Chapter 13

REMEMBERING HOW TO BEHAVE: RECURRENT NEURAL NETWORKS FOR ADAPTIVE ROBOT BEHAVIOR

T. Ziemke

Department of Computer Science
University of Skövde, 54128 Skövde, Sweden

I. INTRODUCTION

The use of artificial neural networks (ANNs) in robots and autonomous agents has during the 1990s become the subject of research in various disciplines, and a number of collections on different aspects of this topic have appeared [Bekey, 1993, Brooks, 1998, Omidvar, 1997, Sharkey, 1997, Ziemke, 1998, Ziemke, 1999]. Roboticists and ANN researchers naturally have an interest in neural robot control and learning, but furthermore this type of system has attracted much attention in embodied AI and cognitive science, as well as artificial life and adaptive behavior research. In particular recurrent neural nets (RNNs) have become the focus of much research, due to their capacity for dealing with temporal and sequential information - a capacity essential to any agent continually interacting with its environment.

Different theoretical perspectives on RNN-controlled agents and examples of recent experimental research are discussed in the following section. Section III will then investigate and compare the suitability of both first- and higher-order recurrent control architectures for the realization of adaptive robot behavior. Two experiments, in which the weights in recurrent control networks are evolved to solve tasks requiring the robot to exhibit state- or context-dependent behavior, are used to demonstrate and analyze different recurrent robot control architectures. The final section will summarize the discussion, present some conclusions, and point out directions for future research on RNNs for adaptive robot behavior.

II. BACKGROUND

As mentioned above, RNN-controlled robots have become the subject of interest in a number of fields of research. From an *engineering perspective,* ANNs in general have a number of properties that make them well suited to deal with the problems of robot control. They are typically considered to be robust to noise and capable to handle incomplete data, which allows them, to a higher degree than conventional algorithms, to deal with, for example, the uncertainties of a dynamic environment and the limitations of sensor measurements. Moreover, the capacity for continual adaptation allows ANN robot controllers to

adapt to, for example, changing environments or failing hardware. Furthermore, ANNs are *model-free* techniques, i.e., they allow robots to learn from the interaction with their environment, and thus they do not require a designer to provide them with an explicit model of the world *a priori*. As Meeden has pointed out, the capacity for adaptation allows for "bottom-up construction" of robot control systems, i.e., it "allows the task demands rather than the designer's biases to be the primary force in the shaping of the system's development" [Meeden, 1996]. Thus, the flexibility of ANN controllers in many cases reduces the task of the designer to the choice of an appropriate control architecture.

These advantages of ANNs are exemplified well by the ALVINN project [Pomerleau, 1993], in which a vehicle is guided by an ANN. The controller, a feedforward ANN trained with the backpropagation algorithm [Rumelhart, 1986], receives as input the 30x32 pixels of the image obtained from a camera mounted on top of the vehicle and directed at the road ahead (see Figure 1). The network's output determines the steering direction. The network initially 'observes' a human driver and is trained on the data (visual input and steering direction) collected during this phase. After training, the network takes over the role of the human driver and steers the vehicle so as to keep it on the road. ALVINN has been shown to work in the real world, including two- and four-lane roads, crossings, dirt roads, and highway traffic at speeds of up to 55 miles per hour.

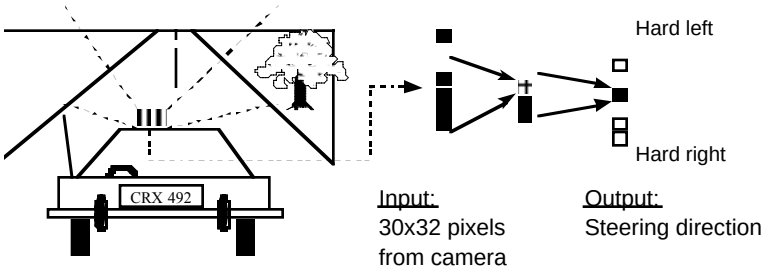

Figure 1. A schematic illustration[1] of the ALVINN vehicle and its control network.

The ALVINN example illustrates the power of standard feedforward networks as well as their limitations. The control network solves a difficult pattern recognition task which certainly would have required complex image preprocessing, the use of line extraction algorithms, etc., if programmed by a human designer. However, due to its use of a feedforward network ALVINN remains a purely reactive system. This means that it has no notion of the temporal aspects of its task and will always react to its visual input in the same fashion, independent of the current context. It neither knows where it came from, nor can it predict what will happen in the future or navigate toward a goal. The reason the systems nevertheless solves a clearly non-trivial task is that the world

[1] In reality ALVINN's control network contains about 1000 units.

it is situated in provides it with the necessary continuity: The road is always 'out there' and functions as a scaffold 'guiding' ALVINN such that the 'navigation' task can be solved in a purely reactive fashion.

For most mobile robots, however, the environment is not that benevolent in the sense that the 'solution' is already built into it. Often they have to solve tasks like homing or finding a goal location, but are faced with the problem of *perceptual aliasing*. That means, many locations in the environment look the same from the robot's current point of view, such that they cannot be distinguished without knowledge/memory of where the robot came from. Moreover, as Meeden [Meeden, 1996] has pointed out, robot/agent problems are often defined in terms of abstract goals rather than specific input-output pairs. Thus, moment-to-moment guidance, as provided in ALVINN's case, is typically not available for a learning robot since for a given situation there is not necessarily only one right or wrong action, and even if there were, it would typically not be known *a priori* [Meeden, 1996].

Furthermore, robots often have to exhibit adaptive behavior to deal with requirements changing over time. Meeden [Meeden, 1993, Meeden, 1996], for example, discusses the case of Carbot, a toy-car-like robotic vehicle placed in a rectangular environment approximately 20 times its own size. Apart from the 'low-level' goal of avoiding bumping into the walls surrounding the environment, Carbot's 'high-level' goal periodically changes between having to approach a light source placed in one corner of the environment and having to avoid it. This means that it should, depending on the current goal, maximize or minimize the readings of two light sensors directed towards the front of the vehicle. Apart from that, Carbot is equipped only with digital touch sensors at the front and the back of the vehicle which detect collisions when they occur, but do not give any advance warning. The task is further complicated by the fact that the smallest dimension of the environment is smaller than Carbot's turning radius. The vehicle can therefore only execute a 180-degree turn in a series of backward and forward movements.

Meeden *et al.* [Meeden, 1993] have carried out extensive experimental comparisons of different feedforward and recurrent control architectures for varying Carbot tasks. The basic recurrent control architecture (Figure 2) was similar to Elman's Simple Recurrent Network [Elman, 1990]. The network's inputs came from light and touch sensors (plus in some experiments an extra input indicating the current goal), its outputs controlled the motor settings, and the hidden unit activation values were copied back and used as extra inputs in the next time step. Not surprisingly, the experimental results show that RNNs consistently outperformed feedforward networks.

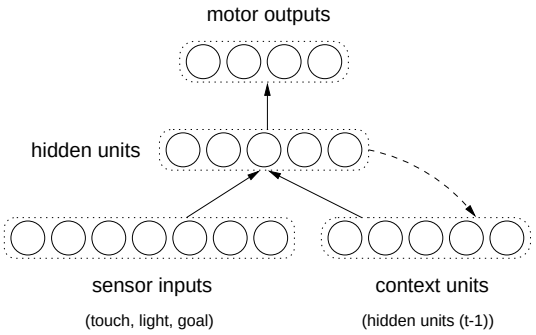

Figure 2. Meeden's recurrent robot control architecture. Solid arrows represent fully connected layer of weights between two layers of units (indicated by surrounding dotted lines). Hidden unit values are fed back via a 1:1 copy connection (dashed arrow) and used as extra inputs in the next time step. Adapted from Meeden [1996].

Meeden *et al.* have analyzed the recurrent control networks and shown that they utilize their internal state (i.e., the hidden unit activation values) to carry out behavioral sequences corresponding to particular motion strategies instead of merely reacting to the current input. For example, to avoid the light the robot would (starting from a position facing the light) first move backwards for a couple of time steps (into the center of the environment) and then carry out a series of alternating forward right and backward left movements until it faces away from the light. Thus, it executes a multi-turn strategy in the center of the environment, which overcomes the problem that the environment's smallest dimension is smaller than its own turning radius. Meeden *et al.* therefore argue that Carbot's behavior is *plan-like* in the sense that (a) it associates abstract goals with *sequences* of primitive actions, (b) the behavior can be described in hierarchical terms (cf. above), and (c) the robot maintains its overall strategy even when flexibly reacting to the environmental conditions. For example, when encountering a wall while carrying out the above light avoidance strategy, it will react to the wall first and then return to its high-level strategy. On the other hand, the behavior is not plan-like in the traditional sense that the robot explicitly anticipates the future, and the number and complexity of Carbot's strategies are admittedly limited. Meeden has further discussed the relation to planning in subsequent work [Meeden, 1994, Meeden, 1996].

For *artificial intelligence* (AI) research the use of ANN-controlled robots, due to their capacity for bottom-up construction of control systems, has received much attention as a methodology for the study of intelligent behavior in artifacts. Traditional AI research, beginning in the mid-1950s, largely ignored earlier work on robotic/cybernetic creatures, such as the work of Grey Walter [Grey Walter, 1950, Grey Walter, 1953]. Instead, AI turned to the computer as a model of mind. In the functionalist framework of *cognitivism* and the *computer metaphor for mind,* having a body, living or artificial, is regarded as a low-level implementational issue. Thus the study of intelligence was largely separated from the interaction between agent and environment, and until the mid-1980s there was relatively little interest in robots in AI research. Instead research focused on internal *representations* corresponding to external objects ('knowledge'), in particular symbolic representations, and the computational,

i.e., formally defined and implementation-independent, processes operating on these representations ('thought'). Problems with this disembodied view of intelligence, which separated internal representations in AI programs from the world they were supposed to represent, were not widely recognized until the 1980s. Searle [Searle, 1980] and Harnad [Harnad, 1990] pointed out that, because there are no causal connections between the internal symbols and the external world, purely computational AI systems lack *intentionality*, i.e., the capacity to relate their internal processes and representations to the external world. During the 1980s many AI researchers therefore (re-) turned to the study of the interaction between agents and their environments. Researchers like Brooks [Brooks 1986, Brooks, 1991] and Wilson [Wilson, 1985, Wilson, 1991] suggested a bottom-up approach to AI, also referred to as *New AI* or *behavior-based AI*, as an alternative to the representationalist/computationalist framework of cognitivism. In particular, it was argued that AI should be approached first and foremost through the study of the interaction between *autonomous agents* and their environments by means of perception and action. For a more detailed review of the bottom-up approach to AI see Ziemke [1998].

Brooks therefore approached the study of intelligence through the construction of physical robots, which were embedded in and interacting with their environment by means of a number of behavioral modules working in parallel in a so-called s*ubsumption architecture*. Each of these behavioral modules was implemented as a finite state machine receiving sensory input from some of the robot's receptors and controlling some of its effectors. While the general idea of a parallel and distributed control architecture was generally accepted, a major criticism of Brooks' original subsumption architecture is that it does not allow for learning. Hence, this type of robot, although autonomous in the sense that during run-time it interacts with the environment on its own, i.e., independent of an observer, still remains heteronomous in the sense that its control mechanism is predetermined by the designer. A number of researchers have therefore pointed out that a necessary element of an artificial agent's autonomy would be the capacity to determine and adapt, at least partly, the mechanisms underlying its behavior [Boden, 1996, Steels, 1995, Ziemke, 1997, Ziemke, 1998]. Much research effort during the 1990s has therefore been invested into making robots 'more autonomous' by providing them with the capacity for self-organization. Typically these approaches are based on the use of computational learning techniques to allow agents to adapt the internal parameters of their control mechanisms. Thus, the use of ANN-controlled robots using learning and/or evolutionary adaptation techniques has become a standard methodology in bottom-up AI research.

From a *cognitive science point of view,* adaptive ANN-controlled robots do not only have the practical advantages described above from an engineering and AI perspective, but they also offer a novel approach to the study of the embodied and situated nature of cognitive processes [Clark, 1997, Sharkey, 1998]. In particular the parallel and distributed nature of weight and unit representations in ANNs, and the fact that these representations can be formed in interaction with an environment, make ANN-controlled robots an interesting approach to the study of cognitive representation. Unlike traditional AI, connectionists do not promote symbolic representations that mirror a pre-given external reality. Rather, they stress self-organization of an adaptive flow of signals between

simple processing units in interaction with an environment, which is compatible with an interactivist [Bickhard, 1995] or experiential [Sharkey, 1997] view of representation, and thus offers an alternative approach to the study of cognitive representation.

However, in most connectionist work the 'environment' is still reduced to input and output values. That means, ANNs, unlike real nervous systems, are typically not embedded in the context of an agent and its environment. Thus, although in a technically different fashion, connectionists were, like cognitivists, mainly concerned with explaining cognitive phenomena as separated from agent-world interaction. Hence, they initially focused on modeling of isolated cognitive capacities, such as the transformation of English verbs from the present to the past tense [Rumelhart, 1986] or the prediction of letters or words in sequences [Elman, 1990]. Thus, early connectionism was mostly concerned with the self-organization of weights to match given input-output pairs, whereas making the connection between inputs, outputs, and internal representations and the actual world they were supposed to represent was still left to the observer. The situation, however, changes fundamentally as soon as ANNs are used as robot controllers, i.e., 'artificial nervous systems' mapping a robot's sensory inputs to motor outputs. Then the network can actually, by means of the robot body (sensors and effectors), interact with the physical objects in its environment, independent of an observer's interpretation or mediation. Dorffner [Dorffner, 1997] has therefore suggested the approach of *Radical Connectionism*, i.e., the use of ANNs for control of and learning in robotic agents, as a natural testbed and a step forward from a *connectionist point of view*.

RNNs play a central role in such approaches to the study of cognitive representation. This is because they account for the (long-term) representation of learning experience in connection weights as well as the (short-term) representation of the controlled agent's current context or immediate past in the form of internal feedback. Peschl [Peschl, 1996] has pointed out that RNNs, like real nervous systems, are "structure determined" (also see Maturana [1980]), which means that, unlike ALVINN, their reaction to environmental stimuli always depends on the system's current state (or structure) and thus is never determined by the input alone. Peschl refers to this as the "*autonomy* of a representational system."

III. RECURRENT NEURAL NETWORKS FOR ADAPTIVE ROBOT BEHAVIOR

A. MOTIVATION

The vast majority of recurrent neural architectures for learning and control of robots and autonomous agents (e.g., [Beer, 1990, Biro, 1998, Meeden, 1996, Nolfi, 1999, Tani, 1996, Tani, 1998, Ulbricht, 1996]) make use of *first-order feedback*. That means, certain neuron activation values are, as in Meeden's architecture (see Figure 2), fed back and used as extra inputs to some of the neurons (typically at the input layer) in a later time step (typically the next one). *Higher-order feedback*, on the other hand, typically modulates/adapts connection weights and/or biases (see, e.g., Figure 3). Unlike in other areas, such as (formal) language recognition, there are only very few cases where higher-order networks have been used as robot control architectures. This section

will therefore demonstrate and analyze both first- and higher-order recurrent neural robot control architectures experimentally. It should be noted in advance that first- and higher-order networks are computationally equivalent [Siegelmann, 1995, Siegelmann, 1998]. That means, every task solved by a higher-order RNN could also be solved by some first-order net. Hence, in the experiments discussed in the following, higher-order RNN-controlled robots will not do anything that could not, at least in theory, be done by first-order RNN-controlled ones. Computational equivalence of network architectures in theory, however, does not say much about their suitability to solve particular tasks in practice. The latter will therefore be investigated here experimentally.

Some of the author's own work [Ziemke, 1996a, Ziemke, 1996b, Ziemke, 1996c, Ziemke, 1997, Ziemke 1998] has been concerned with recurrent robot control architectures inspired by Pollack's *Sequential Cascaded Networks* (SCNs), also referred to as *dynamical recognizers*, which were originally used for formal language recognition [Pollack, 1987, Pollack, 1991]. This architecture (Figure 3) consists of (a) a *function network* mapping input to output and an internal state, and (b) a *context network* mapping the internal state to the next time step's function network weights (including biases). Thus, this type of network utilizes a second-order, multiplicative type of feedback, with the effect that the function network's input-output mapping can change from time step to time step in a context- or state-dependent fashion. The experiments documented in the following subsections will illustrate how this mechanism can be used to realize adaptive *behavioral dispositions* in robots.

B. ROBOT AND SIMULATOR

The experiments discussed here have been carried out using a simulation of a Khepera miniature mobile robot. The Khepera [Mondada, 1993] has a circular body with a diameter of 55 mm and is equipped with eight infrared proximity sensors with a range of approximately 50 mm, six of them at the front

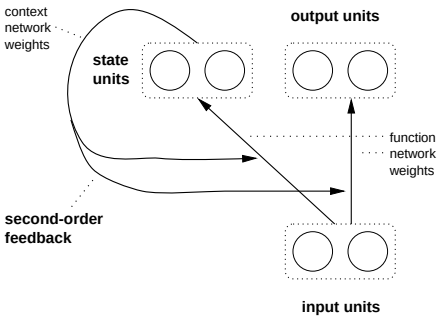

Figure 3. Sequential Cascaded Network (SCN), a second-order recurrent architecture consisting of (a) a function network mapping input to output and internal state, and (b) a context network mapping the internal state to the next time step's function network weights. The solid arrows represent a fully connected layer of weights between two layers of units (indicated by surrounding dotted lines).

and two at the back. It has two motors, which independently control two wheels;

one to the right and one to the left. The wheels can spin forward and backward independently, such that the robot can turn on the spot if they spin in opposite directions.

The robot simulator used here is a slightly adapted version of the one presented by Miglino [1995]. The simulation is based on sensor measurements obtained from a real Khepera robot. It has been shown in a number of papers, (e.g., Nolfi, [1997]), that the simulation of infrared sensors and motors is sufficiently realistic to allow the transfer of controllers trained in simulation to the real robot. An additional ground sensor (see details below) has been used here which is not present on the physical robot, such that the experiments presented here could not be validated on the real robot. Random noise, uniformly distributed in the range of +/- 10% of the maximum sensor readings, has been added to all sensor measurements.

C. ROBOT CONTROL ARCHITECTURES

In the experiments documented here the five different ANN architectures shown in Figure 4 have been used to control the Khepera robot. All of the networks receive five inputs from the robot's sensors (normalized to values between 0 and 1; see details below), and they produce two motor outputs directly controlling the robot's wheels. The output units use the logistic activation function, i.e., the outputs are between 0 and 1, with 0 corresponding to full speed backward rotation, 0.5 corresponding to no motion, and 1 corresponding to full speed forward rotation.

Architecture A is a standard feedforward multilayer perceptron with three hidden units. B is a recurrent network with two memory units, which are used as extra inputs in the next time step (first-order feedback). C is similar to B, but uses an additional layer of three hidden units. The hidden units in A and C use the logistic activation function, such that activation values lie between 0 and 1. D is a Sequential Cascaded Network with two state units, like the one shown in Figure 3. E is a novel variation of the SCN (D), here also referred to as *Extended Sequential Cascaded Network* (ESCN). It has an additional decision unit, which in every time step determines whether to use feedback. The idea behind this extension is that the robot should be able to decide selectively when to change its sensorimotor mapping, instead of (re-) setting the function network weights in each and every time step. Thus, the context network is only used to adapt the function network when the decision unit activation exceeds a certain threshold (here 0.5; the decision unit uses the logistic activation function), otherwise the weights and biases in the function network remain unchanged. The state units in D and E also use the logistic activation function. The output units of the context networks in D and E (cf., Figure 3), however, use a linear activation function such that the function network weights and biases (which are the outputs of the context network) are not limited to values between 0 and 1.

D. EXPERIMENT 1

1. Environment and task

Figure 5 shows the simulated robot in the environment used in experiment 1. The robot is placed in a rectangular environment of 1000 mm x 600 mm, surrounded by walls (the straight lines), which contains a zone (the large circle) outside of which it should keep moving while avoiding collisions, whereas once

it has entered the zone it should simply not leave it anymore. The robot is initially placed with a random orientation in a random position outside the zone, and during learning it is punished for collisions and rewarded strongly for every time step it spends in the zone (for details see below). Moreover, while outside the zone, it is rewarded for moving as quickly and straight as possible and keeping away from walls. It uses four infrared proximity sensors at the front[2]

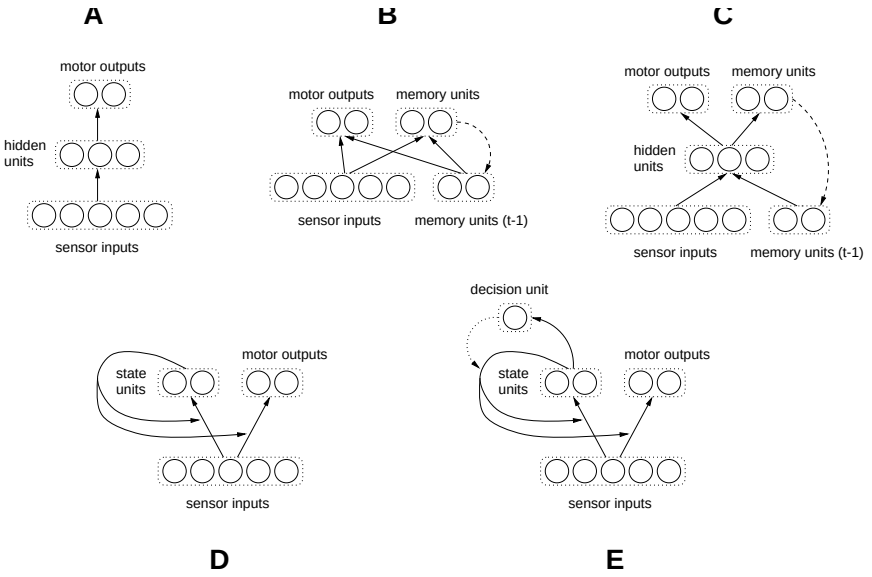

Figure 4. ANN robot control architectures used in the experiments. All networks receive input from the four proximity sensors and the ground sensor and produce two outputs directly controlling the left and right wheels' motors (see text for details). In all cases the solid arrows represent fully connected layer of weights between two layers of units (indicated by surrounding dotted lines). The dashed arrows represent 1:1 copy connections (without weights). A: Feedforward network with three hidden units. B: Recurrent network with two memory units, which are used as extra inputs in the next time step. C: Like B, but with three hidden units. D: Sequential Cascaded Network with two state units (for details see Figure 3). E: Extended Sequential Cascaded Network: like D, but extended with a decision unit, which determines in each time step whether to use feedback.

and a ground sensor,[3] which is only fully active in the time step when the robot passes the black line marking the zone border. As mentioned above, these five sensors provide input to the controller networks (see Figure 4), and the network's two output units directly control the speeds of the two wheels.

The experimental setup is intentionally kept very simple in order to illustrate as clearly as possible the basic mechanisms of behavioral adaptation in recurrent

[2] The left and the right pairs of the six front sensors are averaged and used as if they were one sensor.

[3] As mentioned above, the ground sensor is not actually available on the real robot. A possible physical implementation would be a light sensor directed at the ground below the robot, capable of distinguishing white and black ground.

robot controllers. A significantly more complex scenario will be used in experiment 2 below. However, although this task is fairly simple, it clearly requires some form of memory. Since the ground sensor does not tell the robot whether it is inside or outside the zone, but only when it passes the borderline, the robot needs to 'remember' on which side of the border it currently is. This is necessary, for example, in order to be able to react to the absence of significant sensory stimuli in two completely different ways inside and outside the zone (standstill or circling vs. searching forward motion). Hence, the feedforward networks (architecture A), in both experiments, cannot be expected to achieve the same level of performance as the recurrent networks. They are nevertheless included in the comparisons to illustrate the difference that the use of feedback makes.

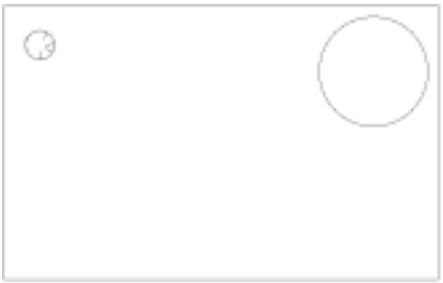

Figure 5. Simulated Khepera robot in environment 1. The large circle indicates the zone the robot should enter and stay in. The small circle represents the robot, and the lines inside the robot indicate position and direction of the infrared proximity sensors used in experiments 1 and 2.

2. Network training

Recurrent networks are known to be difficult to train with, e.g., gradient-descent methods such as standard backpropagation [Rumelhart, 1986] or even backpropagation through time [Werbos, 1990]. They are often sensitive to the fine details of the training algorithm, e.g. the number of time steps unrolled in the case of backpropagation through time (e.g., [Mozer, 1989]). For example, in an autonomous agent context, Rylatt [1998] showed, for one particular task, that with some enhancements Simple Recurrent Networks [Elman, 1990] could be trained to handle long-term dependencies in a continuous domain, thus contradicting the results of Ulbricht [1996] who had argued the opposite. In an extension of the work discussed in the previous section, Meeden [1996] experimentally compared the training of recurrent control networks with (a) a local search method, a version of backpropagation adapted for reinforcement learning, and (b) a global method, an evolutionary algorithm. The results showed that the evolutionary algorithm in several cases found strategies, which the local method did not find. In particular, when only delayed reinforcement was available to the learning robot, the evolutionary method performed significantly better due to the fact that it did not at all rely on moment-to-moment guidance [Meeden, 1996].

In the experiments documented here, all control networks have therefore been trained using an evolutionary algorithm very similar to the one used by Nolfi [1997], a genetic algorithm [Holland, 1975] evolving an initially randomized

population of 100 individuals over 5000 generations. For architectures A, B, and C the artificial genotype of each individual encodes all the connection weights (including biases) of a complete control network as a single bitstring. Each real-valued weight (between –10.0 and +10.0) is represented by a string of 8 bits. For architectures D and E, in which function network weights change dynamically, the genotype encodes the connection weights in the context network plus initial state unit activation values, such that the initial function network weights can be derived by propagating the initial state through the context network.

To evaluate their fitness each individual of every generation is used to control the robot during a trial period of 400 time steps, starting from a random position outside the zone and with a random orientation. While outside the zone individuals score between 0.0 and 1.0 fitness points per time step, being rewarded for moving as fast and as straight as possible while minimizing encounters with walls. While inside the zone, they simply receive 100 fitness points for every time step they remain inside. To encourage the selective use of feedback in networks of architecture E, they only score fitness points during those time steps when they are not using feedback, i.e., when they are not using the context network to re-set function network weights and biases. For all networks, the points collected during the 400 time steps are summed up at the end of the evaluation period to determine the individual's overall fitness. Of the 100 individuals the 20 'fittest' ones of each generation are selected, and each of them produces five 'offspring' which will be part of the next generation, which thus again consists of 100 individuals. 'Reproduction' is carried out by creating a copy of the artificial genotype with a mutation probability of 1% for each of the bits. The reason that only mutation, but no crossover, is used here is that several researchers, e.g., Meeden [1996], have shown/argued that the use of crossover does not improve performance when evolving ANN robot controllers. This was also the case in the experiments of Nolfi [1997], from whom also the other evolutionary algorithm parameter settings used here (population size, mutation rate, bitstring representation) have been adopted. Initial experiments were carried out with a variety of alternative parameter settings; none of which, however, resulted in significantly better results. For both experiments 1 and 2, ten evolutionary runs, starting from different random initial populations, were carried out for each of the five architectures.

3. Results

Networks of architectures B, C, D, and E quickly evolved to robustly solve the task. The evolutionary process was nevertheless continued for 5000 generations to ensure that evolution had converged. Networks of architecture E exhibit best overall performance, although they are not able/allowed to score points while using feedback (cf. previous subsection). For each of the 5000 generations Figure 6 shows, averaged over all ten evolutionary runs for architecture E, the fitness of the best individual, the mean fitness of the 20 best individuals selected for reproduction, and the mean fitness of all individuals in the population. As the figure shows, the best individuals achieve a fitness score around 30000, corresponding to approximately 300 time steps (out of 400) spent in the zone.

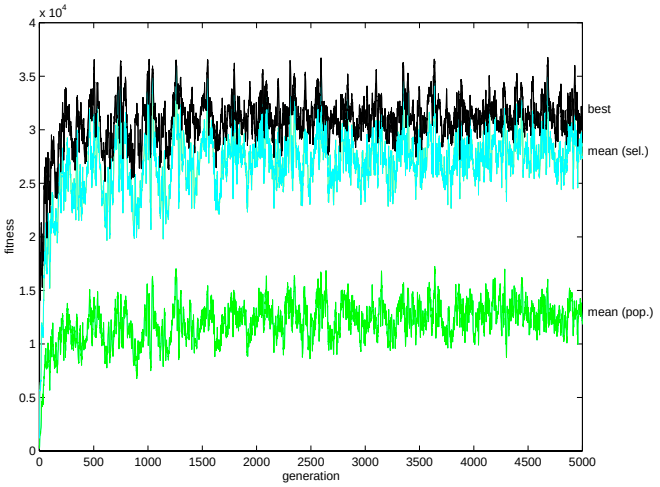

Figure 6. Fitness values for networks of architecture E in experiment 1. For each generation the figure shows the fitness of the best individual ('best'), the mean fitness of the 20 best individuals selected for reproduction ('mean (sel.)'), and the mean fitness of all individuals in the population ('mean (pop.)'). The values shown are rolling averages over three generations, averaged over all ten evolutionary runs.

Table 1 illustrates the performance differences between architecture E and the other architectures using the three fitness measures illustrated in Figure 6. The values shown are the average differences between the architectures during all 5000 generations (and in parentheses the differences during the last 500 generations, which allows the comparison of already trained networks, relatively independent of possible differences in learning speed). Thus, for example, the best individuals in the populations of architecture A have a fitness that is on average 40.8% lower than that of networks of architecture E during all 5000 generations, and 40.1% lower during the last 500 generations.

	Best individuals	mean fitness [20 selected]	Mean fitness [whole population]
A	-40.8% (-40.1%)	-51.2% (-49.5%)	-44.5% (-42.0%)
B	-5.5% (-6.6%)	-1.6% (-2.7%)	+6.6% (+3.8%)
C	-7.0% (-10.4%)	-4.4% (-8.9%)	-10.1% (-14.4%)
D	-2.2% (-3.4%)	-2.1% (-4.7%)	-0.7% (-3.2%)

Table 1. Performance differences between architecture E and other architectures in experiment 1. All differences are stated in percent of the performance of architecture E (as illustrated in Figure 7). Values are averaged over all 5000 generations (in parentheses over the last 500 generations) of all ten evolutionary runs.

Table 1 shows that architectures A, B, C and D perform worse than E according to basically all of the three performance measures. The only exception is that networks of architecture B actually achieve a higher mean fitness when looking at the whole population; they do however achieve lower values when compared to the best networks. Not surprisingly, the feedforward networks (A)

perform significantly worse than all recurrent architectures. Because of their lack of feedback they cannot 'remember' whether they have passed the circle border or not. Recurrent architectures B, C, and D, on the other hand, in experiment 1 come relatively close to the performance of E.

4. Analysis

This subsection will present some analysis of different recurrent networks successfully solving the task. All networks analyzed here are taken from the final generations, and they are representative of how networks of the respective architecture evolved to solve the task, although there are of course differences between individuals in a population as well as between populations in different evolutionary runs. Figure 7 illustrates the performance of a successful robot controller of architecture B. The robot's position at each time step is indicated by a circle. The robot in this case starts off facing the wall to the left, moves forward at maximum speed, correctly avoids the walls twice by turning right, finally enters the zone after about 120 time steps, and keeps spinning in place there for the remaining time steps, thus maximizing its fitness.

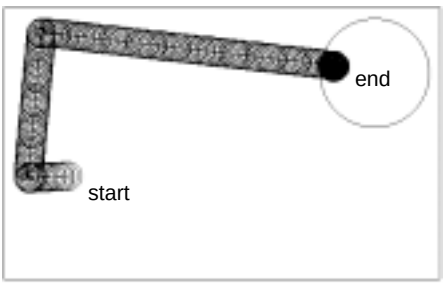

Figure 7. Example trajectory for a robot controller (architecture B) in experiment 1. The robot's position in each time step is indicated by a circle, and its heading by the line inside that circle. The robot correctly avoids the walls twice by turning right, enters the zone, and keeps spinning in place.

Figure 8 further illustrates the robot controller's performance, showing the unit activation values, and the fitness points collected during the 400 time steps. Figure 9 shows the controller network's connection weights (including biases). Together Figures 8 and 9 allow some analysis of how the robot controller solves the task. Initially the robot receives no sensory input, apart from noise, as there are no objects nearby. Due to the strongly positive biases for both left and right motor output unit (see Figure 9), the robot moves forward at maximum speed as long as possible. Twice it quickly avoids walls appearing at the front/left (see Figures 8 and 7), by temporarily inhibiting the right motor using the large negative weights between the sensors and the right motor output unit (see Figure 9). This makes the right wheel spin backwards, which in combination with the left wheel's continued full speed forward motion makes the robot turn away from the wall. So far both memory units have been inactive due to their negative biases. But, when entering the zone, the ground sensor activates memory unit 1 through the large positive weight between the two, and during the remaining time steps (when the ground sensor no longer detects the

borderline) memory unit 1 keeps (re-)activating itself using a large positive weight (see Figure 9). Memory unit 1 also inhibits the right motor through a large negative weight (see Figure 9). Thus, the robot in the absence of significant sensory stimuli keeps spinning in place, such that maximum fitness points can be collected for the rest of the evaluation period.

Figure 8. Activation and fitness values during the 400 time steps of the evaluation trial for the robot controller (architecture B) illustrated in Figure 8 (experiment 1). Activation values are illustrated as vertical black lines whose height corresponds to the represented value. All unit activation values lie between 0 and 1 (full height black line) (NB. In the case of the motors 0 corresponds to full speed backward rotation, 0.5 to no motion, and 1 to full speed forward rotation). The fitness value is either 0 (no line), between 0 and 1 (short line), or 100 (full height black line).

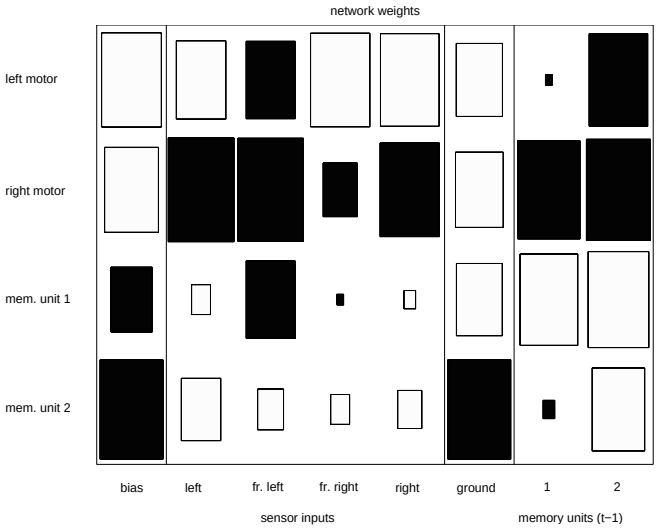

Figure 9. Connection weights (including biases) for the robot controller (architecture B) illustrated in Figures 7 and 8. Positive weights are shown as white rectangles and negative ones in black, with the rectangles' areas corresponding to the weight values.

Networks of architecture C (B plus an additional hidden layer) typically evolved to solve the task in a way very similar the one described above. Since architecture C in experiment 1 actually achieved slightly worse performance than the theoretically less powerful B (see Table 1), it is not discussed in further detail here. Figure 10 instead illustrates the performance of a successful robot controller of architecture D (Sequential Cascaded Network). The robot uses a

similar behavioral strategy as the one above, although internally realized differently as we will see. The robot starts off in the same position, moves forward at maximum speed, correctly avoids the walls twice, this time by turning left, finally enters the zone after about 110 time steps, and, as before, keeps spinning in place there.

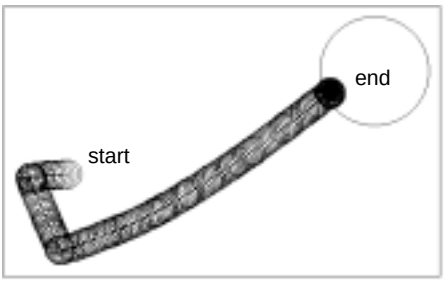

Figure 10. Example trajectory for a robot controller (architecture D) in experiment 1. The robot correctly avoids the walls twice by turning left, then enters the zone and stays there, spinning in place.

Figure 11 further illustrates the robot controller's performance, showing as before the unit activations and fitness values during the evaluation period, but now also the motor units' biases, which in architecture D can be adapted dynamically by the context network. Figures 12 and 13 show the weights in the function network, both outside and inside the zone.

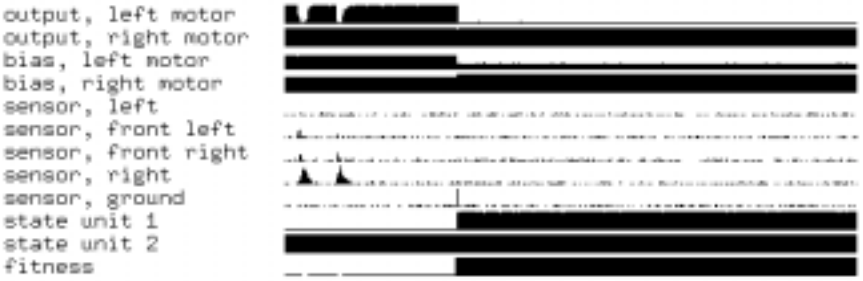

Figure 11. Activation values, motor biases, and fitness points during the 400 time steps of the evaluation trial for the robot controller (architecture D) illustrated in Figure 10 (experiment 1). Biases are shown in the range of –10 (no line) to +10 (full height black line).

Together these figures provide some insight into how the second-order network (D) solves the task. Initially, outside the zone, state unit 1 is inactive, due to a negative bias, while state unit 2 is active, due to a positive bias. The function network, which results from propagating the state unit values through the context network, has strongly positive biases for both left and right motor (see Figure 12). This makes the robot move forward fast in the absence of significant sensory stimuli. When encountering a wall at the front/right the left

motor output unit is inhibited by the sensors, such that the robot avoids the wall by turning left. Outside the zone state unit 1 is inactive (see Figure 11), but it is activated by the ground sensor through a large positive weight (see Figure 12) when passing the borderline. The change in state unit activation leads to a different weight configuration (see Figure 13) in the function network from the next time step. Now both state units have positive biases, and the left motor has a negative bias while the bias for the right motor remains strongly positive. Accordingly, the robot now keeps spinning in place inside the zone and thus collects maximum fitness points during the rest of the evaluation period.

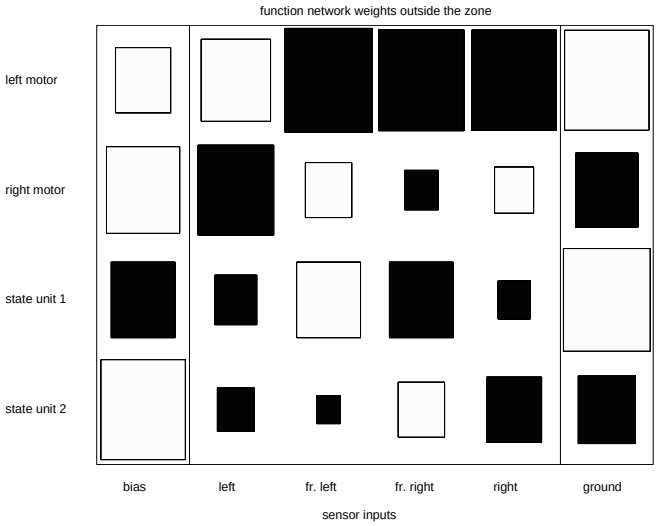

Figure 12. Function network weights as they are used outside the zone by the robot controller (architecture D) illustrated in Figures 11 and 12.

Since the Sequential Cascaded Network (architecture D) makes use of its context network in every time step, the state units have to reflect the current requirements on the function network. In the above case, for example, there have to be two different internal states corresponding to different weight configurations in the function network, embodying appropriate sensorimotor

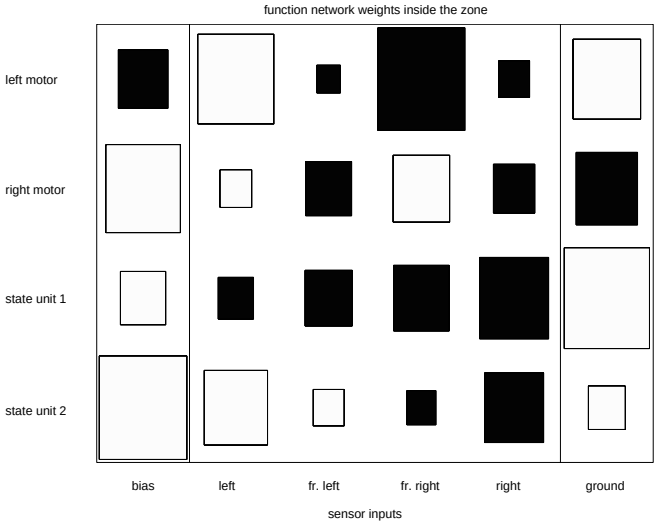

Figure 13. Function network weights as they are used inside the zone by the robot controller (architecture D) illustrated in Figures 10 and 11.

mappings for 'outside' and 'inside.' In the Extended Sequential Cascaded Network (architecture E), on the other hand, where the extra decision unit selectively determines whether to use the context network or not, there is not necessarily a 1-1 mapping between current state unit activations and current function network weight configuration. This is illustrated in the following example of a successful robot controller of architecture E. The trajectory taken by the robot is very similar to the one shown in Figure 7, and therefore not shown here again. Activation and fitness values are shown in Figure 14. It can be seen that state unit 2 is active during both forward motion outside the zone and spinning inside the zone, whereas it is less active during turning/spinning outside the zone. State unit 1, on the other hand, is only fully active while passing the borderline. The decision unit is apparently only active while turning away from walls and while passing the borderline, i.e., feedback through the context network is actually only used in these situations.

Figures 15 and 16 show the function network weight configurations as they are used outside (during forward motion) and inside the zone, respectively. Again, it can be seen that both motors have strongly positive biases during forward motion outside the zone, whereas one of them is negative inside the zone, leading to spinning in place, as in the previous examples. The state units, on the other hand, do not change their biases significantly. State unit 1 is activated slightly by sensory stimuli at the left/front, but only activated fully by the ground sensor through the large positive weight between them (see Figure 15).

Figure 14. Activation values, motor biases, and fitness points during the 400 time steps of the evaluation trial for a robot controller of architecture E (experiment 1). The decision unit is illustrated as rounded to a binary value, with a black line corresponding to an active decision unit (activation exceeds 0.5, in which case the context network will be used) and white corresponding to an inactive decision unit (resulting in no use of feedback).

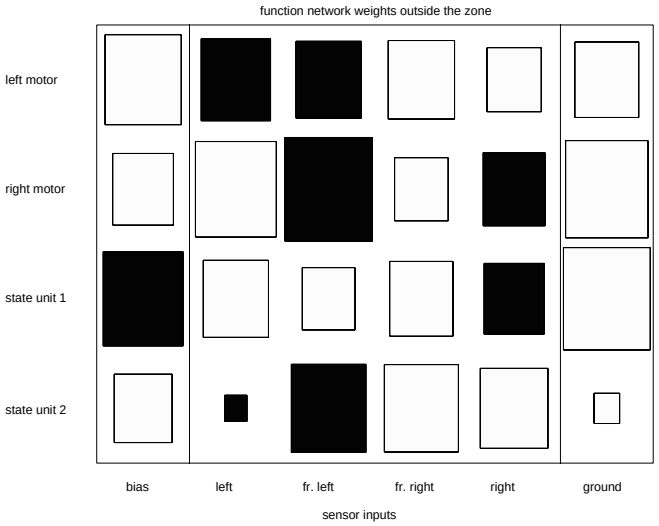

Figure 15. Function network weights as they are used during forward motion outside the zone by the robot controller (architecture E) illustrated in Figure 14.

E. EXPERIMENT 2

1. Environment, task and training

Having illustrated the general workings of first- and higher-order RNN control of adaptive robot behavior with a simple example above, we can now turn to a more complex task. The setup for experiment 2 is illustrated in Figure 18. Again, the robot, using the same sensors and control architectures as in experiment 1, is placed in an environment of 1000 mm x 600 mm, which now contains 11 identical round objects, five of them inside a zone and six of them

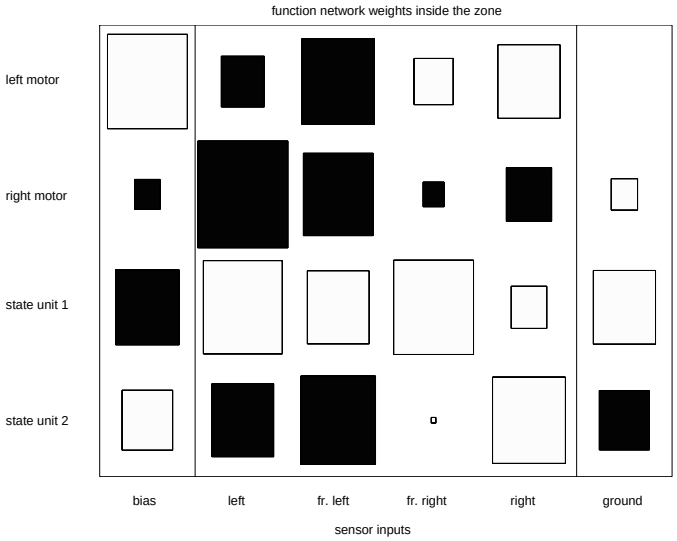

Figure 16. Function network weights as they are used inside the zone by the robot controller (architecture E) illustrated in Figure 15.

outside. As in experiment 1, the robot is initially given a random orientation and placed in a random position outside the zone. The robot's task is to avoid collisions with the objects outside the zone (those are solid objects), but to 'collect' those inside the zone by 'hitting' them, which makes them disappear immediately. In this experiment the robot receives –500 fitness points for collisions outside the zone (possibly multiple times if persisting to bump into the obstacle) and +500 points for 'collisions' inside, but in this case always only once per 'collected' object since it disappears immediately. The robot is not punished for leaving the zone, but inside it receives 0.0 to 4.0 fitness points per time step for moving as fast and as straight as possible and approaching objects. Outside the zone, as in experiment 1, the only positive reward the robot can get is 0.0 to 1.0 fitness points per time step for moving as fast and as straight as possible and staying away from objects.

Apart from the different fitness function/reward scheme, training of the control networks is carried out exactly as described for experiment 1, with the exception that every individual now in each generation is given two evaluation trials of 400 time steps each, starting from different random positions outside the zone.

2. Results

As for experiment 1, the evolutionary process was carried out for 5000 generations, although highly fit and robust control networks evolved far more quickly than that. Again, the Extended Sequential Cascaded Networks (architecture E) exhibited best overall performance. For each of the 5000

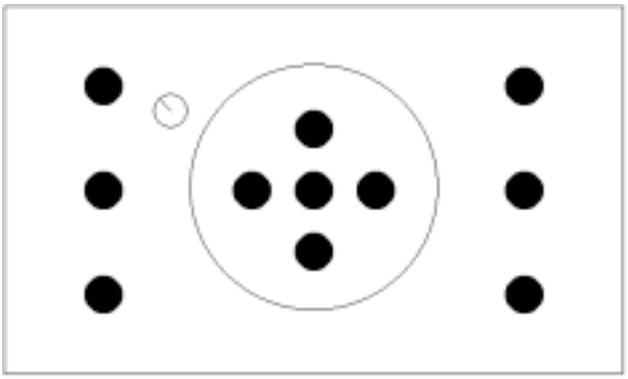

Figure 17. Simulated Khepera robot in environment 2. The black circles represent objects, which are to be avoided outside the zone (indicated by the large circle), but to be 'collected' inside.

generations, Figure 18 shows, averaged over all ten evolutionary runs for architecture E, the fitness of the best individual and the mean fitness of the 20 best individuals selected for reproduction. It turned out that the better an architecture did on these two performance measures, the lower was actually its mean fitness over the whole population. The latter values were strongly negative, due to the fact that some of the controllers in each generation persisted to bump into an object outside the zone, thus collecting –500 points in every time step. This indicates that there is a thin line between highly fit and highly unfit robot controllers, in the sense that a slight mutation of an individual, which achieved high fitness in generation t, can easily result in an individual, which performs very badly in generation t+1. Given the large difference in reward for hitting identical objects inside and outside the zone respectively, this is perhaps not too surprising. The mean fitness of the population was therefore not used as a performance measure in experiment 2. Instead, Figure 19 illustrates the increase in the number of correctly collected objects during the 5000 generations, averaged over the whole population in all ten evolutionary runs for architecture E. It can be seen that the average evolved robot controller collects only about three objects (out of a maximum of ten; five in each of the two evaluation trials). It should, however, be remembered that each generation consists of 100 mutants, i.e., random mutations of (relatively) successful networks, of which only the best 20 are selected. As Figure 18 shows, the 20 best individuals in the end achieve, on average, a fitness score around 3700, corresponding to approximately seven collected objects, and the very best ones often manage to collect all ten objects.
Table 2 illustrates the performance differences between architecture E and the other architectures using the three fitness measures illustrated in Figures 18 and 20. As in Table 1, the values shown are the average differences between the architectures during all 5000 generations (and in parentheses the differences during the last 500 generations, which allows the comparison of already trained networks, relatively independent of possible differences in learning speed).

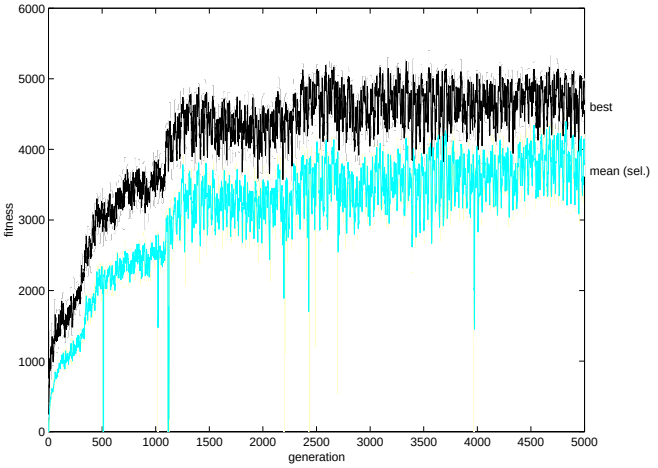

Figure 18. Fitness values for networks of architecture E in experiment 1. For each generation the figure shows the fitness of the best individual (best) and the mean fitness of the 20 best individuals selected for reproduction (mean (sel.)). The values shown are rolling averages over three generations, averaged over all ten evolutionary runs.

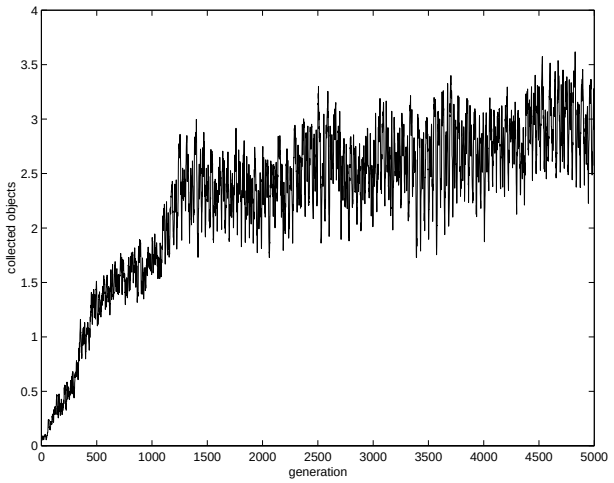

Figure 19. Number of correctly collected objects for networks of architecture E in experiment 2. The values shown are rolling averages over three generations, averaged over all ten evolutionary runs.

Again, architectures A, B, C, and D exhibit worse performance than E according to the three performance measures. The only exception is that, on average, the best networks of architecture C achieve a 1.1% higher fitness during the final 500 generations. However, according to the other performance measures, architecture C, like B and D, performs significantly worse than E. Moreover, architecture B this time outperforms D, but it is no longer clearly better than C. The differences between E and the other recurrent architecture are larger here than in experiment

1, which might be due to the increased complexity of the task. Again, however, all recurrent architectures clearly outperform A due to its lack of feedback.

	best individuals	mean fitness [20 selected]	objects collected [whole population]
A	-68.6% (-69.2%)	-57.2% (-48.1%)	-90.7% (-90.1%)
B	-9.5% (-3.9%)	-10.7% (-11.9%)	-14.6% (-13.8%)
C	-9.6% (+1.1%)	-16.0% (-8.9%)	-32.1% (-26.4%)
D	-23.8% (-20.3%)	-13.8% (-14.7%)	-32.2% (-31.8%)

Table 2. Performance differences between architecture E and other architectures in experiment 2. All differences are stated in percent of the performance of architecture E (as illustrated in Figures 19 and 20). Values are averaged over all 5000 generations (in parentheses over the last 500 generations) of all ten evolutionary runs.

3. Analysis

As for experiment 1, this subsection will present some analysis of successful robot controllers in experiment 2. The analysis will here be limited to networks of architectures E, which exhibited best performance, and B, which evolved similar solutions as C, and is to some degree representative of first-order recurrent networks. Figure 20 illustrates the performance of a successful robot controller of architecture B. To allow better understanding of the robot's behavior during this trajectory the collected objects are shown in their original positions; it should however be noted that for the robot, as discussed above, they disappear upon first contact. The robot starts off facing the upper left obstacle. It turns away from it, faces the wall, and turns away from that also (keeping safe distances in both cases). It enters the zone, collects the left object, and leaves the zone again. When facing the lower wall it starts moving in a curve to the right, which takes it back into the zone. When detecting the upper object on its right it performs a sharp turn to the right to be able to collect that object, and performs another right turn to collect the center object as well. It continues to move straight ahead which allows it to collect the lower object. It leaves the zone, and as before, when detecting the wall ahead starts moving in a curve to the right, which takes it back into the zone. It leaves and enters the zone once more, and eventually the evaluation period ends after 400 time steps.

Figure 21 shows the activation values and fitness points for the robot controller network during this trajectory, and its weights are shown in Figure 22. It can be seen that the network has evolved a strong positive bias for the left motor and a weaker positive bias for the right motor. This gives the robot a tendency to move forward turning slightly to the right in the absence of significant sensory stimuli, at least as long as it outside the zone (see Figure 20). Moreover, the proximity sensors (when active) have a stronger inhibitory effect

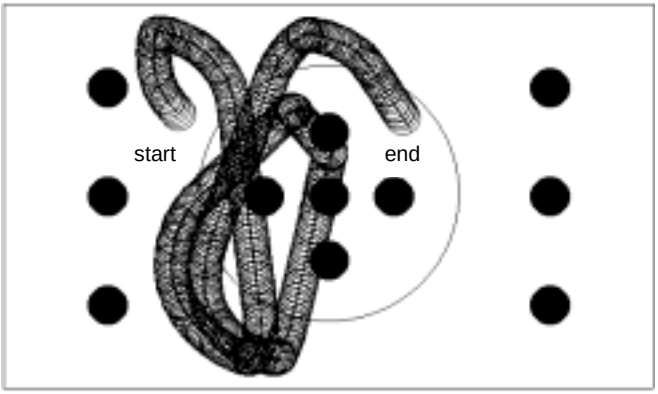

Figure 20. Example trajectory for a robot controller (architecture B) in experiment 2. The robot successfully collects four objects in the zone and correctly avoids objects and walls outside the zone.

on the right motor than on the left. Thus, the robot outside the zone avoids objects by sharp turns to the right. When entering the zone the ground sensor fully activates memory unit 1, but it only slightly activates memory unit 2, which has a larger negative bias and a smaller positive weight. Memory unit 1 keeps (re-)activating itself as long as the robot remains in the zone and it also has a positive influence on the right motor, with the result that inside the zone the robot tends to move forward in a straighter line than outside. As Figure 20 illustrates, inside the zone the robot turns to the right towards objects in order to collect them instead of turning away from them. This is achieved through the combination of the right proximity sensors' strong inhibitory effect on the right motor, memory unit 1's positive influence on the right motor and its slightly negative influence on the left motor (see Figure 22). The latter makes the left wheel slow down slightly too, with the effect that the robot does not turn away from the object before reaching it. When passing the borderline on the way out of the zone, the combination of active ground sensor and active memory unit 1 finally activates memory unit 2. The activity of memory unit 2 during a single time step suffices to trigger a gradual decrease in activity in memory unit 1 over 2-3 steps (see Figure 21), whose self-activation does not suffice to keep itself activated once it is no longer fully active.

For architecture E, roughly speaking, two types of internal organizations evolved in experiment 2. In the one type 'inside' and 'outside' are reflected by distinct state unit activations, whereas in the other type they are only reflected by different function network weight configurations, as demonstrated earlier for a controller network of architecture E in experiment 1. Since the latter type has already been discussed in this paper, a network of the former type will be analyzed in the following. Figure 23 shows a characteristic trajectory of a upper left obstacle. It turns away from it to the left, enters the zone, and collects

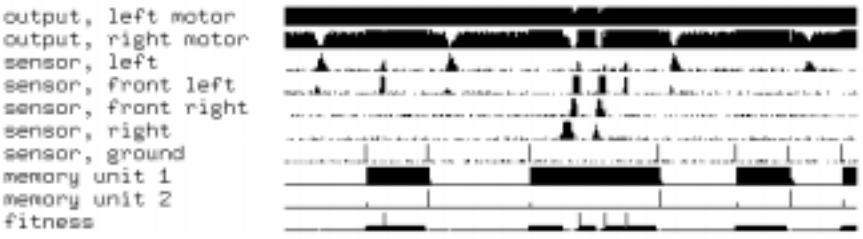

Figure 21. Activation and fitness values during the 400 time steps of an evaluation trial for the robot controller (architecture B) illustrated in Figure 21(experiment 2). The fitness value is either 0 (no line), between 0 and 1 (short line) outside the zone, between 0 and 4 (slightly longer line) inside the zone, or 100 (full height black line) when correctly 'collecting' an object.

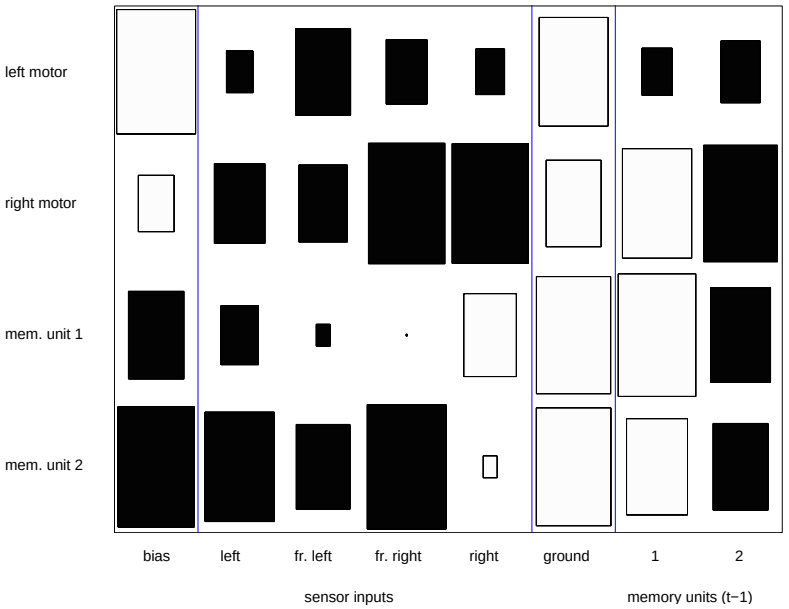

Figure 22. Connection weights for the robot controller network (architecture B) illustrated in Figures 20 and 21.

three objects on its first pass through the zone, turning slightly to the left towards each of them. As soon as it has left the zone it starts moving in a semi-circle to the left, which takes it back into the zone. In the zone it starts moving straight ahead again, takes a slight turn to the right to collect the upper object, and continues straight ahead out of the zone. The same pattern is repeated: as soon as it leaves the zone, it moves in a semi-circle to the left, which takes it back into the zone, where it starts moving straight forward again. Once more it performs a slight turn to the right to collect an object it would otherwise have missed. It continues to move straight ahead, leaves the zone, returns in another

semi-circle, enters once more, and moves straight ahead until the evaluation period ends.

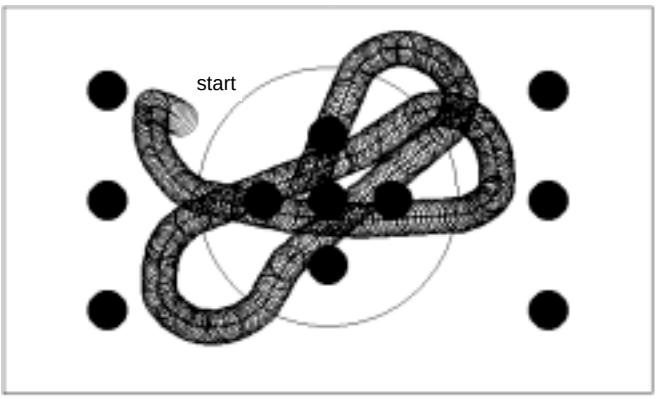

Figure 23. Example trajectory for a robot controller (architecture E) in experiment 2. The robot successfully collects all the objects in the zone, and outside it moves in semi-circles taking it back into the zone.

A look at Figure 24, which illustrates the unit activation and fitness values during the above trajectory, shows that this controller network is slightly smarter than that of architecture B discussed above. Instead of bouncing off obstacles back into the zone as that robot did (see Figures 20 and 21), the one illustrated in Figures 23 and 24 carries out its semi-circle strategy[4] even in the absence of significant sensory stimuli to which it could react.

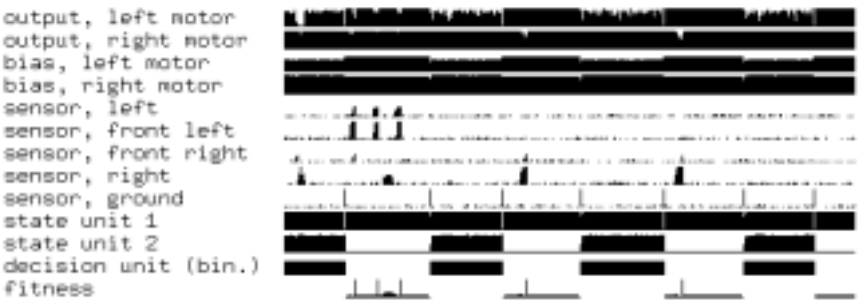

Figure 24. Activation values, motor biases, and fitness values during the 400 time steps of an evaluation trial for the robot controller (architecture E) illustrated in Figure 24 (experiment 2).

That means, in a sense, it 'knows' how to return back into the zone on its own, instead of doing so by merely reacting appropriately to obstacles. Figures 25 and

[4] This type of strategy also evolved in networks of architecture B, C, and D in some cases, but not as often as for architecture E.

26 illustrate how this is realized internally, showing the function networks weights as they are used inside the zone and outside in a semi-circle, respectively.

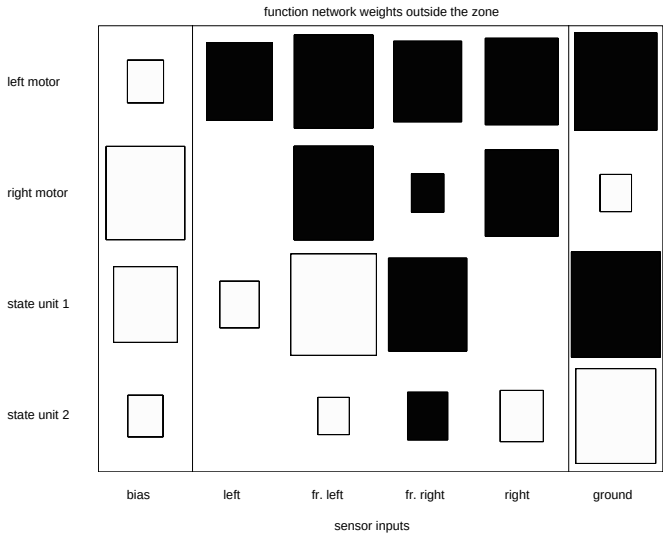

Figure 25. Function network weights as they are used outside the zone (during a semi-circle) by the robot controller (architecture E) illustrated in Figures 23 and 24.

As shown in Figures 25 and 26, the network has again evolved substantially different biases for use inside and outside the zone respectively. In the former case (see Figure 25) the robot has a strong positive bias for its right motor output unit and a weak positive bias for the left one. As a result the robot moves in a semi-circle to the left in the absence of sensory stimuli, which, as illustrated in Figure 23, is a good strategy for getting back into the zone once the robot has left it. The beginning of the trajectory, however, also shows that the control network does of course not ignore sensory stimuli. The obstacle it faces in the beginning is correctly avoided by a sharp left turn (see Figures 23 and 24), which is caused by the proximity sensors' influence which inhibits the left motor more strongly than the right one (see Figure 25). Outside the zone both state units are active due to their positive biases; the activation of state unit 2, however, varies slightly between 0.8 and 1.0, and with it the speed of the left motor (see Figure 24). Furthermore, the decision unit is actually active all the collect any fitness points during this time. When the ground sensor gets activated it inhibits state unit 1, but activates state unit 2, which results in the function network configuration shown in Figure 26. State unit 2 now has a time while the robot is outside the zone, as a result of which the robot does not negative bias and can only be activated again by a high ground sensor reading.

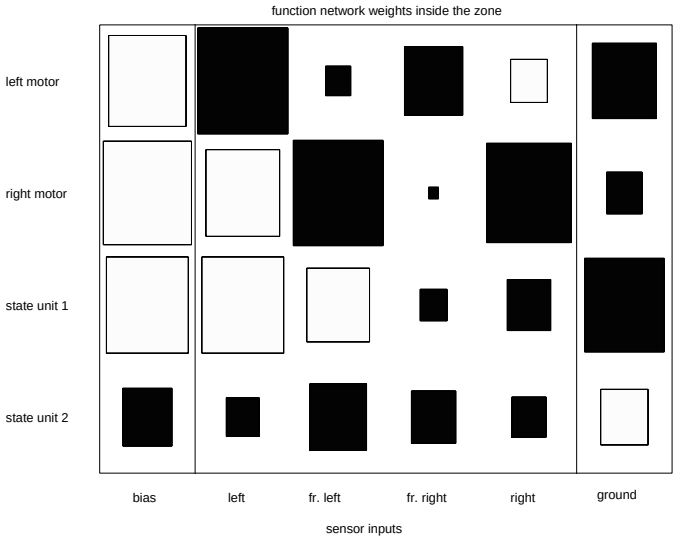

Figure 26. Function network weights as they are used inside the zone by the robot controller (architecture E) illustrated in Figures 24 and 25.

Left and right motors now both have strong positive biases, which results in relatively straight forward motion inside the zone. The connection weights between the proximity sensors are now set such that different sensors inhibit the two motor output units. The left motor is inhibited by the left sensor, which enables the robot to perform (slight) left turns towards objects, as it does during its first pass through the zone when collecting three objects (see Figure 24). The right motor, on the other hand, is inhibited by the sensor on the right. This allows the robot to perform turns to the right in order to collect objects it would have missed otherwise, as it does during its second and third passes through the zone (see Figure 23 and 24).

It should be noted that the robot controller here correctly 'switches' back and forth between different function network configurations several times, and each time the appropriate weights and biases are set or restored. Thus, again the Extended Sequential Cascaded Network (architecture E) realizes the required behavioral adaptation in an interesting way. In this case it switches between two radically different sensorimotor mappings, and thus behavioral dispositions, i.e., at any point in time the robot 'knows' where (on which side of the zone border) it currently is and behaves accordingly. Thus these controllers realize a form of *virtual modularity* through feedback, i.e., they dynamically adapt their behavioral dispositions, such that they act *as if* they were using different modules for different situations/contexts. It is worth noting that this allows the controlled robot to exhibit adaptive, context-dependent responses to otherwise identical stimuli of two types. Firstly, the round objects, which look identical inside and outside the zone, assume different *functional tones* (see von Uexküll [1928], Ziemke [2000]) for the robot. They are therefore correctly attributed completely different meanings, leading to opposite responses. Secondly, although containing no information about direction whatsoever, the zone border

stimulus triggers two opposite responses, switching from an inside behavioral disposition to one for outside, and the other way round. Thus, the robot in this case, unlike ALVINN not able to rely on an external scaffold/structure to guide its behavior, has formed an internal structure that allows it to reliably interact with its environment. This behavioral structure, and the mechanisms realizing it, can be considered *interactive representations* (see Bickhard [1995], Dorffner [1997]), formed by the agent in interaction with its environment, not as an abstract model of the world, but for the purpose of guiding its own behavior.

IV. SUMMARY AND DISCUSSION

This chapter started off with a discussion of the relevance of RNNs for robot learning and control to various lines of research in cognitive science, connectionism, AI, and engineering of robot control systems. In particular it was argued that RNN-controlled robots receive much attention due to their capacity to form internal control structures and representations in interaction with an environment. These issues were then illustrated and analyzed in quite some detail, comparing four different first- and higher-order recurrent robot control architectures in two experiments that required the controlled robot to exhibit context-dependent behavior. The quantitative results showed that best performance in both experiments was achieved by architecture E, the Extended Sequential Cascaded Network (ESCN), a novel variation of Pollack's SCN. However, whether this architecture is actually better suited for robot control or the results are really due to the details of the experimental setups or the evolutionary training procedure (the latter was slightly different for the ESCN), can only be determined by further extensive experimentation.

Further analysis of the experiments showed that the way this type of RNN constructs and utilizes internal structures for adaptive robot behavior closely corresponds to what Peschl in his discussion of RNNs referred to as "representation without representations":

> The internal structures do not map the environmental structures; they are rather responsible for generating functionally fitting behavior which is *triggered* and *modulated* by the environment and *determined* by the internal structure (. . . of the synaptic weights). It is the result of *adaptive* phylo- and ontogenetic processes which have changed the architecture over generations and/or via learning in an individual organism in such a way that its physical structure embodies the dynamics for maintaining a state of equilibrium/homeostasis. [Peschl, 1996]

Thus, RNNs in a sense offer a 'middle way' between (a) the explicit world models of traditional AI, lacking grounding in and interaction with the world they are supposed to represent, and (b) the world-dependence of purely reactive systems, like ALVINN, which rely on the world to 'puppeteer' their behavior. The internal structures and representations formed in RNNs, in the self-organization of adaptive behavior in interaction between a robot and its environment, are the result of a structural coupling between the two, which ensures their structural congruence (see Maturana [1987], Varela [1991]). Thus, they are the result of a constructive process (see Peschl [1996], Ziemke [1999], Ziemke [2000]), and they reflect an agent's *subjective* embedding in the world,

allowing it to attribute varying meaning to stimuli according to its own current behavioral disposition.

Apart from the investigation of higher-order RNNs for more complex robot tasks, the author's current work includes further investigations of (a) the practical suitability of first- and higher-order RNNs for different tasks, in particular their amenability to training through backpropagation and evolutionary algorithms, and (b) cognitive and semiotic aspects of representation formation and sign usage in RNN-controlled robots and their implications for the possibilities and limitations of robot autonomy and subjectivity [Ziemke, 2000].

ACKNOWLEDGMENTS

This work has been supported by funding from The Foundation for Knowledge and Competence Development (1507/97), Sweden. The author would like to thank Noel Sharkey, Larry Medsker, Fredrik Linåker, and Henrik Jacobsson for their useful feedback on earlier versions of the work, and Lars Niklasson for providing Figure 1. Moreover, the author would like to thank Henrik H. Lund and Stefano Nolfi who did most of the development of the kepsim simulator, which has been used (in slightly adapted form) to implement the experiments documented in this paper.

REFERENCES

Beer, R. D., *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*, Academic Press, Cambridge, 1990.

Bekey, G. and Goldberg, K. Y., Eds., *Neural Networks in Robotics*, Kluwer, Boston, 1993.

Bickhard, M. H. and Terveen, L., *Foundational Issues in Artificial Intelligence and Cognitive Science - Impasse and Solution,* Elsevier, New York, 1995.

Biro, Z. and Ziemke, T., Evolving visually guided approach behaviour in recurrent artificial neural network robot controllers. *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, MIT Press, Cambridge, 1998.

Boden, M. A., Autonomy and artificiality, in *The Philosophy of Artificial Life*, Boden, M. A., Ed., Oxford University Press, Oxford, 95, 1996.

Brooks, R. A., A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation*, 2(1), 14, 1986.

Brooks, R. A., Intelligence without reason, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAII-91),* Morgan Kauffmann, San Mateo, 569, 1991.

Brooks, R. A., Grossberg, S., and Optican, L., Eds., *Neural Networks*, 11(7-8), 1998. Special issue on *Neural Control and Robotics: Biology and Technology*.

Clark, A., *Being There_Putting Brain, Body and World Together Again*, MIT Press, Cambridge, MA, 1997.

Dorffner, G., Radical connectionism_a neural bottom-up approach to AI, in *Neural Networks and a New Artificial Intelligence*, Dorffner, G., Ed., International Thomson Computer Press, London, 93, 1997.

Elman, J., Finding structure in time, *Cognitive Science*, 14, 179, 1990.

Grey Walter, W., An imitation of life, *Scientific American*, 182, 42, 1950.

Grey Walter, W., *The Living Brain,* Norton, New York, 1953.

Harnad, S., The symbol grounding problem, *Physica D*, 42, 335, 1990.

Holland, J. H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.

Maturana, H. and Varela, F., *Autopoiesis and Cognition: The realization of the living*, Reidel, Boston, 1980.

Maturana, H. and Varela, F., *The Tree of Knowledge: The Biological Roots of Human Understanding*, New Science Library, Boston, 1987.

Meeden, L. A., *Towards planning: Incremental investigations into adaptive robot control*, PhD dissertation. Indiana University, 1994.

Meeden, L. A., An incremental approach to developing intelligent neural network controllers for robots, *IEEE Transactions on Systems, Man, and Cybernetics*, 26, 1996.

Meeden, L. A., McGraw, G., and Blank, D., Emergence of control and planning in an autonomous vehicle. *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society*, Lawrence Erlbaum, Hillsdale, 735, 1993.

Miglino, O., Lund, H. H., and Nolfi, S., Evolving mobile robots in simulated and real environments, *Artificial Life*, 2(4), 417, 1995.

Mondada, F., Franzi, E., and Ienne, P., Mobile robot miniaturisation: A tool for investigation in control algorithms, *Proceedings of the Third International Symposium on Experimental Robotics,* Kyoto, Japan, 1993.

Mozer, M., A focused back-propagation algorithm for temporal pattern recognition, *Complex Systems*, 3, 349, 1989.

Nolfi, S., Using emergent modularity to develop control systems for mobile robots, *Adaptive Behavior*, 5(3-4), 343, 1997.

Nolfi, S. and Tani, J., Extracting regularities in space and time through a cascade of prediction networks: The case of a mobile robot navigating in a structured environment, *Connection Science*, 11(2), 125, 1999.

Omidvar, O. and van der Smagt, P., Eds., *Neural Systems for Robotics*, Academic Press, Boston, 1997.

Peschl, M., The representational relation between environmental structures and neural systems: Autonomy and environmental dependency in neural knowledge representation, *Nonlinear Dynamics, Psychology and Life Sciences*, 1(3), 1996.

Pollack, J. B., Cascaded back-propagation on dynamic connectionist networks, *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 391, 1987.

Pollack, J. B., The induction of dynamical recognizers, *Machine Learning*, 7, 227, 1991.

Pomerleau, D. A., *Neural Network Perception for Mobile Robot Guidance*, Kluwer, Dordrecht, 1993.

Rumelhart, D. E. and McClelland, J., On learning the past tense of English, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models,* Rumelhart, D. E., McClelland, J., and the PDP Group, Eds., MIT Press, Cambridge, 216, 1986.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J., Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations,* Rumelhart, D. E., McClelland, J., and the PDP Group, Eds., MIT Press, Cambridge, 318, 1986.

Rylatt, M. and Czarnecki, C., Beyond physical grounding and naïve time: Investigations into short-term memory for autonomous agents, *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, MIT Press, Cambridge, 1998.

Searle, J., Minds, brains and programs, *Behavioral and Brain Sciences*, 3, 417, 1980.

Sharkey, N. E., Ed., Neural networks for coordination and control: the portability of experiential representations, *Robotics and Autonomous Systems*, 22(3-4), 1997.

Sharkey, N. E., Ed., *Robotics and Autonomous Systems*, 22(3-4). Special issue on *Robot Learning: The New Wave*, 1997.

Sharkey, N. E. and Ziemke, T., A consideration of the biological and psychological foundations of autonomous robotics, *Connection Science*, 10(3-4), 361, 1998.

Siegelmann, H. T., *Neural Networks and Analog Computation: Beyond the Turing Limit*, Birkhäuser, Boston, 1998.

Siegelmann, H. T. and Sontag, E. D., On the computational power of neural nets, *Journal of Computer and System Sciences*, 50(1), 132, 1995.

Steels, L., When are robots intelligent autonomous agents?. *Robotics and Autonomous Systems*, 15, 3, 1995.

Tani, J., Model-based learning for mobile robot navigation from the dynamical systems perspective, *IEEE Transactions on Systems, Man, and Cybernetics*, 26(3), 1996.

Tani, J. and Nolfi, S., Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems, *Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior*, MIT Press, Cambridge, 1998.

Ulbricht, C., Handling time-warped sequences with neural networks, in *From Animals to Animats 4_Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Maes, P., Mataric, M., Meyer, J.-A., Pollack, J., and Wilson, S., Eds., MIT Press, Cambridge, 180, 1996.

von Uexküll, J., *Theoretische Biologie*, Suhrkamp, Frankfurt/Main, 1928.

Varela, F. J., Thompson, E., and Rosch, E., *The Embodied_Mind Cognitive Science and Human Experience*, MIT Press, Cambridge, 1991.

Werbos, P., Backpropagation through time: What it does and how to do it, *Proceedings of the IEEE*, 78(10), 1990.

Wilson, S. W., Knowledge growth in an artificial animal, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum, Hillsdale, NJ, 16, 1985.

Wilson, S. W., The animat path to AI, in *From Animals to Animats_Proceedings of the First International Conference on Simulation of Adaptive Behavior*, Meyer, J.-A. and Wilson, S. W., Eds., MIT Press, Cambridge, 15, 1991.

Ziemke, T., Towards adaptive behaviour system integration using connectionist infinite state automata, in *From Animals to Animats 4_Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Maes, P., Mataric, J.A., Meyer, J.-A., Pollack, J., and Wilson, S., Eds., MIT Press, Cambridge, 15, 1996.

Ziemke, T., Towards autonomous robot control via self-adapting recurrent networks, in *Artificial Neural Networks_ICANN 96*, von der Malsburg, C., von Seelen, W., Vorbrüggen, J. C., and Sendhoff, B., Eds., Springer Verlag, Berlin/Heidelberg, 611, 1996a.

Ziemke, T., Towards adaptive perception in autonomous robots using second-order recurrent networks, *Proceedings of the First Euromicro Workshop on Advanced Mobile Robots,* IEEE Computer Society Press, Los Alamitos, 89, 1996.

Ziemke, T., The ''environmental puppeteer'' revisited: A connectionist perspective on ''autonomy,'' *Proceedings of the Sixth European Workshop on Robot Learning,* Brighton, UK, 100, 1997.

Ziemke, T., Adaptive behavior in autonomous agents, *Presence*, 7(6), 564, 1998.

Ziemke, T., Rethinking grounding, in *Understanding Representation in the Cognitive Sciences. Does Representation Need Reality,* in Riegler, A., Peschl, M., and von Stein, A., Eds., Plenum Press, New York, 1999.

Ziemke, T. and Sharkey, N. E., Eds., *Connection Science*, 10(3-4), Special Issue on *Biorobotics*, 1998.

Ziemke, T. and Sharkey, N. E., Eds., *Autonomous Robots*, 7(1), Special Issue on *Artificial Neural Networks for Robot Learning*, 1999.

Ziemke, T. and Sharkey, N. E., A stroll through the worlds of robots and animals: Applying Jakob von Uexküll's theory of meaning to adaptive robots and artificial life, *Semiotica*, special issue on Jakob von Uexküll, to appear in 2000.