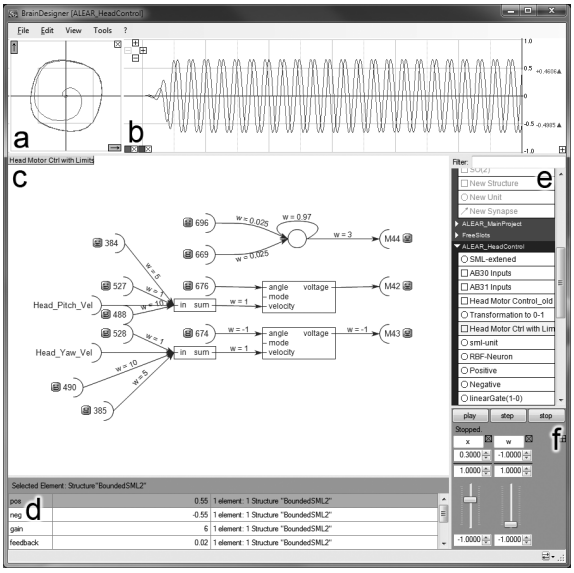

Figure 7: The software BRAINDESIGNER, with a simple neural network loaded (c). Output values can be shown over time (b) or against each other (a). At the bottom (d), parameter changes of included units and structures are possible. (e) Library of units and structures. (f) Parameters can be changed during runtime in transparent mode, using graphical sliders.

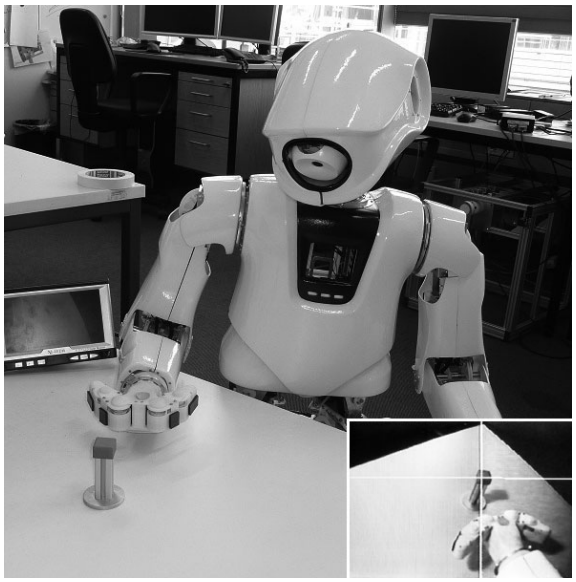

Figure 8: *Myon* sitting at a table and performing a gripping task using hand-eye coordination. In the lower right the image of the robot's camera is shown. The recognized object is shown using a cross hair.

Since all sensory data is available in the SPINALCORD, and all values needed to drive actuators are taken from the SPINALCORD, this is fully sufficient.

The user can choose from a wide range of plugins for different robots which are using the DISTAL architecture. For the robot MYON, the two operating modes *transparent mode* and *deployment mode* are available.

5.2 Whole-System Example

A hand-eye-coordination task was successfully implemented using the BRAINDESIGNER software.

In this scenario, the robot is sitting at a table and grips an object (e.g., in order to put it onto another object). The object recognition is done by the FPGA in the BRAINMODULE, using the data from the connected camera. A simple color marker is used for this purpose. In the same way, the position of the robot's gripper is identified.

Using quadric-representing neurons (Hild et al., 2011a) the arm is kept in a plane just above the table, while moving the elbow joint. The shoulder joints are used to navigate the hand into the direction of the object to grip. No world model is needed, as the position data of the objects is updated at a rate of 50 Hz.

If either the object, or the hand is not visible in the field of view, the robot's head starts to move in search for the objects.

6 CONCLUSION

We presented the distributed control architecture DISTAL along with the modular humanoid robot MYON, which seamlessly supports DISTAL. Having addressed important use cases of different experimental settings, we detailed the mechanisms of DISTAL which allow for the specific characteristics of those settings. Most important, and at the same time unique amongst humanoid robot platforms, are the ability of stand-alone operation of single limbs and the enabling of runtime-metamorphosis.

Using the appealing computational simplicity of time-discrete neural networks (the complexity of which being only bound by the number of processor nodes), we could illustrate that the proposed neural byte-code (NBC) is suitable for graphical editing of neural networks, and at the same time also almost directly translates into compact machine code for the 32-bit ARM RISC processors.

Not only did we present a theoretical framework and a corresponding computational infrastructure, but also the fully functional robot platform MYON, the accompanying application software BRAINDESIGNER, and a whole-systems example of the robot which is able to autonomously locate, grip and relocate objects by purely neural control paradigms which have been realized with DISTAL. Further research will focus on adaptive neurons and synapses, learning rules, and networks for self-explorative behavior.

REFERENCES

Amoretti, M. and Reggiani, M. (2010). Architectural paradigms for robotics applications. *Advanced Engineering Informatics*, 24(1):4 – 13.

Arkin, R. and Balch, T. (1997). AuRA: Principles and practice in review. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2):175–189.

Baillie, J. (2005). Urbi: Towards a universal robotic low-level programming language. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005)*, pages 820–825.

Balkenius, C., Morén, J., Johansson, B., and Johnsson, M. (2010). Ikaros: Building cognitive models for robots. *Advanced Engineering Informatics*, 24(1):40 – 48.

Hawes, N. and Wyatt, J. (2010). Engineering intelligent information-processing systems with cast. *Advanced Engineering Informatics*, 24(1):27 – 39.

Heintz, F., Kvarnström, J., and Doherty, P. (2010). Bridging the sense-reasoning gap: DyKnow - Stream-based middleware for knowledge processing. *Advanced Engineering Informatics*, 24(1):14 – 26.

Hild, M. (2007). *Neurodynamische Module zur Bewegungsteuerung autonomer mobiler Roboter*. PhD thesis, Institut für Informatik, Humboldt-Universität zu Berlin.

Hild, M., Kubisch, M., and Höfer, S. (2011a). Using Quadric-Representing Neurons (QRENs) for Real-Time Learning of an Implicit Body Model. In *Proceedings of the 11th Conference on Mobile Robot and Competitions*.

Hild, M., Siedel, T., Benckendorff, C., Kubisch, M., and Thiele, C. (2011b). Myon: Concepts and Design of a Modular Humanoid Robot Which Can Be Reassembled During Runtime. In *Proceedings of the 14th International Conference on Climbing and Walking Robots*, Paris, France.

Hülse, M. and Hild, M. (2010). Informatics for cognitive robots. *Advanced Engineering Informatics*, 24(1):2 – 3.

Konolige, K. and Myers, K. (1998). The Saphira architecture for autonomous mobile robots. *Artificial Intelligence and Mobile Robots: case studies of successful robot systems*, pages 211–242.

Laird, J., Newell, A., and Rosenbloom, P. (1987). Soar: An architecture for general intelligence. *Artificial intelligence*, 33(1):1–64.

Lindstrom, M., Oreback, A., and Christensen, H. (2000). Berra: A research architecture for service robots. In *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA'00*, volume 4.

Martínez-Barberá, H. and Herrero-Pérez, D. (2010). Programming multirobot applications using the thinkingcap-ii java framework. *Advanced Engineering Informatics*, 24(1):62 – 75.

Mitchinson, B., Chan, T.-S., Chambers, J., Pearson, M., Humphries, M., Fox, C., Gurney, K., and Prescott, T. J. (2010). Brahms: Novel middleware for integrated systems computation. *Advanced Engineering Informatics*, 24(1):49 – 61.

Orebäck, A. and Christensen, H. (2003). Evaluation of architectures for mobile robotics. *Autonomous Robots*, 14(1):33–49.

Spranger, M., Thiele, C., and Hild, M. (2010). Integrating high-level cognitive systems with sensorimotor control. *Advanced Engineering Informatics*, 24(1):76 – 83.