Evolving Reinforcement Learning-Like Abilities for Robots

Jesper Blynel

Autonomous Systems Lab
Institute of Systems Engineering
Swiss Federal Institute of Technology (EPFL)
CH-1015, Lausanne, Switzerland
`Jesper.Blynel@epfl.ch`

Abstract. In [8] Yamauchi and Beer explored the abilities of continuous time recurrent neural networks (CTRNNs) to display reinforcement-learning like abilities. The investigated tasks were generation and learning of short bit sequences. This “learning” came about without modifications of synaptic strengths, but simply from internal dynamics of the evolved networks. In this paper this approach will be extended to two embodied agent tasks, where simulated robots have acquire and retain “knowledge” while moving around different mazes. The evolved controllers are analyzed and the results are discussed.

1 Introduction

Yamauchi and Beer [8] showed in 1994 that continuous time recurrent neural networks (CTRNNs) can be evolved to display reinforcement learning-like abilities. The investigated tasks were generation and learning of short bit sequences. These tasks were accomplished without modifications of synaptic strengths, but simply by exploiting the rich internal dynamics of this class of networks. The claim was that the kind of “learning” showed by the networks is more biological plausible than more traditional approaches to learning such as reinforcement learning [4]. In order to justify their choice of bit sequence tasks, the authors made the analogy that a rat, when navigating a maze, is faced with a sequence binary decisions of which way to turn. Despite of this analogy the tasks studied were very artificial seen from an evolutionary robotics perspective.

This issue will be picked up upon in this paper by applying a similar approach in a more “realistic” setting of a simulated robot navigating a maze. In the first of several trials in an environment the robot has to a acquire and retain “knowledge” which can then later on be exploited. In order to show the feasibility of the approach the first task investigated is a simple setup where the movement of the robot is constrained to one line. In the second task the robot has to navigate in the more challenging environment of a T-maze. The learning-like properties of a successful controller is analyzed and it is shown how to modulate the robot behavior by changing the activity of one neuron of an evolved network.

Tuci et. al. [6] have recently, in parallel to the work presented in this paper, applied a similar approach in order to solve an extended version of a landmark task also originally studied by Yamauchi and Beer [7]. A simulated robot had to sometimes approach a light source and sometime avoid it, based on the position a reward-zone in the environment. A very complex and task specific fitness function, however, had to be used in order to evolve successful controllers. The reason for this might be that task studied required evolved controllers to use the same sensory modality for the two very different behaviors of light approaching and light avoidance. One could suspect that this fact made the search problem for the genetic algorithm much harder. In the work presented in this paper the focus is on applying much simple fitness functions, but shaping the experimental settings in oder to make to search task for the genetic algorithm accomplishable.

We have previously compared the class of CTRNNs used in this article to Plastic Neural Networks (PNNs) with online modification of network weights, on a set of tasks requiring sensory-motor adaptivity and learning [1]. The results obtained were that the PNNs were more adaptive with respect to changes applied to the environment after the evolutionary process, but the CTRNNs, on the other hand were easier to evolve to display learning-like abilities. For this reason focus of this paper will be on CTRNNs.

The rest of this paper is organized as follows. Section 2 describes the neural network used, section 3 describes the first simple experiment, section 4 describes and analyses the more complex T-Maze experiment. Section 5 discusses the results obtained and concludes the paper.

2 Continuous-Time Recurrent Neural Networks

In the continuous-time recurrent neural networks used for the experiments in this paper the state of each neuron can be described by the following differential equation:

$$\frac{d\gamma_i}{dt} = \frac{1}{\tau_i}\bigg(-\gamma_i + \sum_{j=1}^{N} w_{ij}A_j + \sum_{k=1}^{S} w_{ik}I_k \bigg) \tag{1}$$

, where N is the number of neurons, i $(= 1, 2, ..., N)$ is the index, γ_i describes the neuron state (cell potential), τ_i is the time constant, w_{ij} is the strength of the synapse from the presynaptic neuron j to the postsynaptic neuron i, $A_j = \sigma(\gamma_j - \theta_j)$ is the activation of the presynaptic neuron where $\sigma(x) = 1/(1 + e^{-x})$ is the standard logistic function and θ_j is a bias term. Finally, S is the number of sensory receptors, w_{ik} is the strength of the synapse from the presynaptic sensory receptor k to the postsynaptic neuron i and I_k is the activation of the sensory receptor ($I_k \in [0, 1]$). As in [8] the *Forward Euler* numerical integration method was used. The iterative update rule for the state of each neuron becomes:

$$\gamma_i(n+1) = \gamma_i(n) + \frac{\Delta t}{\tau_i}\bigg(-\gamma_i(n) + \sum_{j=1}^{N} w_{ij}A_j(n) + \sum_{k=1}^{S} w_{ik}I_k \bigg) \tag{2}$$

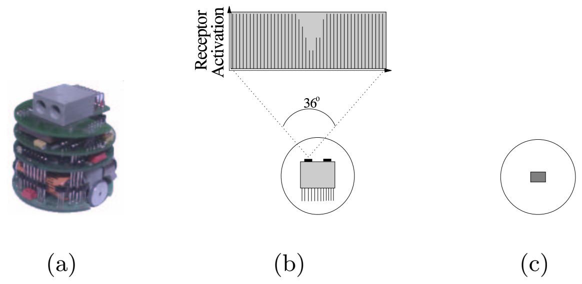


Fig. 1. The Khepera robot simulated in the experiments. The robot has (a) 8 infrared sensors distributed around the body used for object proximity and ambient light intensity measurements; (b) a linear vision module with 64 equally-spaced photoreceptors covering a visual field of 36°; and (c) a floor sensor measuring surface brightness.

, where n is the iteration step number and Δt is the step size. Initially the state of each neuron was $\gamma_i(0) = 0 \;\; \forall i$, the step size set to $\Delta t = 1$. The range of the other parameters were the following:

$$\tau \in [1, 70], \;\; \theta \in [-1, 1] \;\; and \;\; w \in [-5, 5]$$

Notice from equation (2) how the time constant τ determines the dynamics of the neuron. A high time constant results in a slowly changing neuron state while decreasing it makes the neuron more responsive to current synaptic inputs approximating a reactive neuron with no internal state in the limit.

3 The Simple "Reinforcement Learning" Task

The first experiment is a basic setup which requires acquisition and storage of "knowledge". A simulated Khepera robot (figure 1) is placed in a rectangular arena with two potential reward-zone positions (figure 2), a light bulb to the left and a black vertical stripe on the right. The robot has 6 trials to find out where the reward-zone is, go there and stay there. At the beginning, the position of the reward-zone (grey half-circle in figure 2) is randomly chosen, either to the left (below the light bulb) or to the right (below the stripe), and remains the same for 3 consecutive trials. After 3 trials, the reward-zone is switched to the other end of the environment. At the beginning of each trial, the robot is randomly positioned within the center third of the dashed line shown in figure 2, always facing the black stripe. In order to make the task simpler, the robot can only move back and forth along this line, i.e. it cannot rotate. This is realized by having one motor output neuron controlling the speed of *both* wheels of the robot.

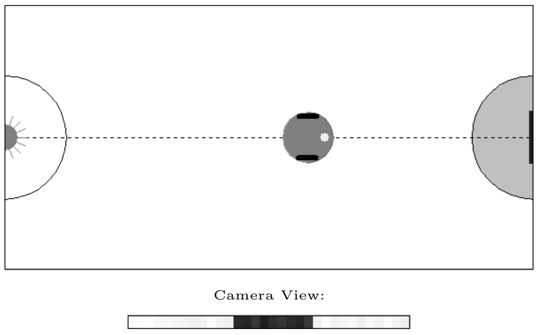


Fig. 2. The environment used in the simple reinforcement learning task. To the left there is a light bulb and to the right there is black stripe on the wall. A gray reward-zone is randomly placed in either end (here shown to the right). The robot is constrained to move along the dashed line always facing the black stripe on the wall. Below the environment the view from the linear camera is shown.

The reinforcement signal comes from a floor sensor [1] (figure 1(c)) which is *on* when the robot is inside a gray reward-zone and *off* otherwise. Notice that this information is a sensory input just like any other, in contrast to conventional reinforcement learning systems where the reward signal plays a special role in the architecture and in the learning algorithm [4].

3.1 Network Architectures and Genetic Encoding

The neural network architecture is shown in figure 3. The network consists of 5 fully interconnected neurons (4 hidden + 1 motor output) and 6 sensory receptors. Every neuron has synaptic connections from all neurons and all sensory receptors. The sensory receptors are configured the following way (see also figure 4):

– *2 Light receptors*: The robots infrared sensors in passive mode are used to measure the ambient light. Only the two sensors on the back of the robot are used.
– *3 Visual receptors*: The linear vision module has 64 equally spaced photoreceptors spanning a visual field of 36° (figure 1(b)). The visual field is divided into 3 sectors and the average pixel-value in each sector is passed to the corresponding visual receptor.
– *1 Floor receptor*: An infrared sensor in the center of the robot body pointing downwards (figure 1(c)) measures surface brightness. If the robot is inside the reward-zone the corresponding receptor is *on* (=1) and otherwise it's *off* (=0).

[1] This sensor can be easily be mounted on a standard Khepera robot by placing an additional IR sensor under the robot body and connecting it one of the A/D channels.

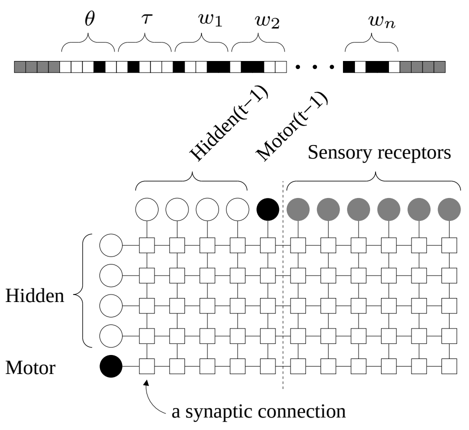


Fig. 3. Genetic encoding of the parameters for one neuron (top) and architecture of the neural network used in the simple reinforcement learning task. *Genetic Encoding*: Each neuron parameter is encoded using 5 bits. θ is the bias, τ is the time constant, and $w_1 \ldots w_n$ are the strengths of the incoming synapses to this neuron. *Neural Architecture*: The network consists of 5 neurons (4 hidden + 1 motor output) and 6 sensory receptors. Every neuron has synaptic connections from all neurons and all sensory receptors.

The value from each receptor is scaled into the range $[0, 1]$. The activation of the motor output neuron in scaled linearly in the range $[-10, 10]$ and used to set the speed of *both* wheels of the simulated robot. The parameters of the network are encoded in a bitstring genotype (figure 3, top). Each neuron has 13 encoded parameters: a time constant (τ), a threshold (θ), and 11 synaptic strengths (w_{ij}). Each parameter is encoded linearly within it's range using 5 bits, giving a total genotype length of 325 bits.

3.2 Experiments

The experiments were carried out in a realistic simulation of the Khepera robot based on samplings of infrared sensor values [3], computing geometric projections for linear camera inputs and adding 5% uniform noise to every value. A simple genetic algorithm with rank-based selection was used. A population of 100

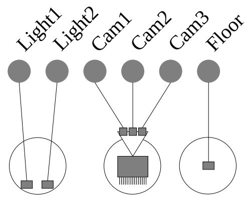


Fig. 4. Configuration of the 6 sensory receptors used in the simple reinforcement learning task.

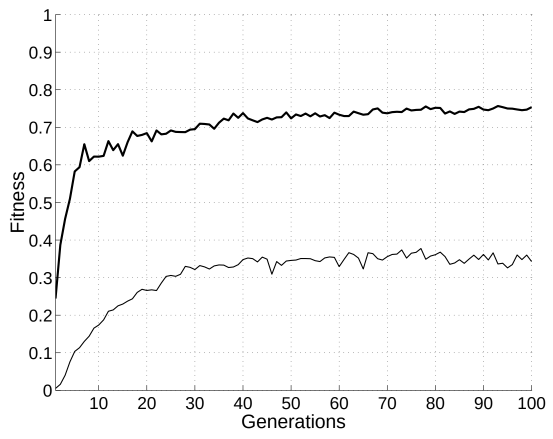


Fig. 5. *Simple Reinforcement Learning Task.* Thick line shows best fitness and thin line shows population mean (both are averages over 10 replications of the experiment).

neural controllers was evolved for 100 generations. At every generation the best 20 individuals made 5 copies each. One copy of the best individual remained unchanged (elitism). Single-point crossover with a 0.04 probability and bit-switch mutation with a 0.02 probability per bit was used. Every neural controller was tested for 3 epochs of 6 trials each. Each trial lasted 150 sensory-motor steps (corresponding to 15 seconds of simulated time). At the beginning of each *epoch*, the neural controller was re-initialized by setting the state of each neuron to 0. This means that the robot could potentially build up and store information in the dynamic neuron states between trials within the same epoch. The fitness of an individual was proportional to the amount of time spend on the reward-zones subtracted a penalty for spending time in only one of the two zones:

$$fitness = \frac{\sum_{i=1}^{epochs} f_1(i) + f_2(i) - |f_1(i) - f_2(i)|}{epochs \times trials\_per\_epoch \times steps\_per\_trial}$$

, where f_1 is the number of steps spend in the reward-zone in trials with reward to the left and f_2 is the number of steps spent in the reward-zone in trials with reward to the right. Notice that if f_1 or f_2 is zero in an epoch the total fitness will also be zero in that epoch. Without taking the absolute difference between f_1 and f_2, evolved controllers found the sub-optimal solution of only accumulating fitness in one end of the environment.

The experiment was repeated 10 times with different initializations of the computer's pseudo-random number generator. The results of the experiments are shown in figure 5. The thick line shows best fitness and the thin line shows population mean, both are averages over 10 replications of the experiment. As can be seen from the graph only about 40 generations on average were required for the fitness values to reach a stable high level.

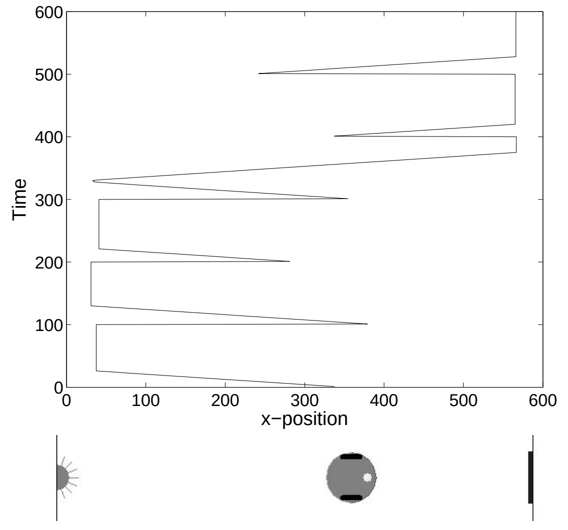


Fig. 6. Typical behavior of an evolved robot in the simple reinforcement learning task. *Top*: Robot x-position is plotted against time over 6 trials. *Bottom*: Environment layout. See text for description.

The typical behavior of an evolved controller is visualized in figure 6. The x-position of the robot is plotted over time for an entire epoch of 6 trials [2]. Before the first step of each trial the robot is randomly placed within the center third of the x-axis. This evolved robot begins to move to the left where the reward-zone is positioned for the first 3 trials. After 3 trials the reward position is switched. The robot still moves to the left by default, but when it discovers that the floor sensor remains off, it reverses direction and moves right. For the remaining trials, it "remembers" that the reward-zone is to the right and moves there directly.

4 The T-Maze Task

In the second experiment the robot had to navigate the T-maze shown in figure 7. The task for the robot was in principle the same as in the previous experiment, but because of the increased complexity of the environment the robot was now allowed to move both wheels independently. The reward-zone (black square in figure 7) could be positioned in either the left or the right arm of the maze. The position of the reward-zone stayed fixed during each epoch. The robot was tested for 4 epochs of 5 trials each - two epochs with the reward-zone in each arm of the maze As before the neural network is re-initialized between epochs but *not* between trials. The optimal behavior of the robot in this environment

[2] Each trial is shortened from 150 to 100 steps in figure 6 because the position of the robot always stabilizes within this time-frame for this individual.

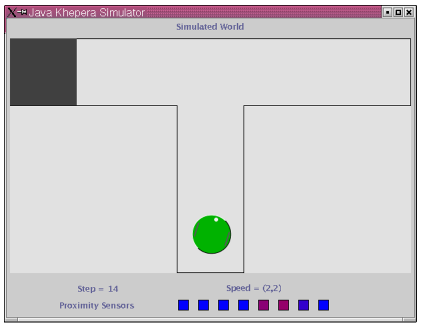


Fig. 7. The T-maze environment used in the second experiment. The reward-zone (black square) can be positioned in either the left arm (shown above) or in the right arm of the maze.

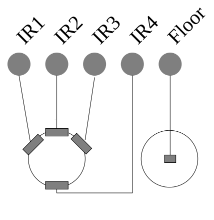


Fig. 8. Configuration of the 5 sensory neurons used in the T-maze task. The eight IR proximity sensors are paired two by and the fifth input is from the floor sensor.

is to use the first trial of each epoch to locate and "remember" the position of the reward-zone, and thereafter move directly towards it for the remaining trials of the epoch. To put additional evolutionary pressure on this behavior, the number of available sensory-motor steps was 360 in the first trial of each epoch and only 180 in the remaining 4 trials. Given the size of the maze this means that the robot only had time the explore the whole maze during the first trial of each epoch. In addition a poison-zone (white square in figure 10(b)) was positioned opposite the reward-zone in the last 4 but *not* the first trial of each epoch. An individual was immediately killed if it stepped over the poison-zone. With these changes the *fitness function* could be simplified to the sum of trials an individual ends it's life inside the reward-zone. Since each individual was tested for a total of 20 trials the maximal possible fitness was 20. Notice that there is no direct pressure on evolving fast moving robots given this fitness function. This is however compensated by the fact the the number of steps in each trial is limited and has been adjusted to fit the size of the environment.

In this experiment, in addition to the floor sensor, only one sensory modality, the infra proximity sensors of the simulated robot, was used (see figure 8). The network architecture was similar to the one shown in figure 3. The network still contained 4 hidden neurons, but now had 2 motor output neurons and only 5 sensory receptors, giving a total of 66 connection weights and a total genotype size of 390 bits. Because of the increased complexity of the task the population size was increased to 200, where 40 parents make 5 copies each. The elite size was set to 5 and the range of the time-constant τ was changed to $[0, 50]$. The evolution was run for 200 generations and replicated 10 times.

The fitness results of the evolutionary runs on this T-maze experiment are shown in figure 9. The evolutionary process found individuals able to collect the maximal fitness of 20 in 6 out of the 10 replications of the experiment. The maximal fitness in the 4 remaining runs was around 16. The behavior of

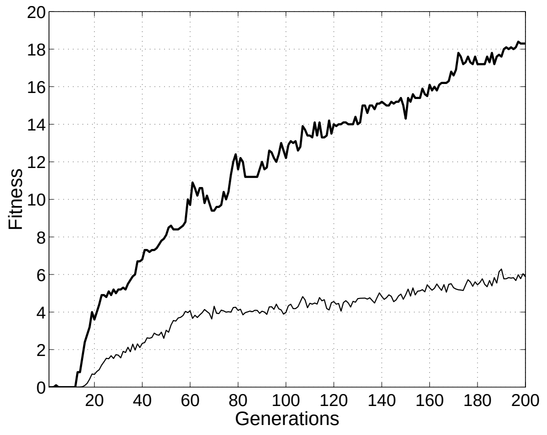
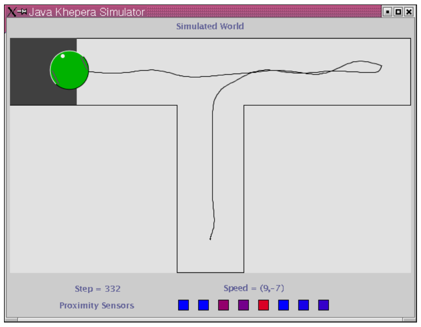


Fig. 9. *T-Maze Task.* Thick line shows best fitness and thin line shows population mean (both are averages over 10 replications of the experiment).

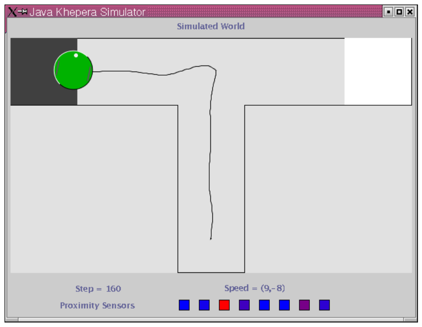
an individual from the final generation of one of the successful runs is shown in figure 10 and 11. The robot starts out in trial 1 of the first epoch (figure 10(a)) by exploring the maze until it locates the reward-zone where it stays the remaining time of the trial. In the following trials (figure 10(b)), the robot is able to retain the "knowledge" gathered during trial 1 and always turns left at the T-junction in order to move towards and stay on the reward-zone. In the epochs with reward to the right the robot moves directly towards the reward-zone in trial 1 (figure 11(a)), since the default behavior of this individual is to turn right at the first T-junction after a re-initialization of the neural controller. For the remaining trials this successful behavior is repeated (figure 11(b)).

4.1 Analysis

In order to better understand how the evolved neural networks worked, some further analysis on an individual from one of the successful evolutions on the T-maze task was done. The neuron activities of the network was recorded over two epochs, one with the reward in each end. The plots of the first trial of each of these two epochs are shown in figure 12(a) and 12(b), both these trial lasting 360 sensory-motor cycles as during evolution. The top four graphs shows the activity of the hidden neurons (H0 to H3) and the next two shows the activity of the motor output neurons (M0 and M1). The output of the floor sensor (F) is added at the bottom for clarification. The activities vary between 0 and 1 as defined by the logistic activation function. The neuron activities of figure 12(a) correspond to the robot trace shown in figure 10(a) and figure 12(b) corresponds to trace shown in figure 11(a). In the first case (reward left) the robot searches the whole environment before if finds the reward-zone, whereas in the second case it finds it directly. This can be seen by comparing the plots of the floor sensor (F) which



(a) *Trial 1*: The robot explores the environment, and after some time locates and stays on the reward-zone.

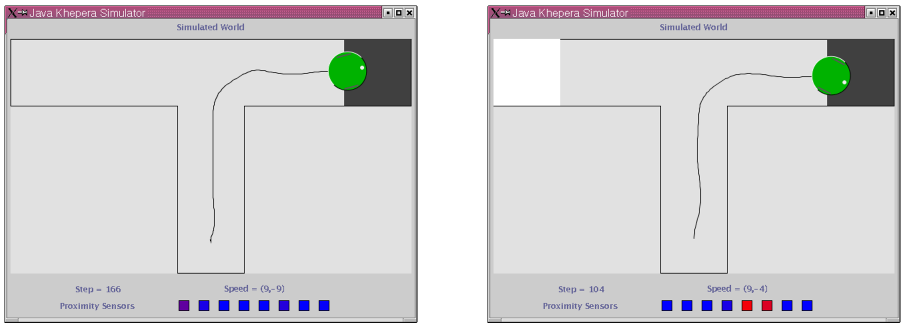


(b) *Trials 2-5*: For the remaining trials the robot exploits the "knowledge" gained in trial 1 and moves directly towards the reward-zone.

Fig. 10. Robot traces of an epoch with the reward-zone to the left.

goes high when the robot enters the reward-zone. In order to better understand how the network retains "knowledge" between trials it is useful to compare the activity of the hidden neurons towards the end of each trial.

This can tell something about how the information which the network seems to retain is captured. It can be seen that the dynamics of neurons H0, H1 and H3 are very similar in the two plots whereas the activity of H2 differs significantly. In figure 12(a) the activity of H2 falls almost linearly from about 0.5 to about 0 across the trial. In figure 12(a), on the other hand, the graph of H2 is similar for the first 150 sensory-motor steps, where paths of the two robots are the same, but then starts to rise when the robot enters the reward-zone, and is about 1 at end of the trial. The state of H2 at the end of the trials was -1.27 in the first case and 2.11 in the second. It seems that this neuron play a main role in the memory capabilities of this controller. In order to further test this hypothesis, an additional set of tests was done. 51 epochs of the robot moving in the maze for 1 trial were performed. Every neuron state was set to 0 (as during evolution) in the beginning of each epoch except for neuron H2 which initial state value was increased by 0.1 for each epoch, starting at -2.5 and ending at +2.5. The behavior of the robot when encountering the T-junction was monitored during these epochs. The robot was able to successfully navigate the maze in every case, although a bit slower than before. It generally took a couple of seconds of adaptation by jiggering movements before the robot was at normal speed. The results showed that the robot turned left at then T-junction whenever the initial state value of H2 was set below -0.3, and it turned right every time when the value was above -0.1. This result shows as hypothesized above, that the overall behavior of the robot can be modulated by changing the state of single neuron which plays a crucial role in this network's "memory capabilities".



(a) *Trial 1*: The default behavior of this individual of turning right takes the robot directly to the reward-zone in trial 1.

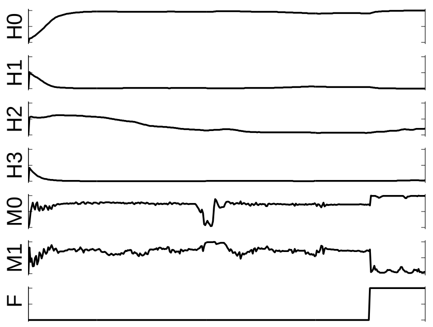
(b) *Trials 2-5*: For the remaining trials the robot repeats this behavior.

Fig. 11. Robot traces from an epoch with the reward-zone to the right.

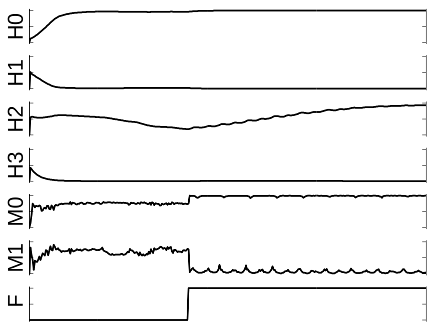
5 Conclusion

The experiments in this paper have shown that the approach of evolving continuous time recurrent neural networks to display reinforcement learning-like abilities initiated by Yamauchi and Beer [8] can indeed be extended to robotics task where an embodied agent has to move around in an environment. The first experiment was on purpose kept simple in order to show the feasibility of the approach. The second experiment showed that similar results could be obtained in a more complex environment. Successful individuals were in both cases able acquire and retain "knowledge" by in trial 1 exploring the environment it happened to be situated in. This information was thereafter exploited in the following trials where direct traces towards the reward-zone was seen. This kind of "learning" resembles biological learning in animals much more than more traditional approaches to learning such as reinforcement learning [4], where an agent normally need a substantial number of learning trials.

The analysis of an individual in T-maze experiment showed that one of the hidden neurons in the network was responsible for storing crucial "knowledge" and thus generating the learning-like behavior. This shows that, at least in this kind of environments investigated in this paper, it is not necessary for an successful agent to build a cognitive map of the environment. If this is also the case in more complex environments needs to be investigated. Current research focuses on extensions to double T-maze environments similar to those recently investigated in [5] on a series of delayed response tasks. Future work also includes transfers of the evolved behaviors to a physical Khepera robot.



(a) Activity of each neuron plotted over trail 1 of an epoch with reward to the left.



(b) Activity of each neuron plotted over trail 1 of an epoch with reward to the right.

Fig. 12. Neuron activities.

Acknowledgements

This work was supported by the Swiss National Science Foundation, grant nr. 620-58049. Parts of this paper was written during a short stay at the Adaptronics Group, The Maersk Institute, University of Southern Denmark.

References

1. J. Blynel and D. Floreano. Levels of dynamics and adaptive behaviour in evolutionary neural controllers. In Hallam et al. [2], pages 272–281.
2. B. Hallam, D. Floreano, J. Hallam, G Hayes, and J-A Meyer, editors. *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*. MIT Press-Bradford Books, Cambridge, MA, 2002.
3. O. Miglino, H. H. Lund, and S. Nolfi. Evolving Mobile Robots in Simulated and Real Environments. *Artificial Life*, 2(4):417–434, 1995.
4. R. Sutton and A. Barto. *Introduction to Reinforcement Learning.* MIT Press, Cambridge, MA, 1998.
5. M. Thieme and T. Ziemke. The road sign problem revisited: Handling delayed response tasks with neural robot controllers. In Hallam et al. [2], pages 228–229.
6. E. Tuci, I. Harvey, and M. Quinn. Evolving integrated controllers for autonomous learning robots using dynamic neural networks. In Hallam et al. [2], pages 282–291.
7. B. Yamauchi and R. D. Beer. Integrating reactive, sequential, and learning behaviour using dynamical neural networks. In D. Cliff, P. Husbands, J. Meyer, and S. W. Wilson, editors, *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 382–391. MIT Press-Bradford Books, Cambridge, MA, 1994.
8. B. Yamauchi and R. D. Beer. Sequential behavior and learning in evolved dynamical neural networks. *Adaptive Behavior*, 2(3):219–246, 1994.