Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Robot control and the evolution of modular neurodynamics

by

Frank Pasemann, Uli Steinmetz, Martin Hülse

and Bruno Lara

Preprint no.: 36 2001

Robot Control and the Evolution of Modular Neurodynamics

Frank Pasemann[‡†], Uli Steinmetz[‡], Martin Hülse[‡], Bruno Lara[†]

[‡]Max-Planck-Institute for Mathematics in the Sciences
D-04103 Leipzig

[†]TheorieLabor, Universität Jena
D-07740 Jena

Abstract

A modular approach to neural behavior control of autonomous robots is presented. It is based on the assumption that complex internal dynamics of recurrent neural networks can efficiently solve complex behavior tasks. For the development of appropriate neural control structures an evolutionary algorithm is introduced, which is able to generate neuromodules with specific functional properties, as well as the connectivity structure for a modular synthesis of such modules. This so called ENS^3-algorithm does not use genetic coding. It is primarily designed to develop size and connectivity structure of neuro-controllers. But at the same time it optimizes also parameters of individual networks like synaptic weights and bias terms. For demonstration, evolved networks for the control of miniature Khepera robots are presented. The aim is to develop robust controllers in the sense that neuro-controllers evolved in a simulator show comparably good behavior when loaded to a real robot acting in a physical environment. Discussed examples of such controllers generate obstacle avoidance and phototropic behaviors in non-trivial environments.

1 Introduction

The widespread recurrent structures found in biological neural networks imply the possibility of complex neural dynamics and, in fact, oscillatory and chaotic activity has been observed frequently in brains [8], [11], [12], [25]. This suggests that complex dynamics plays an important role for specific brain functions, and especially for cognitive processes. Although dynamical approaches to cognition date back to the cybernetics era (e.g. [4]), and there is a tremendously increasing amount of specialized data available from the neuro sciences, it still remains an open question to which extent and through what kind of mechanisms oscillatory and chaotic dynamics can contribute effectively to the cognitive abilities of living organisms.

It seems obvious that also artificial neural networks with higher information processing capabilities will use complex neurodynamics of a large collection of neurons, and – as their biological counterparts [3] – a modular structure of these systems should be appropriate. Thus, the basic hypothesis of the approach to cognitive systems adopted here states that cognitive abilities of biological or artificial systems are founded on the complex dynamical properties of modular neural networks.

Following such an approach implies to understand the emergent dynamical properties of such a network in terms of interacting smaller subnetworks. But that this may become a very difficult task is suggested by the observation that already small artificial neural networks with recurrent connectivity inherit complex dynamical features [36], [31], [28]. These are, for example, variable oscillatory modes including chaos, and the existence of many different accessible modes at the same time. The dynamical properties even of small neural networks are almost unprdictable by a mathematical theory. So, even if one knows, what kind of dynamical output should be generated for a specific behavior, a desired recurrent network is almost impossible to construct - and for the general case of networks and behaviors, there are even no learning rules so far.

Furthermore, if one thinks about these small neural networks as basic building blocks for larger systems, i.e. using them as interacting modules, things become even worth, because the (recurrent) coupling of non-linear subsystems can lead to many undesired or unexpected behaviors of the composed system. But it is exactly the emergence of such unpredictable attributes on which our hope to find interesting brain structures rests. Thus, to be able to find some examples pointing to the appropriateness of the modular neurodynamics hypothesis we are interested in the development of artificial systems acting in a sensori-motor loop. Simulated or real robots which have to survive in a changing environment provide an optimal means to study network

structures and dynamics dealing successfully with situations where temporal sequences of sensor data have to be processed in a behavior-relevant way. Therefore experiments are made in the spirit of a synthetical approach to neural systems based on ideas coming from fields like embodied cognition [29], artificial life [18], [1], dynamical systems and modular recurrent neural networks. Evolutionary computation techniques will be used as has been suggested for instance in [5], [15], [10], [13], [24].

The combined application of neural networks and evolutionary algorithms turned out to be a very effective tool for general problem solving [33]. It was also shown to generate interesting classes of robot behaviors (see e.g. [23]). Instead of standard genetic algorithms, here an ENS^3-algorithm (*evolution of neural systems by stochastic synthesis*) [27] is used. It is applied to networks of standard additive neurons with sigmoidal transfer functions and sets no constraints neither on the number of neurons nor on the connectivity structure of networks. It develops network architecture and optimizes parameters like weights and bias terms simultaneously. In acquiring both network topology and parameter optimization at the same time it is similar to the GNARL algorithm [2]. In contrast to genetic algorithms it does not quantize network parameters, and it was tested successfully for some non-linear control problems [27].

For the solution of extended problems (more complex environments, more complex sensori-motor systems, more complex survival conditions, etc.) the synthesis of evolved neuromodules forming larger neural systems can be achieved by evolving the coupling structure between modules. Of course, modularity is generally supposed to be a good design strategy for artificial systems. On the other hand, investigations in the context discussed here can shed a light also on mechanisms underlying the evolutionary origin of modularity [9]. We suggest that ENS^3 is better suited for evolving modular networks for embodied cognitive systems than genetic algorithms, because it can make use of primarily undefined recurrent connectivity and therefore can chose from a practically infinite reservoir of dynamical mechanisms and features to solve survival tasks.

2 The ENS^3 evolutionary algorithm

The ENS^3 evolutionary algorithm was designed to develop neural control systems for autonomous agents acting in a given - perhaps dynamically changing - environment. In general, it is used to study two types of problems at the same time: First, what type of internal dynamical properties of neural networks are used to solve a given behavioral task. Second, what type of

(recurrent) connectivity structures are able to generate this behavior relevant dynamics. Here we use ENS^3 to control specific behaviors of Khepera robots [21].

Because ENS^3 develops structure and optimizes parameters simultaneously, one has to decide only the type of neurons to use. The number N_{in} of input and N_{out} of output units is provided by the definition of the problem. They are not changed by the algorithm. Input nodes will always be used as buffers. All non-input neurons will be supposed to be of the standard additive type with sigmoidal transfer function. But every other neuron model may be implemented as well. Nothing else is determined, neither the number of internal units nor their connectivity, i.e. self-connections and every kind of recurrences are allowed, as well as excitatory and inhibitory connections, with only one restriction: Because input units are only buffering data, no backward connections to input units are allowed. Weights are real valued and also real valued bias terms are use, but may be avoided if anti-symmetric transfer functions, like $tanh$, are used for neurons.

The ENS^3 algorithm is initialized by a population of "empty" networks; i.e. they consist only of input and output neurons without any hidden neurons and connections. Having chosen the average number n of neuromodules in a population p, the evolutionary process is determined by submitting a population $p(t)$ with $n(t)$ individuals at time t to a variation-evaluation-selection cycle, i.e. $p(t+1) = S\,E\,V\;p(t)$. The parents are copied and the copies are mutated according to the *variation operator* V. This operator can be written as a product $V = V_s \cdot V_p$, and it provides both, *structural mutations* as well as *parametric mutations*. The operator V_s acts by insertion and deletion of neurons and connections according to fixed per-neuron and per-connection probabilities. Parametric mutations V_p are realized by perturbing the values of weight and bias terms with random numbers drawn from a user defined interval due again to fixed probabilities. All this parameters can be controlled on-line by the user during the evolution process.

The *evaluation operator* E is defined problem-specific, and it is usually given in terms of a fitness function, which associates to every neuromodule a real value. After evaluating the performance of each individual network in the population $p(t)$ the number of network copies passed from the old to the new population - the off-spring - depends on the *selection operator* S. It realizes the differential survival of the varied members of the population according to evaluation results.

In consequence of this selection process the average performance of the population will tend to increase. Thus, after repeated passes through the variation-evaluation-selection cycle populations with networks solving the problem can be expected to emerge.

2.1 Evolving modular neuro-controllers

For developing modular neuro-controllers in a sensory-motor loop we will use ENS^3 as follows: Suppose for a given agent there is given a task α defined by fitness functions F_α. Let A denote a neuro-controller which solves this task. One now may want to generate an extensive behavior β, given in terms of a fitness function F_β, which presupposes that the system is able to solve task α as well. Task β may involve also additional signal inputs and outputs to be processed. Then there are three possibilities to use ENS^3 for the generation of this type B of neuro-controller:

1. Trivially, one may start from scratch using the fitness function F_β; i.e., one does not use modules A at all, and the initial population is the standard one; i.e., it has networks with no internal neurons and no connections.

2. *Restricted module expansion*: The initial population consists of modules A, with an extended number of unconnected input and output neurons if task β is demanding it. The architecture and parameters of modules A stay fixed during the following evolution process. Thus, the modules A will be fixed subnetworks of evolving larger networks B which now may use additional connections and internal neurons.

3. *Semi-restricted module expansion*: Same as module expansion, but now only the architecture of A is fixed; parameters, like synaptic strengths and bias terms are subjected to the variation operator V_p.

4. *Free module expansion*: Same as module expansion, but neither the architecture nor the parameters of modules A are fixed during the evolution process. Thus, modules A may not leave any trace in the resulting network B.

The suggestion is that semi-restricted module expansion may be most effective because the original structure pertain visible during the evolution process but their plasticity will allow a better adaptation to a newly developing structure. More generally, if module A is already a combination of submodules A_1 and A_2, then ENS^3 will generate larger systems C, containing A_1 and A_2 as subsystems. In this case ENS^3 will probably evolve couplings between neurons of modules A_1 and A_2 as well as some additional control structure. It is expected that in this way one can evolve more complex behaviors following a hierarchical sequence of more and more acquired behaviors. This is somehow in the spirit of the subsumption architecture approach [7], [29], but here different techniques are used.

The different modification strategies listed above are implemented in the ENS^3 program, and they are used to develop multi-functional neuro-controllers for Khepera robots. The resulting structures and the effectiveness of the generating processes may then be compared. An example of the restricted module expansion using ENS^3 can be found in [19]. In the following only the trivial method is explored for generating a phototropic behavior in an environment with light sources hiding behind obstacles. The resulting actions resemble obstacle avoidance with added light seeking behavior.

An especial point for the successful evolution of modular neural systems is to keep the fitness functions as simple as possible. This means that preferably only one task should be encoded in a fitness function. To evolve an extended behavior, it is therefore favorable to start with a population of modules which have already established specific functional behaviors. The "plasticity" of the underlying neural networks then should guaranee the interplay with and adaptation to the newly developing structures.

3 Evolved networks for Khepera control

The Khepera robots [21] are miniature cylindrical robots (diameter of 55 mm, figure 1). The basic configuration of the robot has 8 Infra Red (IR) sensors (6 at the front of the robot and 2 at the rear), and two wheels, each controlled by a DC motor with an incremental encoder (10 pulses per mm of advancement of the robot). Each sensor can be used in two modalities: as a proximity sensor (by emitting and measuring the reflected Infra Red light), and as a light sensitive sensor (by measuring the Infra Red component of the environment light).

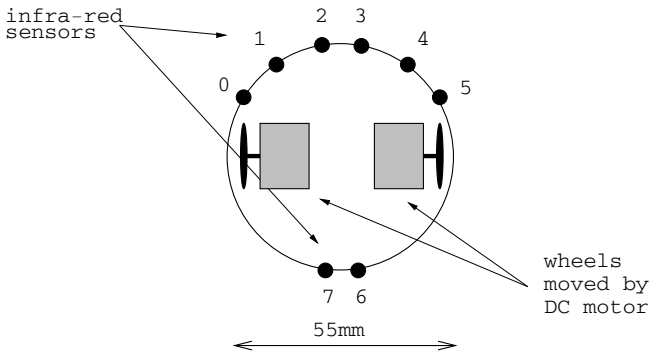

Figure 1: Principle design of the Khepera robot.

The signals of the light and proximity sensors are preprocessed for the

inputs to the neuro-controllers. The implemented preprocessings realize the following conditions: Values of proximity sensor are between -1 and 1 and increase with decreasing distance between sensor and obstacle. Light sensors generate values between -1 and 0, decreasing with increasing light intensity.

To control the two motors of the Khepera with positive and negative signals standard additive neurons with transfer function *tanh* are used; and all bias terms are set to zero. Hence, the time-discrete activity dynamics of the controller neurons is given by

$$a_i(t+1) = \sum_{j=1}^{n} w_{ij} \tanh(a_j(t)) + \sum_{k=1}^{m} w'_{ik} I_k(t) , \quad i = 1, \ldots, n , \qquad (1)$$

where n is the number of controller neurons, m the total number of inputs (i.e. buffer neurons), a_i denotes the activity of neuron i, and I_k the k-th input signal. Synaptic weights from controller units j and input units k to unit i are denoted by w_{ij} and w'_{ik}, respectively. The same neuron model is chosen for all hidden and ouput neurons of the controllers.

Following the philosophy of the artificial life and embodied cognition approaches, for fitness functions only quantities will be preferred which are accessible for the agent (robot) itself. No coordinates or observer dependent quantities should enter if possible. For the following experiments an average population size of 30 individuals is chosen. The time interval for evaluating these individuals was set to 2000 simulator time steps. There was also a stopping criterion for the evaluation of an individual: It was terminated when bumping into an obstacle. Furthermore, the size of resulting networks can be influenced by adding cost terms for neurons and connections to a given fitness functions. Experiments with the ENS^3-algorithm will be used to test such an assumption.

Environments for evolution and testing of neuromodules where constructed using the tools of the Khepera simulator [20]. For tests with the physical robots real walls and movable obstacles where used. The goal was to evolve networks in the simulator in such a way that when loaded to the physical robot they produce a comparable behavior in the physical world. This was achieved for instance by gradually increasing the number of obstacles in the simulated worlds during the evolution process, by introducing for instance walls at angles of around 45 degrees or narrow passages.

3.1 The first experiment: obstacle avoidance

The simplest behavior an autonomous robot should master is obstacle avoidance, and there exist many solutions to this problem, from Braitenberg vehicles [6] to neural network controllers [30], [14], [22]. Thus, the first goal

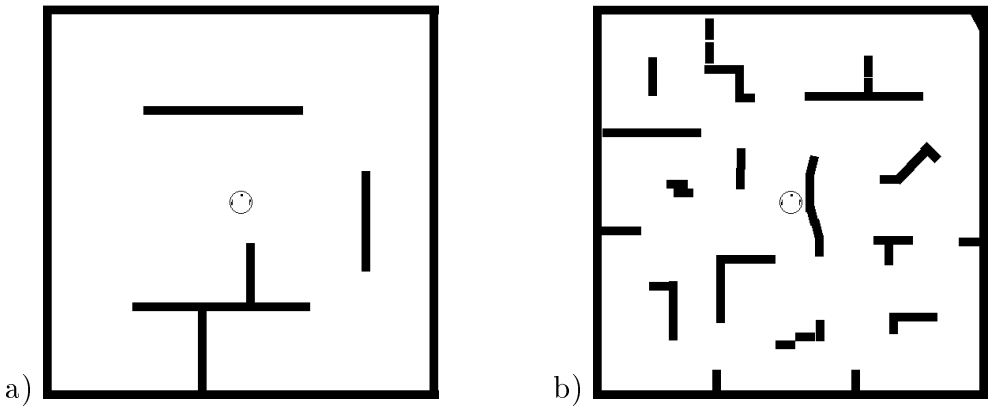

Figure 2: Two example environments used for incremental evolution of robot controllers. a.) A simple one, and b.) a more challenging one.

is to evolve a network which allows the Khepera robot to move in a given environment as long as possible without hitting any obstacle. For solving it, the eight infra-red proximity sensors of the Khepera are used; six in the front and two in the rear of the robot. Thus, initially the individual neuromodules have only eight linear input neurons as buffers and two nonlinear output neurons.

The fitness function F_α which implements the given problem simply states: For a given time T go straight ahead as long and as fast as possible. This is coded in terms of the network output signals out_1 and out_2 as follows. First the quantities m_1 and m_2 are defined by

$$\text{if} \quad out_i \leq 0 \quad \text{then} \quad m_i = 0, \quad \text{else} \quad m_i = out_i, \quad i = 1,\, 2\,.$$

Then the fitness function is defined by

$$F_\alpha := \sum_{t=1}^{T} k_1 \cdot (m_1(t) + m_2(t)) - k_2 \cdot |\, m_1(t) - m_2(t)\,|\,, \tag{2}$$

with appropriate parameters k_1 and k_2.

Starting the evolution with simulated robots in a simple environment like the one shown in figure 2, it takes around 100 generations (depending on the parameter settings of the ENS^3 programme) to get individuals having satisfactory fitness values. Although generating a comparable good robot behavior, the corresponding neural networks can differ in size as well as in their connectivity structures.

The development of brain size (number of neurons and number of connections) of a typical evolutionary process in one and the same environment can

be followed in figure 3. At the beginning robots do not move or are spinning around having only one wheel active, but already after a few generations the first robot is moving slowly on a straight line. After thirty generations the fittest robots are exploring the accessible space, moving faster on straight lines and turning when near a wall. During this phase networks are still growing in general. The irregular development of network size corresponds to the different initial conditions from which robots in every generation have to start; they are more or less difficult to cope with. At around generation 65 the number of hidden neurons stabilizes around networks with no internal units and around 16 synapses.

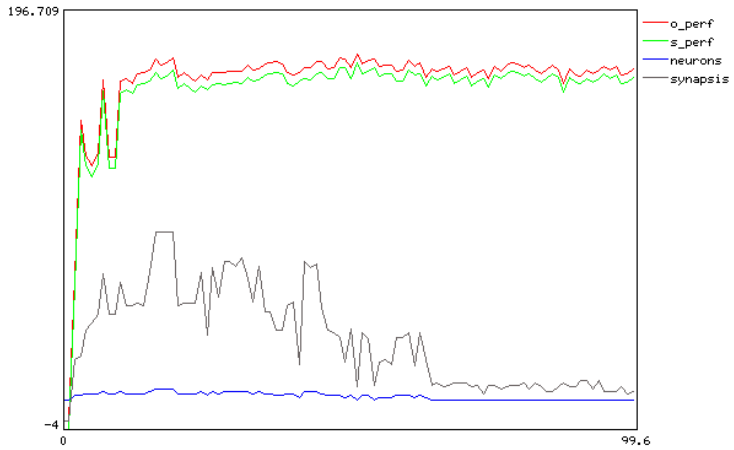

Figure 3: The performance of simulated robots and the development of network size during an evolutionary process.

To make the simulator solutions more robust in the sense, that they can control the real robot in its very different physical environment with equally good efficiency, we change the simulator environments to gradually having more obstacles or walls including for instance walls at 45 angles. Example environments used for the evolution are shown in figure 2.

In figure 4b a typical path of a simulated robot in one of the environments is depicted. The paths for the left and the right wheels start at the point marks and end at the cross marks. It can be seen, that the robot turns left as well as right in different situations. It achieves this by turning the corresponding wheel backwards for a short moment.

The network solutions, which generated the desired behavior for simulated as well as for real robots, turned out to be astonishingly small. Many of them did not even use internal neurons; they use only direct connections from inputs to the output neurons. For example, the fully connected network

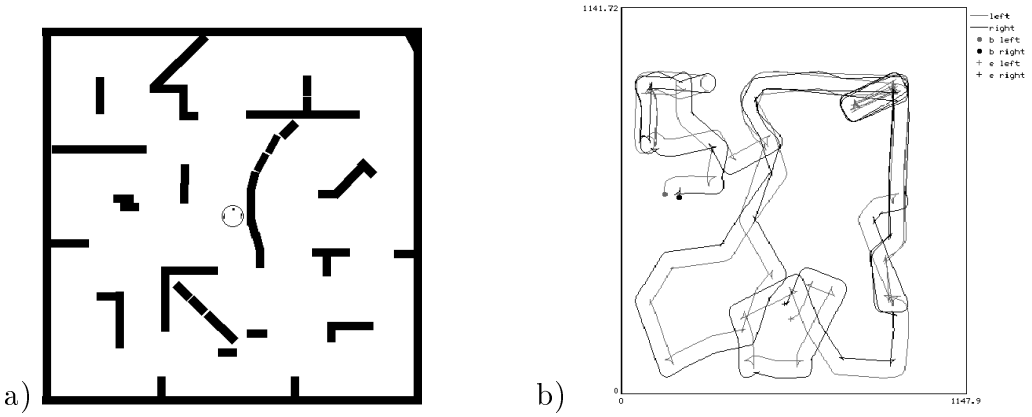

Figure 4: a.) A simulator environment and b.) a robot path (left and right wheels) in this environment for 3000 time steps.

of figure 5 was not only effectively avoiding obstacles in the simulator as well as in real world environments, but it was also able to navigate out of narrow channels, turned in small angled corners, and found its way out of dead ends. The explanation for this efficient behavior can be found simply in the excitatory self-connection w^s of one of the output neurons. The value of this connection is larger than 1 and therefore this neuron is working as an hysteresis element [26], jumping forward and backward between two states at different sensor input configurations. In fact, the turning angle of the robot is proportional to the value of this self-connection. This was verified by changing the positive self-excitation $w^s > 1$ by hand, and loading the manipulated network again back to the real robot. For large values of w^s the robot starts to rotate if it looses "sight" contact to walls. Although there are more recurrences involved – the other output neuron has an excitatory self-connection and the output neurons themselves are recurrently coupled – their weights are not strong enough to generate oscillations or more complex internal dynamics.

Observing the actions of the real robots in there environments, the impression is that they do not only avoid obstacles but that they also show an exploratory behavior. Letting the robot move in a given environment, after a while it will have visited almost all areas within reach; it moves through small openings in the walls, wanders through narrow corridors and even turns in dead ends. This can of course not be achieved by a pure wall following behavior, as it is usually learned by robots.

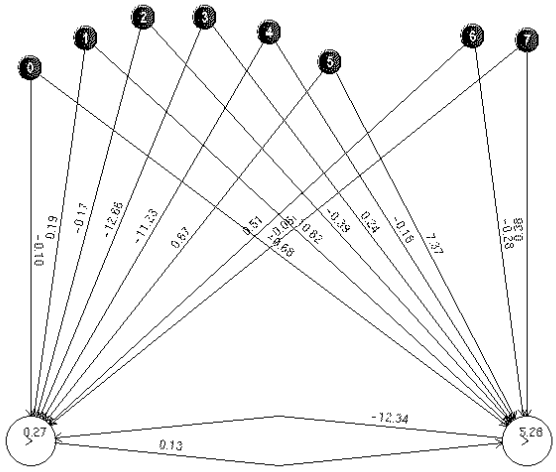

Figure 5: An evolved, almost fully connected network without internal neurons generating an effective obstacle avoidance behavior also for the real robot. Input signals 0 to 5 are coming from the front proximity sensors, inputs 6 and 7 from the rear proximity sensors.

3.2 The second experiment: Phototropism

The second fundamental behavior may be called phototropism, finding food, or the like [17], [35], [32]. In general this will be implemented as a gradient following behavior. Here the goal of the experiment is to approach a light source as fast as possible and stay there "eating", i.e. with face to the light source. For this second experiment in addition to the 8 proximity sensors the 8 light sensors of the Khepera are used; i.e., now there are 16 sensor inputs.

To find an appropriate solution we use the following fitness function stating: For a given time T and a given environment "eat" as much light as possible. This is coded as follows

$$F := \sum_{t=1}^{T} k_3 \cdot in_f(t) + k_4 \cdot |\, in_f(t) - in_r(t) \,|\,, \tag{3}$$

where k_3 and k_4 are appropriate parameters, and in_f and in_r are given in terms of the inputs $i[0]$, ..., $i[7]$ to the additional light sensors by

$$in_f := i[0] + i[2] + i[3] + i[5], \quad in_r := i[6] + i[7]\ .$$

Thus, for determining the fitness we use only four of the six front light sensors and the two light sensors at the rear. But the proximity sensors are still used to generate the terminating "bump"-signal.

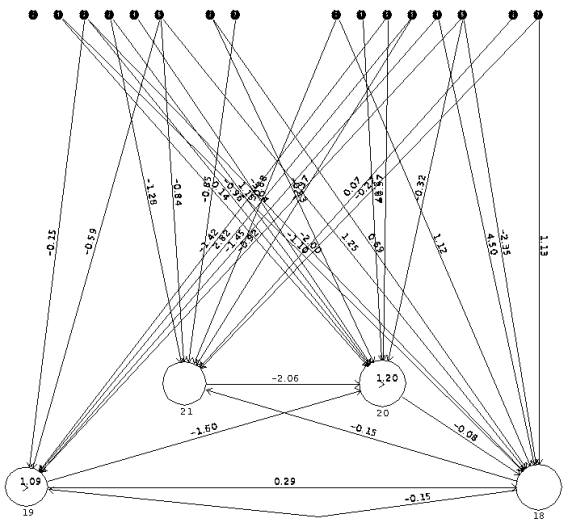

Figure 6: Evolved network with phototropic behavior doing obstacle avoidance as well.

The following experiment does not make use of obstacle avoiding modules, but starts from scratch, now with all 16 sensors as inputs and two output units driving the motors as before. Recall that the two sensor types have different preprocessing: proximity sensors generate signals in the interval $(-1, 1)$, the light sensors give only negative values in $(-1, 0)$. The evolution process is started for simulated robots in an environment with a few light sources spread over an environment without obstacles. We then gradually make the environmental boundary conditions more challenging; i.e., the number of light sources is reduced and some obstacles are added.

Again there were reasonable solutions of different network size and structure. One of the networks with only a few connections but with an remarkably good performance is shown in figure 6. Its architecture has the often observed configuration of output neurons: They are recurrently connected (an odd 2-loop) and one output neuron (19, driving the left motor) has a "super-critical" excitatory self-connection $w^s_{19} = 1.09 > 1$. The two hidden neurons, 20 and 21 together with the output neuron 18 (driving the right motor) act in an odd 3-loop. The hidden neuron 20 has also a "super-critical" excitatory self-connection $w^s_{20} = 1.2 > 1$. Remarkable are also the inhibitory back-projections from the output neurons to the hidden neurons. Both types of signal inputs (proximity and light sensors) have connections to the hidden neurons as well as to the output neurons. But this connectivity structure is again highly un-symmetric. The leftmost proximity sensor is not even con-

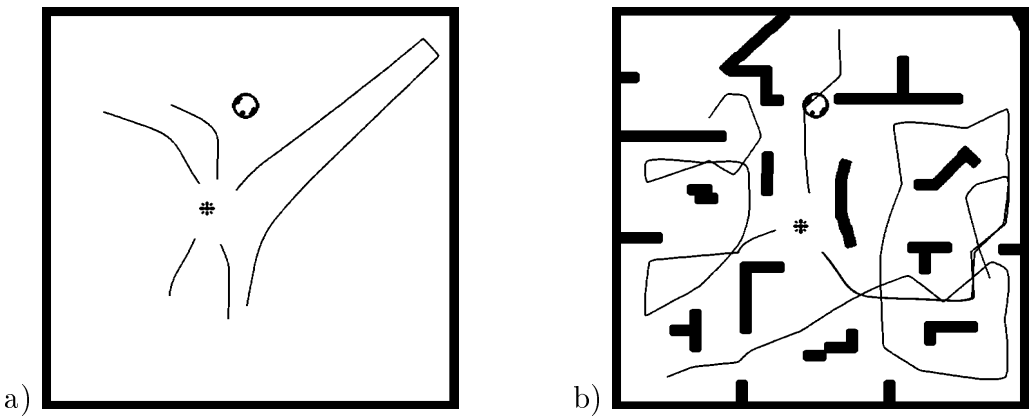

Figure 7: Different paths of a light seeking robot in simulator environments: a.) without obstacles and b.) with many obstacles.

nected. Although this network has not the optimal performance it "survives" in simulated environment with high fitness. But also the real robot in the physical environment finds a light source (a small lamp) and follows it, when it is moved.

The behavior generated by this network can be read from figures 7a and 7b: In a simulator environment free of obstacles the robot turns towards the light source if it comes near to it. But figure 7a also shows that it turns away from the source if the light comes from its left side. It moves straight if no light source is around. In an environment with obstacles, like the one shown in figure 7b, it follows a kind of random walk without hitting an obstacle until it finally finds the light source and stays there.

4 Discussion

Although the presented results are only preliminary, they clearly indicate that the ENS^3-algorithm is able to generate interesting solutions to control problems in the domain of autonomous robots. The problems considered in this paper are of course simple ones, and their different neural network solutions have been discussed in several places, as cited above. But also for these simple problems we learned from the ENS^3-solutions, that generated networks can be smaller than the usual feedforward controllers [30] or more elaborate controllers [14], and that internal dynamical effects may provide surprising additional "tricks" to solve control tasks. This is, for instance, the case in the first example, where a simple hysteresis element enabled the system to leave dead ends or sharp corners.

Another lesson to learn from these first experiments is the fact that there is no simple structure-function relationship for neuro-controllers. This can be deduced from the observation that evolved networks of *different size* and of *different structure* solve problems equally good; i.e., they have the same fitness with respect to a given problem. This can be studied, because ENS^3 neither fixes the number of neurons and of connections, nor the type of connectivity structure in advance. Size and structure of neuro-controllers depend crucially on cost terms for neurons and connections in the fitness function. For the tasks discussed above, for instance, networks with up to 10 internal neurons and more than 200 connections evolved, all having more or less the same good performance as the small networks presented in the sections above. They use very different recurrent structures (e.g. loops of different lengths). To study the functional role of the corresponding non-trivial internal dynamical properties of these networks will be the goal of further investigations.

Dynamical properties of recurrent networks for robot control have been analysed for instance also in [16]. To be able to correlate neuron activities of the evolved controllers with the dynamical sensor input and motor output signals, corresponding data can be visualized during the action of the simulated or real robot using the simulator interface. Using this part of the ENS^3-program one can test different hypothesis' about the functional role and behavioral relevance of recurrent network structures and their dynamical properties.

Furthermore, the ENS^3 program is completed with a tool [37] giving a graphical output of evolved network structures. Using this tool, parameters like synaptic weights can be changed and connections can be deleted or added by hand. The modified networks then can be loaded again back into the simulator for the control of the robot. Thus, kind of lesion experiments can be performed, the role of internal oscillators for behavior generation can be studied, and specific dynamical properties may be implemented and tested. This will extend the experimental feasibility to study the role of internal dynamics in systems developing cognitive abilities.

The above described experiments suggest that using an appropriate simulator for a robot and developing a behavior in terms of incremental evolution, i.e. with gradually increasing complexity of the environmental boundary conditions, ENS^3 will produce networks, which are surprisingly robust in the sense that the real robots manage to solve the tasks under very different physical boundary conditions. Having controllers of different size and with different connectivity structure at hand, we also learned that larger networks in general are not more robust than small ones, like those presented here.

Finally, the approach can be used to study and analyse also theoretical

issues of artificial evolution like correlation of fitness landscapes, the role of neutral networks, and the like. Furthermore, implementing an operator L for individual learning during a generation in the ENS^3-algorithm will allow the discussion of phenomena like the Baldwin effect [34]. Learning in the sense of parameter optimization is no simple task in this context, because, following the ALife approach to autonomous systems, there is no teacher signal, no definite procedure to solve a task, and the general recurrent structure of the networks is a further obstraction to the known learning procedures. Future work will therefore concentrate also on the developement of behavior oriented learning rules for complex adaptive systems.

References

[1] Adami, C. (1998), *Introduction to Artificial Life* Springer Verlag, New York.

[2] Angeline, P.J., Saunders, G.B. and Pollack J.B. (1994), An Evolutionary Algorithm that Evolves Recurrent Neural Networks, *IEEE Transactions on Neural Networks*, **5**, 54–65.

[3] Arbib, M A., Érdi, P., and Szentágothai, J (1998), *Neural Organization - Structure, Function, and Dynamics*, MIT Press, Massachusetts.

[4] Ashby, W. R. (1952), *Design for a Brain*, Wiley, New York.

[5] Beer, R. D., and Gallagher, J. C. (1992) Evolving dynamical neural networks for adaptive behavior, *Adaptive Behavior*, **1**, 91–122.

[6] Baitenberg, V. (1984), *Vehicles: Experiments in Synthetic Psychology*, MIT Press, Cambridge, MA.

[7] Brooks, R. A. (1986), A robust layered control system for as mobile robot, *IEEE Journal of Robotics and Automation*, **RA-2**, 14–23.

[8] Buzsáki, G., Llinás, R., Singer, W., Berthoz, A., and Christen, Y. (Eds.) (1994), *Temporal Coding in the Brain*, Springer Verlag, Berlin.

[9] Calabretta, R., Nolfi, S., Parisi, D., and Wagner, P. (1998), A case study of the evolution of modularity: towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science, in: Adami, C., Belew, R. K., Kitano, H., and Taylor, C. E. (Eds.), *Artificial Life VI, Proceedings of The Sixth International Conference on Artificial Life*, MIT Press, Cambridge.

[10] Cliff, D., Harwey, I., and Husbands, P. (1993), Exploration in evolutionary robotics, *Adaptive Behavior*, **2**, 73–110.

[11] Duke, W., and Pritchard, W.S. (Eds.) (1991), *Proceedings of the Conference on Measuring Chaos in the Human Brain*, World Scientific, Singapore.

[12] Elbert, T., Ray, W.J., Kowalik, Z.J., Skinner, K.E., Graf, K.E., and Birbaumer, N. (1994), Chaos and physiology: deterministic chaos in excitable cell assemblies, *Physiological Review* **74**, 1–47.

[13] Floreano, D. (1997) Ago ergo sum, in: Mulhauser, G (Ed.), *Evolving Consciousness*, J. Benjamins, Amsterdam.

[14] Gaussier, P., and Zrehen, S. (1994) A topological neural map for on-line learning: Emergence of obstacle avoidance in a mobile robot, in: Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. (eds.), *From Animals to Animats III: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Cambridge, MA: MIT Press-Bradford Books, pp. 282–290.

[15] Husbands, P., and Harwey, I. (1992), Evolution versus design: Controlling autonomous robots, in: *Integrating perception, planning and action: Proceedings of the Third Annual Conferences on Artificial Intelligence*, IEEE Press, Los Alamitos.

[16] Husbands, P., Harwey, I., and Cliff, D. (1995), Circle in the Round: State Space Attractors for Evolved Sighted Robots, in: Steels, L. (ed.), *The Biology and Technology of Intelligent Autonomous Agents*, NATO ASI Series, Series F: Computer and Systems Sciences, Vol. 144, Springer, pp. 222–257.

[17] Kodjabachian, J., and Meyer, J. A. (1998) Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects, *IEEE Transactions on Neural Networks*, **9**, 796–812.

[18] Langton, C. G. (1989) Artificial Life, in: Langton, C. G. (Ed.) Artificial Life, Addison-Wesley, Redwood City CA.

[19] Lara, B., Hülse, M., and Pasemann, F. (2001) Evolving different neuromodules and their interfaces to control autonomous robots, MPI-MIS-Preprint 21, accepted paper for "The 5th World Multi-Conference on

Systemics, Cybernetics and Informatics" (SCI2001), July 22-25, 2001, Orlando, Florida USA, Proceedings.

[20] Michel, O., *Khepera Simulator* Package version 2.0: Freeware mobile robot simulator written at the University of Nice Sophia-Antipolis by Oliver Michel. Downloadable from the World Wide Web at `http://wwwi3s.unice.fr/~om/khep-sim.html`

[21] Mondada, F., Franzi, E., and Ienne, P. (1993), Mobile robots miniturization: a tool for investigation in control algorithms, in: *Proceedings of ISER' 93*, Kyoto, October 1993.

[22] Mondada, F. and Floreano, D. (1995), Evolution of neural control structures: Some experiments on mobile robots, *Robotics and Autonomous Systems*, **16**, 183–195.

[23] Nolfi, S., and Floreano, D. (2000), *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, MIT Press, Cambridge.

[24] Nolfi, S. (1998), Evolutionary robotics: Exploiting the full power of self-organization, *Cnnection Science*, **10**, 167–183.

[25] Pantev, C., Elbert, T., and Lutkenhöhner B. (Eds.) (1995), *Oscillatory Event-related Brain Dynamics*, Plenum Press, London.

[26] Pasemann, F. (1993), Dynamics of a single model neuron, *International Journal of Bifurcation and Chaos*, **2**, 271–278.

[27] Pasemann, F. (1998), Evolving neurocontrollers for balancing an inverted pendulum, *Network: Computation in Neural Systems*, **9**, 495–511.

[28] Pasemann, F. (1998), Structure and Dynamics of Recurrent Neuromodules, *Theory in Biosciences*, **117**, 1–17.

[29] Pfeifer, R., and Scheier, C. (2000), *Understanding Intelligence*, MIT Press, Cambridge.

[30] Pfeifer, R., and Verschure, F. M. J. (1993), Designing efficiently navigating non-goal-directed robots, in: Meyer, J.-A., Roitblat, H., and Wilson, S. (eds.), *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Cambridge, MA: MIT Press-Bradford Books, pp. 31–38.

[31] Renals, S., and Rohwer, R. (1990), A study of network dynamics, *Journal of Statistical Physics*, **58**, 825–848.

[32] Scutt, T. (1994), The fife neuron trick: Using classical conditioning to learn how to seek light, in: Cliff, D., Husbands, P., Meyer, J.-A., and Wilson, S. (eds.), *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, Cambridge, MA: MIT Press-Bradford Books, pp 264–370.

[33] Schaffer, J. D., Whitley, D., and Eshelman, L. J. (1992), Combination of genetic algorithms and neural networks: A survey of the state of the art, in: Whitley, D., and Schaffer, J. D. (Eds.) *International Workshop on Combinations of Genetic Algorithms and Neural Networks - Proceedings COGANN-92*, IEEE Computer Society Press, Los Alamitos, CA.

[34] Turney, P., Whitley, D., and Anderson, R. (1996) Evolution, learning, and instinct: 100 years of the Baldwin effect, Editorial for the Special Issue: The Baldwin Effect, *Evolutionary Computation*, **3**,4.

[35] Verschure, F. M. J., and Pfeifer, R. (1993), Categorization, representation, and the dynamics of system-environment interaction: a case study in autonomous systems, in: Meyer, J.-A.,Roitblat, H., and Wilson, S. (eds.), *From Animals to Animats II: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, Cambridge, MA: MIT Press-Bradford Books, pp. 210–217.

[36] Wang, X., and Blum, E. K. (1995), Dynamics and bifurcation of neural networks, in: Arbib, M. A. (Ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press, Cambridge, Massachusetts.

[37] Drawing Graphs with VGJ, `http://lsiibm15.epfl.ch/graph_drawing.html`, 07.06.2001.