TRANSPOSING SIMULATED SELF-ORGANIZING ROBOTS INTO REALITY USING THE PLUG&LEARN ARCHITECTURE

Frank Güttler, Wolfgang Rabe, Jörn Hoffmann, Martin Bogdan
Department of Computer Engineering, Universität Leipzig, Johannisgasse 26, 04109 Leipzig, Germany

Ralf Der
Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, 04103 Leipzig, Germany

Keywords: Robotics, Self-organization, Neural networks.

Abstract: Simulations for robots like the robot simulator LPZROBOTS allow a fast proof of theoretical concepts using self-organizing neural networks. This publication presents a hardware platform as a solution to transpose these theoretical results to real robots without the time consuming reimplementation of algorithms and without the loss of computational power a standard desktop PC offers. This is shown by the example of the THREECHAINED TWOWHEELED robot which gains embodiment and shows the same emergent behaviour in comparison to the simulated counterpart.

1 INTRODUCTION

The rapid technical progress in the fields of sensor, mechanotronic, and processing technology gives rise to more and more complex robotic systems. The two major problems in these fields are first the control of such objects under complex environmental conditions and second the technical realization. The controller has to learn to maximally exploit the physical peculiarities of the body in its interaction with the environment. This can be related to the general idea of embodiment (Pfeifer and Scheier, 1999), meaning that the brain, body and environment form a common dynamical system that can not be simply divided into separate operational units (Lichtensteiger, 2003). Of course, such a complex dynamical system is convenient to nearly any kind of dynamics. Using the phenomenon of self-organization is an established objective in the embodied intelligence approach (Pfeifer and Bongard, 2006) and in the dynamical systems theory as applied to robotics, e.g. (Tschacher and Dauwalder, 2003).

Another kind of self-motivated learning approach to get is to let the robot gain maximal information about itself and its environment as first done by (Schmidhuber, 1990). From information theory, it is well known that information theoretic measures used in the sensorimotor loop is beneficial (Polani et al., 2006), (Lungarella and Sporns, 2005). Cooperative and emergent behaviour can be observed if the information measure in the sensorimotor loop is high (Ay et al., 2008; Der et al., 2008) being self-organized in an interesting matter.

The realization of agents which are able of self-organizing their behaviour forms a major challenge for the engineering of real artificial systems. However, this implies that control principles found in theory have to be proofed. For this simulations are intended, wherein the real robots are imitated, e.g. the robot simulator LPZROBOTS. This is suitable for many control principles, but it is not clarified if these can cope with robots in reality.

It's evident that the research in this field is diverged into work with simulated and real robots. Whereas using simulations, the proof of concepts are investigated quite expeditious. However, we believe that real robots make this investigations much more difficult and time-consuming. This has many known reasons, first of all that the used hardware platform of the robot is too distinct related to the simulated counterpart, e.g. that the used motors and sensors in reality are not ideal as in simulation due to their missing exact characteristic curves or the varied control of such used periphery. Additionally, it's a time consuming challenge to transfer the control principles used in the simulation to the real hardware platform.

Güttler F., Rabe W., Hoffmann J., Bogdan M. and Der R..
TRANSPOSING SIMULATED SELF-ORGANIZING ROBOTS INTO REALITY USING THE PLUG&LEARN ARCHITECTURE.
DOI: 10.5220/0003086103500357
In *Proceedings of the International Conference on Fuzzy Computation and 2nd International Conference on Neural Computation* (ICNC-2010), pages 350-357
ISBN: 978-989-8425-32-4

Unfortunately these platforms often lack computational power and furthermore need specialized code for their special units like Float-Pointing-Unit (FPU) or Digital-Signal-Processor (DSP), e.g the BunnyBot platform (Wolf et al., 2009), CotsBots (Bergbreiter and Pister, 2003) or LEGO Mindstorms (Vamplew, 2004).

The major problem is to get the sophisticated algorithms into the robots, which is in most cases impossible, e.g. information measure based theories which need an extreme computational power. For example, the Beobot platform ensures this by four 1.1GHz CPUs (Chung et al., 2010). But the realized computational power is much lower than a standard Desktop PC offers and they are not practical. Additionally, off-the-shelf analysis tools cannot be used.

The approach of the PLUG&LEARN architecture presented in this paper concerns and solves this problems by establishing an easy comparison for real robots against their simulated counterparts. This is possible due to the fact that the differences between the two sides simulation and reality are minimized to a level which has not to deal with hardware platform specific problems.

At first, in the section 2 the approach presented in this paper is outlined. After that, the section 3 clearifies the approach to obtain robots with self-organizing behaviour, including a brief introduction of the used simulator for this theoretical results. These self-organizing neural networks are used as an example for sophisticated algorithms. After the presentation of the PLUG&LEARN architecture in section 4 a comparison between a simulated robot and their real counterpart is shown by example of the THREECHAINED TWOWHEELED robot (section 5).

2 MODUS OPERANDI

In order to clearify the approach of this paper, the idea is outlined in this section as shown in Figure 1. In comparison to other robot platforms here the sophisticated algorithms are computed on the same standard desktop PC as in the simulation. In order to close the sensorimotor loop, the motor values generated by the algorithm are transferred as commands to the robot, who executes the commands. With a defined frequency, the robot transmits the sensory information to the standard desktop PC, where the algorithm can compute the next step as in the simulation. The fact that the sophisticated algorithm runs on the standard desktop PC enables all available computational power and the use of off-the-shelf-tools. For example, self-organizing architectures and research can easily be transferred from simulation to real robots without rewriting any code.

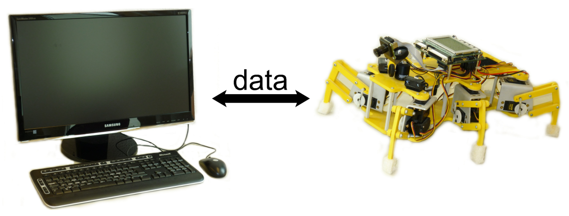

Figure 1: The outlined idea of the PLUG&LEARN architecture in a schematic overview.

3 THE GENERAL APPROACH TO SELF-ORGANIZATION

Consider a robot which produces in each instant $t = 0,1,2,\dots$ of time the vector of sensor values $x_t \in R^n$ where $x_t = (s_{t1},\dots,s_{tn})$ are n sensors, s_{ti} measuring the angle of the joint or the velocity of the motor i at time t. The controller is given by a function $K : R^n \to R^m$ mapping sensor values $x \in R^n$ to motor values $y \in R^m$

$$y = K(x) \tag{1}$$

for variables being at time t. For the example demonstrated in figure 2 as well as in section 5 we have

$$y_t = \begin{pmatrix} y^t_{left} \\ y^t_{right} \end{pmatrix} \tag{2}$$

being the target velocity of the wheel *left* and *right* and x_t the measured velocities at time t. Our controller has to be adaptive, i.e. it depends on a set of parameters $c \in R^C$, here $c \in R^2$. In the cases considered explicitly below the controller is realized by a one layer neural back propagation network defined by the pseudolinear expression

$$K_i(x) = g(z_i) \tag{3}$$

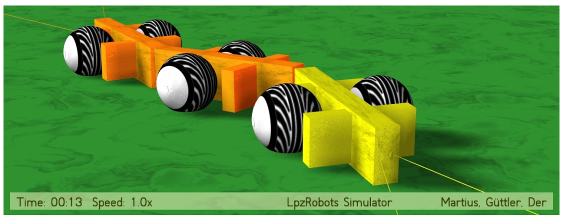

Figure 2: The THREECHAINED TWOWHEELED robot in the 3D physically realistic robot simulator called LPZROBOTS. The chain consists (here) of three robots jointed to each other. Each robot has two wheels $y_t \in R^2$ and therefore two measured velocities $x_t \in R^2$.

where $g(z) = \tanh(z)$ and

$$z_i = \sum_j c_{ij} x_j + H_i \tag{4}$$

being H a bias (threshold).

Due to the non-linearities obtained by $g(z)$ in addition with pseudo-chaotic inputs from the feedback sensors the whole system turns into very highly non-trivial system. The behaviours are generated essentially by an interplay of neuronal and synaptic dynamics.

3.1 World Model and Sensorimotor Dynamics

We assume that our robot has a minimum ability for cognition. This is realized by an additional world model $F : R^n \times R^m \to R^n$ mapping the actions y of the robot on the new sensor values, i.e.

$$x_{t+1} = F(x_t, y_t) + \xi_t \tag{5}$$

where ξ_t is the model error. The model F can be learned by the robot using any learning algorithm of supervised learning.

In the case considered below we have $x, y \in R^n$ and we assume that the response of the sensor is linearly related to the motor command, i.e. we write

$$x_{t+1} = Ay_t + B + \xi_t \tag{6}$$

where A is a $n \times m$ matrix, B a column, and ξ the modelling error. The model is learned by gradient descent as

$$\Delta A = \varepsilon_M \, \xi y^T \tag{7}$$

both ξ and y taken at time t. The model learning is very fast so that the model parameters change rapidly in time hence different world situations are modelled by relearning. Furthermore the model only is to represent the coarse response of the world to the actions y of the robot. The behavior is organized such that this reaction is more or less predictable. Hence the world model mainly is to give a qualitative measure of these response properties.

With these notions we may write the dynamics of the sensorimotor loop in the closed form

$$x_{t+1} = \psi(x_t) + \xi_t \tag{8}$$

where in our specific case

$$\psi(x) = AK(x) \tag{9}$$

3.2 Realizing Self-organization

As is well known from physics, self-organization results from the compromise between a driving force which amplifies fluctuations and a regulating force which tries to re-stabilize order in the system. In our paradigm for both, the reality and simulation, the destabilization is achieved by increasing the sensitivity of the sensory response induced by the actions given by the controller. Since the controls (motor values) are based on the current sensor values, increasing the sensitivity in this sense means amplifying small changes in sensor values over time which drives the robot towards a chaotic regime.

The counteracting incentive is obtained from the requirement that the consequences of the actions taken are still predictable. This should keep the robot in ”harmony” with the physics of its body and the environment. More details may be found in (Hesse, 2009; Martius, 2010).

The robot (see figure 2) is driven into a working regime where the noise and external forces are able to switch between the behaviours of the robot, e.g. from one direction to the opposing one. Furthermore the robot may discern between light and heavy movable objects. He is able to synchronize between other coupled robots resulting in composed emergent behaviours, see section 5.2.1. This working regime is also called the effective bifurcation point. In this region the robot executes long distance sweeps of different lengths into both directions without loosing its sensitivity to perturbations (Der et al., 2008).

3.3 The Robot Simulator LpzRobots

These and other results were mainly acquired through the 3D physically robot simulator LpzRobots developed by the *Robotics Group for Self-Organization of Control* (Martius et al., 2007). Videos about the research results gathered with the robot simulator LpzRobots can be found under (Der et al., 2010).

A main advantage compared to many other Open-Source simulators (e.g. Gazebo simulator, 2008) is that the robot simulator LpzRobots uses much more realistic rigid body dynamics provided by the Open Dynamics Engine (ODE) to achieve results close to reality (Smith, 2005). For example, LpzRobots supports the use of materials in order to get more realistic frictions. So as to obtain bodies with correct movement, the ODE considers not only contacts with collision and friction completely, it also provides common types of motors, joints and limits in a realistic manner.

The robot simulator LpzRobots is provided by the comprehensive LpzRobots package and can be found under (Martius et al., 2007). It also contains the SelfOrg package which includes the self-organizing neural networks in order to control the robots. A more detailed description of LpzRobots can be found in (Martius, 2010).

4 THE Plug&Learn ARCHITECTURE

The major disadvantage of simulators is that they cannot reflect the reality in all details. This applies also to the robot simulator LpzRobots, whereas the real robots take a major role for the research to proof the theoretical concepts gathered with the simulator. For this reason this publication presents a solution to compare these behavioural results of simulated self-organizing robots (Der et al., 2008) with their real counterparts under the premise of self-organizing behaviour.

The task transposing simulated robots into reality is usually a very long and complex way accompanied by many problems. One of a major problem is that real robots use in most cases a completely different hardware platform than the standard desktop PC running the simulation whereby the robot is controlled by e.g. a self-organizing neural network. Additionally, real hardware is not build up uniform as in the simulator, e.g. driving the motors and reading out the sensors in reality are not as easy as in simulation hence there is a real communication demand between the periphery and the used hardware. Because of the different architecture (e.g. ARM, AVR32) of the used hardware platform on the one hand the code for the neural network has usually to be entirely rewritten. On the other hand, if the code is written for the PC and robot platform in general way, the code cannot be specialized enough in order to run in realtime for big neural networks. Remember, the simulation timesteps can take as much time as needed, but in reality not. Furthermore the used hardware platform must have the needed computation power in order to control the robot with big neural networks. Though such hardware platforms consume much electrical energy which have to be supplied by the battery pack, by what either the available utilization time of the robot is shortened or the heaviness of the robot is further increased through a bigger battery pack.

Another crucial point is the restricted availability of off-the-shelf analysis tools used in the simulation, for real robots these tools are not as usable as in simulation because the data to be analyzed has to be transferred to the standard desktop PC. However, this implies that if analyzing a big neural network in realtime a huge amount of data has to be transferred via cable or remote connection. If using a cable the robot lacks mobility, whereas the use of remote connections limits the bandwidth and therefore not all data can be used for realtime analysis.

An important aspect related to the reproducibility of simulated results in reality is that these differences between a simulator and a common hardware platform are too heavy. It's very complicated to obtain the same or similar test conditions. In order to overcome all these problems mentioned the Plug&Learn architecture is introduced. The Plug&Learn architecture minimizes these differences with respect to the robot simulator LpzRobots to a level, which allows a fast transposition from simulated robots into reality.

The Plug&Learn architecture is realized by several components, both software and hardware. The software framework is directly integrated in the comprehensive software package LpzRobots which also contains the robot simulator LpzRobots.

With the Plug&Learn architecture each robot can contain many sensors and motors as shown in Figure 3. They are managed by one or more Embedded Controller Boards (ECB). A robot may consist of more than one ECB in order to extend the composed robot design. This resulting generic structure of the robots constrain a wireless communication between the ECB and the standard desktop PC where the EcbRobots package ensures that the robots are controlled by a self-organizing neural network provided by the SelfOrg framework. EcbRobots is able to manage more than one robot at the same time which is restricted only by the limited bandwidth of the wireless communication. The Plug&Learn architecture supports the use of more than one communication channel in order to extend the communication bandwidth as shown in Figure 4.

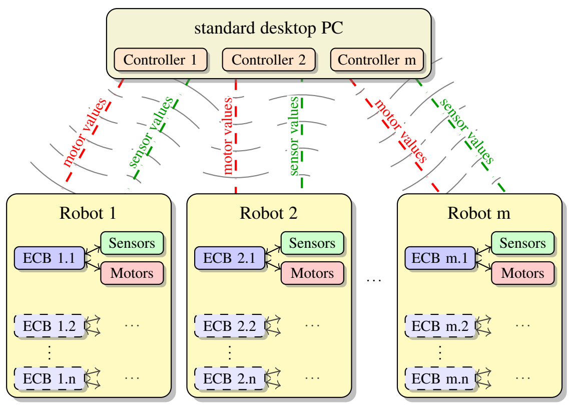

Figure 3: The Plug&Learn architecture in a schematic overview. Each composed robot contains many sensors and motors, which are managed by several Embedded Controller Boards (ECBs). The composed robot design requires that each ECB communicates wireless with the standard desktop PC where EcbRobots ensures that the robots are controlled by a self-organizing network.

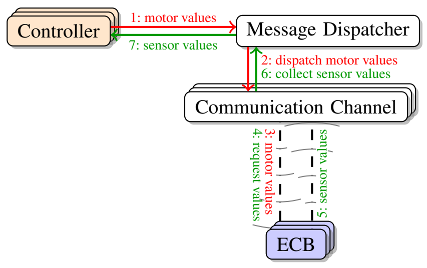

Figure 4: Schematic overview of the communication sequences in order to establish the sensorimotor loop. The Controller computes new motor values (1) which are dispatched to the Communication Channels (2). Each Channel can both serve multiple Controllers and manage multiple ECBs. It transmits the motor values to the ECB, this is done sequentially for all ECBs managed by this Communication Channel. When the time is close to the configurable reacting time, the Communication Channels transmit a "request values" message to the ECBs (4) whereup the ECBs sends the new sensor values back (5). After collecting all sensor values (6) they are delivered to the Controller (7).

Using a wireless communication leads to the challenge that the communication may be interrupted by loss of data packages which is reduced by the used underlying wireless protocol. In the case of a packet loss two methods are provided by the PLUG&LEARN architecture. First, given that some Communication Channels (see figure 4) finish earlier than other ones by reason that this channels manage less ECBs than other ones, they have enough time in order to request new sensor values from the ECB with data packet loss again. All that even goes so far as an ECB of a robot can be addressed again when other ECBs of this robot have not sent their sensor values yet. Second, if the first method does not help, the self-organizing neural network gets older sensor values which effect is discussed in section 5.3.

4.1 The Embedded Controller Board

The ECB could be marked as the heart of a robot. Not only all periphery is connected with this ECB, it can communicate wireless with other remote hardware, e.g. the standard desktop PC. It provides voltage regulators in order to supply both the ECB and the periphery. There are several busses served by the ECB which can be used to attach external periphery. In addition the ECB can read out analog sensors and drive servo motors itself. Many data links are lead through in order to clip on an extension board which is able to enhance the capabilities of the ECB in addition.

4.2 PLUG&LEARN for Motors and Sensors

One of the advantages of the PLUG&LEARN architecture is that there is no reprogramming needed if new sensors or motors are attached to the robot. The firmware running on the ECB and on other periphery like the motorboard is designed to detect and integrate new modules into the sensorimotor loop without any help.

This leads to the assumption that at least the self-organizing neural network should be reconfigured. The architecture of such networks allows the control software ECBROBOTS to configure the network once again without any help. This self-organizing neural networks are usable for any kind of robot in order to create customized behaviour based on the structure of the robot, e.g. alignment of the joints and motors (Martius, 2010). This kind of feature is so-called PLUG&PLAY brain, which reaches for real robots in combination with the PLUG&LEARN architecture a next level.

An interesting point is that the recognition of new motors and sensors does not require a restart of the self-organizing neural network. This is possible when the network is initialized with a higher number of motors and sensors than really used in the beginning. So if this maximum number is not exceeded by attaching new motors and sensors, the already computing self-organizing neural network is able to learn how to interact with them, using them for the new emerging behaviour of the robot. This takes usually less than 1 minute (Der et al., 2010). Using a higher maximum number of motors and sensors is naturally supported by ECBROBOTS, also their rescan is possible at the push of button.

4.3 PLUG&LEARN for Segmented Robot Components

While every ECB can control a bunch of motors and sensors itself, a robot may consist of more than one ECB. Exploiting their wireless communication capabilities, these ECBs do not need to be connected to each other. Thus the robots can be constructed in a generous and preferred way utilizing the modularity of the PLUG&LEARN architecture. The PC software package ECBROBOTS allows to configure which ECBs belong to one robot controlled by one self-organizing neural network. An example of such a modularized robot is the THREE-CHAINED TWOWHEELED robot driven by only one self-organizing neural network described in section 5.2.2.

5 THE THREECHAINED TWOWHEELED ROBOT

In this section the THREECHAINED TWOWHEELED robot as an example for transposing simulated self-organizing robots into reality is presented, see Figure 5. This robot uses the PLUG&LEARN architecture described in section 4 and is a composition of three individual two wheeled robots, see figure 5.

5.1 Machinery

Every two wheeled robot of the chain has a speed motor with gear mechanism for each wheel. These motors are directly controlled by a self developed motorboard connected to the ECB via the "two wire interface"-bus. This motorboard includes a current limitation in order to allow the configuration of maximum force transmitted to the wheels. If the current utilized by the motors is over a configurable defined value, the four quadrant chopper of the motorboard switches the motor into idle speed, thus wheelspins can be avoided which is discussed in section 5.3.

Additional sensors can be mounted at the robot case, e.g. infrared sensors. They put the controller in a position to recognize walls or barriers previous to a possible collision. The THREECHAINED TWOWHEELED robot has in summation 6 infrared sensors, 3 on front of the first robot and 3 on back of the last robot. The top of the cases of the robots are assigned to hold more sensory devices such as a camera. If additional sensors are attached, they are automatically recognized and integrated into the sensorimotor loop.

5.2 Application Cases

The PLUG&LEARN architecture allows the operation of two mainly different application cases, using one SELFORG controller for either each robot or all robots. These use cases are easy configured with the PC software package ECBROBOTS. Mixed use cases are also possible.

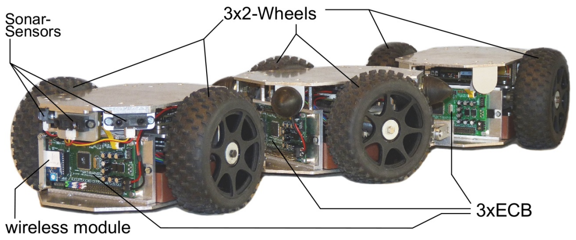

Figure 5: The THREECHAINED TWOWHEELED robot designed after the simulated counterpart. It consists of 3 robots, each one individually controlled by a separate ECB. The robot can be seen in action in the corresponding videos (Gttler et al., 2010).

5.2.1 Decentralized Control

The first configuration is obtained by the use of one SELFORG controller for each robot of the THREECHAINED TWOWHEELED robot. Therefore each controller gets a robot with 2 degrees of freedom and 2 corresponding motor speed sensors. The first and the last robot can have additionally 3 infrared sensors each, but are not considered in the preliminary comparison to the simulated results obtained in (Der et al., 2008). This results show that the synchronization effects observed with the simulated counterpart (see figure 2) can also be noticed in reality, see the two corresponding videos (Gttler et al., 2010). The robot moves not only into one direction but also keeps explorative enough in order to invert its direction of motion after some time. If the robot collides with a barrier each TWOWHEELED robot often changes the velocity resulting in that they become synchronous after some time and therefore the chain moves away from the barrier. The arising coherent motions of the chain is an emerging phenomenon based on this synchronization. This tends to explorative behaviours exploring the spatial extensions of a room the chain is put into. The generated behavioural patterns are similar in comparison with the simulation, especially the wiggly line.

5.2.2 Centralized Control

In this use case the THREECHAINED TWOWHEELED robot is controlled by only one SELFORG controller. As an result we obtain a complex composed robot with 6 degrees of freedom, 6 motor speed sensors and 6 infrared sensors. Future research studies are needed in order to investigate if similar explorative behaviour as in the decentralized case can be observed.

5.3 Crucial Points

Compared to the simulated counterpart the THREECHAINED TWOWHEELED robot produces similar explorative behaviour but with some difficulties. The first is that in the simulation the number of computation updates both of the physics engine ODE and the self-organizing controller is exact 10 ms in order to obtain a realistic simulation. This number can't and don't needs to be achieved with real robots due to the higher response times from the ECBs that approaches 15-30 ms combined with the fact that transmitting new motor values to the ECBs and then receiving the sensor values must be handled with two separate communication sequences resulting in about 10-15 updates per second for the controller, dependent on the configured reacting time for the motors. This reacting time is very important and can vary for the used motors and sensors, going up to 100 ms. As for example, the motorboard has a pid regulation to obtain the motor speed given by the self-organizing neural network. As is well known from measures, this achievement of the correct speed takes some time exceeding 100 ms. This has to be recognized for the control of real robots with self-organizing neural networks in order to get comparable test conditions of simulation and reality.

The second challenge is that the motors are very powerful and therefore wheelspins can easy occur. In the simulation the maximum force transmitted to the wheels is restricted. Additionally, the simulation allows to change the mass of the robot in order to scale the forces generated by the motors. So as to get similar behaviour in reality, the maximum force transmitted to the wheels can be restricted by the maximum motor current described in section 5.1. This is very important by reason that the controller needs a feedback through the motor speed sensors whether the robot is moving or not and recognizing some barrier the robot must circumnavigate. This feedback enables the sensitivity of the sensory response induced by the actions given by the controller discussed in section 3.2.

One problem is that the wireless communication is not always stable whereby the self-organizing neural network gets older sensor values. This is not critical because the network tries to reduce the weight of this sensors by time. As a consequence of this the network learns how to interact with the remain of the robot in order to create emergent behaviour. When the communication is stable again, the self-organizing neural network could be a little bit “irritated”, but reintegrates the motors and sensors into the self-organizing process. An example of that can be seen in the videos (Gttler et al., 2010) where one segment of the THREECHAINED TWOWHEELED robot is temporarily disabled.

Another point is the noise of the sensors. While in simulation the noise is generated by a pseudorandom function, in reality the sensors have natural noise given by physical conditions. This noise is different from the noise generated in simulation, where for example white normal noise with no shift of the centre of the gaussian curve is used. In reality the noise has in most cases a shift depending on the sensor physical properties. They are altered through e.g. manufacturing processes and therefore differ from one sensor to another one. This knowledge is important in order to understand how the synchronization effects are established, since the noise together with the controller weights is accountable for over- or understeering the motors of the THREECHAINED TWOWHEELED robot. In the simulation each sensor gets the same noise type, whereby the simulation is a more perfect world regardless to the real conditions of the reality.

Another example for having differences between the simulation and the reality is that the wheels have some end play through the gear mechanism. The outcome of this is that a touching force generated from the motors cannot be transmitted immediately to the wheels, the end play has to be passed first. This takes time and therefore the synchronization of the robots is more difficult and takes longer than in simulation because the simulation does not consider end plays. Additionally, the consideration would make no much sense since each gear mechanism has a different end play, changes in time and the gear has to be readjusted on occasion.

As a consequence for comparisons, there are differences between simulated and real robots. This are especially the different noise and mechanical properties like end play or the wheelspins which must be avoided by special electronic regulation. Due to the higher response times the self-organizing neural network cannot make as much updates as in simulation, which can delicate affect the properties of the network and therefore the control of the robot. Many differences cannot be eliminated in remote future. This is another example why the reality can never be completely reflected by the simulation. Because of this differences, experiments with real robots are essential in order to proof the theoretical results, as it is not clear by default if these theoretical concepts stand one’s ground in reality. Due to the fact that the sophisticated algorithms don’t fit to the available hardware of real robots, the PLUG&LEARN architecture avoids this problems thereby computing the algorithms on the standard desktop PC.

6 CONCLUSIONS AND FURTHER WORK

This paper presented a solution in order to transpose simulated self-organizing robots into reality using the PLUG&LEARN architecture. The architecture is designed to rapidly equip real robots with sophisticated algorithms running on a standard desktop PC. Up to the best of own knowledge, this approach is a novel one in the field of robotics.

The advantages are twofold. First, physically correct simulated robots can be compared with their real counterparts using the same "brain". Thus, no code written for the robot simulator LPZROBOTS needs to be changed or adapted regarding to self-organizing neural networks. Second, it allows to use the computational power of a standard desktop PC as well as easy to deploy extensions or off-the-shelf analysis tools.

The THREECHAINED TWOWHEELED robot example demonstrated the PLUG&LEARN architecture shows that similar behaviours can be observed in as reality as in simulation, but with difficulties. These are caused by the differences between the simulation and the reality. This implies that it is essential to use real robots in order to proof theoretical concepts gathered with the simulation. The PLUG&LEARN architecture minimizes the differences between the simulation and reality to a level whereas a comparison between simulated and real robots are possible.

As for example, the PLUG&LEARN architecture is used to drive an artificial human hand (Franke and Bogdan, 2009). The task is to control this hand with a self-organizing neural network in order to get emergent motions from which Postures can be derived. A simulated counterpart in LPZROBOTS is used to proof theoretical concepts which are then tested with the real artificial human hand.

A task for future research is to investigate if e.g. the information theoretical aspects considered in (Der et al., 2008) can also be measured for the real counterpart or the centralized control involves the same behavioural results as with decentralized control.

REFERENCES

Ay, N., Bertschinger, N., Der, R., Güttler, F., and Olbrich, E. (2008). Predictive Information and Explorative Behavior of Autonomous Robots. *The EPJ B - Condensed Matter and Complex Systems*, 63(3):329–339.

Bergbreiter, S. and Pister, K. (2003). CotsBots: An Off-the-Shelf Platform for Distributed Robotics. *IROS 2003*.

Chung, D., Hirata, R., Mundhenk, T. N., Ng7, J., Peters, R. J., Pichon, E., Tsui, A., Ventrice, T., Walther, D., Williams, P., and Itti, L. (2010). *A New Robotics Platform for Neuromorphic Vision: Beobots*, volume 2525 of *Lecture Notes in Computer Science*, pages 325–340. Springer Berlin/Heidelberg.

Der, R., Güttler, F., and Ay, N. (2008). Predictive Information and Emergent Cooperativity in a Chain of Mobile Robots. In *Artificial Life XI*. MIT Press.

Der, R., Martius, G., Gttler, F., Herrman, M., and Hesse, F. (2010). Videos of Self-Organized Robot Behavior. http://robot.informatik.uni-leipzig.de/videos.

Franke, M. and Bogdan, M. (2009). A New Lightweight, Robust and Forceful Finger for an Artificial Limb. *WC2009 - MEDICAL PHYSICS AND BIOMEDICAL ENGINEERING*, pages 339–342.

Gttler, F., Rabe, W., and Bogdan, M. (2010). Videos of THREECHAINED TWOWHEELED robot showing Self-Organized Robot Behavior. http://robot.informatik.uni-leipzig.de/Videos/PLA/TCTW.

Hesse, F. (2009). *Self-Organizing Control for Autonomous Robots. A Dynamical Systems Approach Based on the Principle of Homeokinesis*. PhD thesis, Georg-August-Universität Göttingen.

Lichtensteiger, L. (2003). The Need to Adapt and Its Implications for Embodiment. In *Embodied Artificial Intelligence*, pages 98–106.

Lungarella, M. and Sporns, O. (2005). Information self-structuring: Key principle for learning and development. In *Proceedings 2005 IEEE Intern. Conf. Development and Learning*, pages 25–30.

Martius, G. (2010). *Goal-Oriented Control of Self-Organizing Behavior in Autonomous Robots*. PhD thesis, Georg-August-Universität Göttingen.

Martius, G., Der, R., and Gttler, F. (2007). LpzRobots - Simulation Tool for Autonomous Robots. http://robot.informatik.uni-leipzig.de/software.

OpenSource-Community (2008). Player/Stage/Gazebo - free software tools for robot and sensor applications. http://playerstage.sourceforge.net.

Pfeifer, R. and Bongard, J. (2006). *How the Body Shapes the Way We Think. A New View of Intelligence*. MIT Press.

Pfeifer, R. and Scheier, C. (1999). *Understanding Intelligence*. MIT Press,Cambridge, MA.

Schmidhuber, J. (1990). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 222–227, Cambridge, MA, USA. MIT Press.

Smith, R. (2005). Open Dynamics Engine. http://ode.org/.

Tschacher, W. and Dauwalder, J. (2003). *The Dynamical Systems Approach to Cognition: Concepts and Empirical Paradigms Based on Self-Organization, Embodiment, and Coordination Dynamics*. World Scientific Publishing Company, Singapore.

Vamplew, P. (2004). Lego Mindstorms Robots as a Platform for Teaching Reinforcement Learning. In *AISAT2004*, Hobart, Australia.

Wolf, J., Vicente, A., Gibbons, P., Gardiner, N., and Tilbury, J. (2009). BunnyBot: Humanoid Platform for Research and Teaching. *Buch Progress in Robotics*, 44:25 – 33.