Pro-active agents with recurrent neural controllers

Guido de Croon

March 7, 2004

Preface

This M.Sc. thesis completes my study of Knowledge Engineering at the Universiteit Maastricht, the Netherlands. It represents the completion of the study specialisation of Artificial Intelligence, at the IKAT (Institute for Knowledge and Agent Technology).

The research for my thesis has predominantly taken place at the CNR (Consiglio Nazionale della Ricerca) in Rome. I have spent six months at the CNR, which enabled me to collaborate with my supervisor at the CNR, Dr. S. Nolfi. I would like to thank him for both granting me this unique opportunity and for the guidance during my research. In addition I would like to thank Prof. Dr. D. Parisi, Dr. G. Baldassare, and R. Bianco, M.Sc., for the fruitful discussions on my thesis subject.

This thesis would not have been realised without the help of my supervisors at the IKAT. I would like to thank Prof. Dr. H.J. van den Herik for his in-depth comments on an advanced version of the thesis. In addition, I would like to thank Prof. Dr. E.O. Postma, for gently guiding me in the difficult writing process. Both supervisors have shown me that form and content go hand in hand. Moreover, I would like to thank Dr. I.G. Sprinkhuizen-Kuyper who took the effort to help me improve the mathematical formulations of concepts important to this thesis.

Finally, I would like to thank my parents, brother, girl-friend, and friends for their everlasting support and for their patience when I enthousiastically ramble on about artificial intelligence.

Guido de Croon
Maastricht, December 2003

Abstract

Embodied cognitive science is regarded as a bottom-up approach within artificial intelligence. It envisages the realisation and study of increasingly complex artificial agents. As a consequence, the research emphasised reactive agents, i.e., agents that always respond in the same way to the same sensory inputs. To reach higher agent capabilities, the research focus currently shifts from reactive to pro-active agents. The actions of pro-active agents do not only depend on the inputs, but also on the "internal state". The internal state of an agent is defined as the subset of variables that co-determine the future input-output mapping of the agent.

Typically, a pro-active agent has a neural controller. In this thesis, we investigate three mechanisms that realise an internal state in a neural controller: recurrency, neural inertia, and varying time delays. Since the internal state determines the agent's behaviour in the environment, it influences the agent capabilities. At the moment it is a challenging question how the different mechanisms that realise an internal state determine the capabilities of pro-active agents. This leads us to the following problem statement.

How do the mechanisms that realise an internal state influence the agent capabilities?

To structure our investigations, we formulate two research questions:

1. What are the capabilities of agents with different types of recurrent neural controllers?

2. How are these capabilities related to the mechanisms realising an internal state?

To answer the research questions, the following five types of recurrent neural controllers are investigated: the standard recurrent neural network, the recurrent neural network with a delta, the continuous-time recurrent neural network, the nonlinear autoregressive model with exogeneous inputs, and the time delay recurrent neural network. We employ the different types of recurrent neural networks as controllers of agents that have to perform robotic tasks. Following the methodology of evolutionary robotics, we performed experiments on the following three tasks: (1) the nest-finding task, (2) the driving task, and (3) the self-localisation task. All three tasks require an internal state to be solved.

The results of the experiments demonstrate that the mechanisms realising an internal state are tightly related to the capabilities of a pro-active agent.

In our discussion we answer the first research question by defining two types of agents with corresponding capabilities: (1) variable time-scale agents, and (2) single time-scale agents. Variable time-scale agents are agents that can determine when sensory inputs affect the outputs. Hence, they can exploit regularities in the environment on short and long time scales, while single time-scale agents cannot. As a result, single time-scale agents that operate on a short time scale are more dependent on sensory-motor coordination than variable time-scale agents.

To answer the second research question we observe that the mechanisms realising an internal state determine the agent capabilities by establishing whether the agent is a variable time-scale agent or a single time-scale agent. We conclude that the use of two mechanisms, viz. that of neural inertia or that of varying time delays, result in variable time-scale agents. The reason is that they have parameters that determine the time scales on which sensory signals affect the outputs of the controller. Agents controlled by the recurrent neural network with a delta, continuous-time recurrent neural network or time delay recurrent neural network are variable time-scale agents. We also conclude that the mechanism of recurrency is not a sufficient condition to lead to variable time-scale agents. Hence, the agents controlled by a standard recurrent neural network or a nonlinear autoregressive model with exogeneous inputs are single time-scale agents. Since the activations in their internal state always change on short time scales, they cannot determine when sensory inputs affect the outputs. This implies that they cannot exploit regularities in the environment on short and long time scales.

Contents

Chapter 1

Introduction

Embodied cognitive science represents a bottom-up approach to artificial intelligence that emphasises the importance of the interaction between an agent and its environment [28, 7, 15]. Many investigations focussed on how reactive agents use their interaction with the environment to solve problems [22, 27].

1.1 Reactive and pro-active agents

Reactive agents have been called "minimally cognitive agents" [4], since they always respond in the same way to the same sensory inputs [22, 33, 35]. Several studies have deepened our understanding of the capabilities and limitations of reactive agents [27, 32].

To study a slightly higher level of cognition, there is a shift in focus from reactive towards pro-active agents. The actions of pro-active agents do not solely depend on the sensory inputs, but also on their "internal state" [4, 26, 2]. The internal state of an agent is defined as the subset of variables that co-determine the future input-output mapping of the agent. Typically, an internal state is realised by a neural controller that translates instantaneous and past sensory inputs into actions [23, 33, 4, 12, 34]. We investigate three mechanisms that realise an internal state in a neural controller; recurrency [10], neural inertia [3, 23] and varying time delays [17, 9].

It is evident that the capabilities of a pro-active agent depend on the properties of its internal state, since the internal state co-determines the agent's behaviour in the environment. Hence, the capabilities of a pro-active agent also depend on the mechanisms realising the internal state.

1.2 Problem statement

At present it is a challenging question how the different mechanisms that realise an internal state determine the capabilities of pro-active agents. This leads us to the following problem statement.

How do the mechanisms that realise an internal state influence the agent capabilities?

1.3 Research questions

To structure our investigations, our study focuses on two research questions:

1. What are the capabilities of agents with different types of recurrent neural controllers?

2. How are these capabilities related to the mechanisms realising an internal state?

To answer the research questions, the following five types of recurrent neural controllers are investigated: the standard recurrent neural network, the recurrent neural network with a delta, the continuous-time recurrent neural network, the nonlinear autoregressive model with exogeneous inputs, and the time delay recurrent neural network.

We employ the different types of recurrent neural networks as controllers of agents that have to perform robotic tasks. Following the methodology of evolutionary robotics, we performed experiments on the following three tasks: (1) the nest-finding task, (2) the driving task, and (3) the self-localisation task. All three tasks require an internal state to be solved. The tasks demonstrate different capabilities of agents controlled by the five recurrent neural controllers. We explain the relation between the agent capabilities and the mechanisms of recurrency, neural inertia and varying time delays.

1.4 Thesis outline

The outline of the remainder of the thesis is as follows. In chapter 2 we explain the methodology of evolutionary robotics. Chapter 3 describes the general experimental procedure and the five types of recurrent neural networks. Then in the chapters 4, 5, and 6, the experimental tasks and results are presented for the nest-finding task, the driving task and the self-localisation task, respectively. Chapter 7 contains a general discussion of the results, in which we answer the research questions. Finally, in chapter 8 we present the conclusions of the research.

Chapter 2

Methodology

An agent performing a task has to choose the right actions based on the sensory information it receives about the environment. A prevailing question is: how does the agent learn to choose the right actions? A common method is to evolve the agent's controller according to a Darwinistic evolution process favouring agents performing well on the task over agents that do not perform well. The scientific method of evolving agent controllers is called evolutionary robotics [24, 19, 13]. In this thesis, the methodology of evolutionary robotics is used to optimise controllers.

2.1 Evolutionary algorithm

An "Evolutionary Algorithm" [24] optimises neural controllers by searching the space of possible controllers for one that results in a good performance. The evolutionary process starts by creating a population of controllers, for example a group of randomly-initialised neural networks. All controllers in the population are applied to the task. To determine their performance, a fitness function is defined that assigns a fitness to every controller. After the fitness has been determined for all controllers in the population, the fittest controllers have a higher chance to be selected to generate a new population. The new generation of controllers is created by changing the selected controllers. The evolutionary algorithm then repeats the process and generates new generations until some stopping criterion is met.

2.2 Three approaches to evolutionary robotics

Embodied cognitive science stresses the importance of the embodiment of the agent. That is why in embodied cognitive science the use of real robots is preferred for evolutionary experiments. Evolutionary optimisation with real robots is called the physical approach [25]. A disadvantage of the physical approach is that it takes a large amount of time even to evolve a small number of generations. There are two alternatives. One is to employ the simulated approach [25], in which the evolution is performed in a simulator of the environment and

the agent. A second alternative is the hybrid approach [25], in which agents are first evolved in a simulator. Then the evolution is continued on real robots.

2.3 Simulation by Evorobot

In this thesis we employ the simulated approach by using Evorobot [20], that enables realistic simulations. It uses samples taken from a Kephera robot [29] in a real environment to simulate the sensory readings and motor effects. The transition from an agent that has been developed with the simulator to an agent implemented on a real Kephera robot takes only a few generations, as demonstrated in [25]. The simulator has shown to be satisfactory in a number of studies, such as [27, 21].

Chapter 3

Experimental setup

We investigate the capabilities of pro-active agents controlled by recurrent neural networks. The capabilities of pro-active agents depend on their internal state. For recurrent neural networks there are many mechanisms that realise internal states. As stated in chapter 1, we study three mechanisms: recurrency, neural inertia and varying time delays. We select five types of recurrent networks that all employ one or more of these mechanisms. In the experiments to be discussed in chapters 4, 5 and 6, we study the relation between the mechanisms and the capabilities of the associated agents.

In this chapter, we discuss the setup for those experiments. In section 3.1 we give background information on the concept of internal state. In section 3.2 we introduce the three mechanisms and the corresponding recurrent neural networks. In sections 3.2.1 to 3.2.3 we discuss the mechanisms with the corresponding recurrent neural networks that apply them. The first mechanism we discuss is recurrency that has as experimental platform the standard recurrent neural network (RNN). The second mechanism discussed is neural inertia that has as experimental platform the recurrent neural network with a delta (RNND) and the continuous-time recurrent neural network (CTRNN). The third mechanism we discuss is varying time delays that has as experimental platform the nonlinear autoregressive model with exogeneous inputs (NARX) and the time-delay recurrent neural network (TDRNN). In section 3.3 we discuss the three robotic tasks that are used for the experiments.

3.1 Background

In this section, we give an explanation on the concept of internal state. In general, we analyse the internal state in the context of neural networks. As a start, we distinguish two types of neural networks: feedforward neural networks and recurrent neural networks.

In figure 3.1, a simple feedforward neural network is drawn. A circle represents a neuron and an arrow represents a connection between two neurons. The network has three layers: the input layer, the hidden layer, and the output layer. The output neuron, labelled ‘o’, has as input a bias, labelled ‘bias’, and the hidden neurons, labelled ‘h’. The hidden neurons have as input a bias and

the input neurons, labelled 'i'. The activation of a bias is a constant, while the activation of an input neuron is a variable that is determined externally.

A feedforward neural network represents a function from the activations of the input neurons to the activations of the output neurons. This means that one specific input vector $\mathbf{i}$ is always associated with one output vector $\mathbf{o}$, in a way that depends on the weights of the connections. This is expressed in equation 3.1, with $f_{\mathbf{w}}$ representing a function parameterised by the weights $\mathbf{w}$.

$$\mathbf{o}(t) = f_{\mathbf{w}}(\mathbf{i}(t)), \tag{3.1}$$

where the argument t represents the time at which the input and output are determined. Time is assumed to be discrete. For the network shown in figure 3.1, $\mathbf{i} = (a_{i1}, a_{i2})$ and $\mathbf{o} = (a_{o1})$, where a represents the activation of an individual neuron.

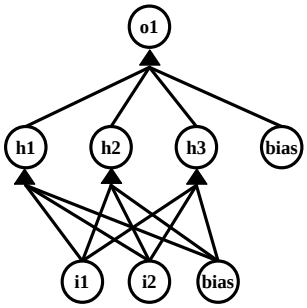

Figure 3.1: A feedforward neural network

In a recurrent neural network at least one past activation serves as neural input. The neural network shown in figure 3.2 is an example of a recurrent neural network. The connection from the grey node to itself represents a feedback of the activation value of the neuron to its neural input. Assuming that the feedback connection has a delay of a single time step, the neuron receives its activation of the last time step as an additional input. An alternative representation is displayed in figure 3.3. A neural network with one or more recurrent connections can use information from the past. By changing the weights of the network, the importance of past information and the way in which it is used can vary.

All values of the network that are adjusted by an evolutionary algorithm, constitute the set of parameters of the network. In our experiments, the weights are part of the parameters. The set of variables of the network consists of the activations of all neurons. In a recurrent neural network a subset of these variables is used to determine future outputs. Hence, the outputs $\mathbf{o}$ of a recurrent neural network are determined by the input vector $\mathbf{i}$, a function parameterised by the weights $f_{\mathbf{w}}$ and the vector $\mathbf{s}$ containing these variables.

$$\mathbf{o}(t) = f_{\mathbf{w}}(\mathbf{i}(t), \mathbf{s}(t)), \tag{3.2}$$

where $\mathbf{s}(t)$ is determined as follows.

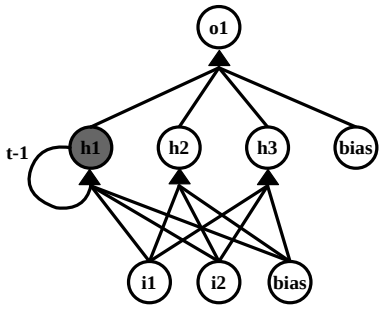

Figure 3.2: A recurrent neural network

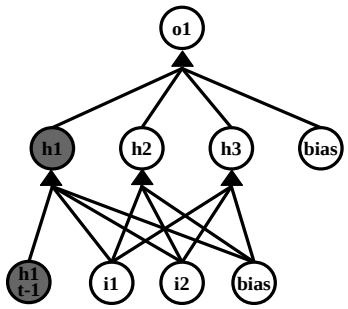

Figure 3.3: A recurrent neural network (alternative representation)

$$\mathbf{s}(t) = g_{\mathbf{w}}(\mathbf{i}(0), \mathbf{i}(1), \ldots, \mathbf{i}(t-1), \mathbf{s}(0)) \tag{3.3}$$

Equation (3.3) clarifies that by means of $\mathbf{s}(t)$ the outputs of an agent depend on the input history and $\mathbf{s}(0)$. The subset of variables whose values constitute $\mathbf{s}(t)$ is called the internal state of an agent, $S(t)$. Hence, we define the internal state as follows.

Definition 3.1 *The internal state of an agent is the subset of variables that co-determine the future input-output mapping of the agent*

Definition 3.1 agrees with the concept of internal state in other studies [4, 26, 35]. The definition implies that feedforward neural networks with fixed weights do not have an internal state. The recurrent neural network shown in figure 3.2 does have an internal state which corresponds to the activation of the shaded hidden neuron $h1$, $S(t) = \{a_{h1}(t-1)\}$. Its activation at time step $t-1$ determines the activation of the output neuron at t together with the input vector at t. If the inputs remain the same and the activation of $h1$ has changed, the activation of the output neuron can change as well. In other words, the shaded hidden neuron $h1$ determines the mapping from inputs to outputs together with $f_{\mathbf{w}}$ (3.2).

3.2 Three mechanisms to realise an internal state

In this section, we introduce the three mechanisms and the corresponding recurrent neural networks. We focus on three mechanisms that all establish an internal state consisting of neural activations:

- Recurrency
- Neural inertia
- Varying time delays

We define these mechanisms as follows.

Recurrency: We define the mechanism of recurrency as follows.

Definition 3.2 *A neural network employs the mechanism of recurrency if at least one neural activation serves as input to another neuron in a future time step.*

Neural inertia: We define the mechanism of neural inertia as follows.

Definition 3.3 *A neural network employs the mechanism of neural inertia, if at least one neuron's activation does not only depend on its neural connections and external input but also directly on its past activations.*

The function that determines the activations $\mathbf{a}$ of a network with neural inertia is not only parameterised by the weights, but also by a matrix $\mathbf{D}$ containing the values corresponding to the neural inertia of individual neurons.

$$\mathbf{a}(t) = f_{\mathbf{w},\mathbf{D}}(\mathbf{i}(t), \mathbf{s}(t)) \tag{3.4}$$

In the mechanism of neural inertia, the activation of a neuron of the last time step influences the current activation in a more direct manner than through a recurrent connection. To clarify this, we show function 3.5 that determines the activations of an RNND.

$$\mathbf{a}(t) = \mathbf{D}(\mathbf{a}(t-1)) + (\mathbf{I} - \mathbf{D}) f_{\mathbf{w}}(\mathbf{i}(t), \mathbf{s}(t)), \tag{3.5}$$

where $\mathbf{D}$ is a diagonal matrix whose diagonal elements all have values in the interval $[0, 1]$. These elements represent the proportion of neural inertia of the individual neurons. $\mathbf{s}(t)$ is an argument of $f_{\mathbf{w}}$, since the RNND applies both recurrent connections and neural inertia. In section 3.2.2 we discuss the activation function of the RNND in further detail.

Varying time delays: We define the mechanism of varying time delays as follows.

Definition 3.4 *A neural network employs the mechanism of varying time delays if the internal state contains variables that are delayed more than one time step.*

$$a_i(t-d) \in S(t), \quad d > 1 \tag{3.6}$$

As mentioned, the recurrent neural network in figure 3.2 has an internal state that consists of the activation of the shaded hidden neuron, $h1$, of last time step, $S(t) = \{a_{h1}(t-1)\}$. If the recurrent connection of the recurrent neural network shown in figure 3.2 concerns two time steps,

then the internal state contains the activation $h1$ of the last two time steps, as expressed in equation 3.7.

$$S(t) = \{a_{h1}(t-1), a_{h1}(t-2)\} \tag{3.7}$$

The activation $a_{h1}(t-2)$ is part of the internal state, since it influences the input-output mapping at time step t. Although the activation $a_{h1}(t-1)$ does not influence the outputs at time step t, it is part of $S(t)$. The reason for this is that it is 'remembered' at time step t to influence the input-output mapping at time step $t+1$.

We study five recurrent neural networks that incorporate one or more of these mechanisms. Table 3.1 indicates with the label 'X' what mechanisms are applied by the five recurrent neural networks. In sections 3.2.1 to 3.2.3 we discuss the three mechanisms and the associated recurrent neural networks shown in table 3.1 in further detail. In section 3.2.1 we discuss the mechanism of recurrency, which is incorporated in all selected neural controllers (see table 3.1). The mechanism of recurrency has as particular experimental platform the standard recurrent neural network (RNN), since the RNN applies no other mechanism than that of recurrency. In section 3.2.2 we discuss the mechanism of neural inertia, which has as experimental platform the recurrent neural network with a delta (RNND) and the continuous-time recurrent neural network (CTRNN). Finally, in section 3.2.3 we discuss the mechanism of varying time delays, which has as experimental platform the nonlinear autoregressive model with exogeneous inputs (NARX) and the time-delay recurrent neural network (TDRNN). We explain for all mechanisms what variables are part of the internal state and how the internal state influences future input-output mappings.

Neural network	Recurrency	Neural inertia	Varying time delays
RNN	X	-	-
RNND	X	X	-
CTRNN	X	X	-
NARX	X	-	X
TDRNN	X	-	X

Table 3.1: Mechanisms employed by the neural networks

3.2.1 Recurrency

The first mechanism to be discussed is recurrency. Since recurrency is typically used in the controller of a pro-active agent, all of the selected neural networks are recurrent neural networks. The simplest recurrent neural network is called an Elman network [10]. It is considered a standard recurrent neural network and we will refer to it as an RNN. What follows below is relevant for all five recurrent neural networks.

Standard Recurrent Neural Network: RNN

The RNN is the only network that solely applies the mechanism of recurrency. In this section, we first discuss the structure of the RNN and then we demonstrate that the RNN applies the mechanism of recurrency.

The structure of the RNN is shown in figure 3.4. A box represents a collection of neurons. An arrow from one box to another means that all neurons of the boxes are connected. The hidden and output layers include a bias neuron.

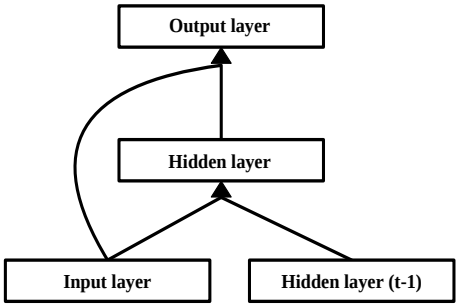

Figure 3.4: A standard recurrent neural network (RNN)

Elman [10] introduced the RNN. In the RNN, the activations of the hidden neurons of the last time step serve as neural input to the hidden layer. The recurrency is indicated in figure 3.4 by the box labelled "Hidden layer (t-1)". In the RNN, the internal state consists of the activations of the hidden neurons. These activations constitute the elements of vector $\mathbf{s}$ in equation 3.2 and are the variables that influence future input-output mappings.

To explain that the RNN complies with the definition of a network that uses recurrency, we show the activation functions of the individual neurons. These activations determine together with the weights the function $f_{\mathbf{w}}$ in equation 3.2. In a standard recurrent neural network, the activation functions for hidden and output neurons are:

$$a_i(t) = \sigma(netinput_i(t) + bias_i) \tag{3.8}$$

$$netinput_i(t) = \sum_{j=1}^{N} w_{ji} a_j(t), \tag{3.9}$$

in which $a_i(t)$ denotes the activation of neuron i at time t and σ is the logistic function, $\sigma(x) = \frac{1}{1+e^{-x}}$. N is the number of neurons connected to neuron i. In the case of the hidden neurons this is the number of neurons in the input layer plus the number of neurons in the hidden layer. w_{ji} is the weight of the connection from node j to node i.

The RNN applies the mechanism of recurrency, since the neurons in the hidden layer have all activations of the hidden layer of the last time step as an input. Therefore, there is at least one $a_j(t)$ in equation 3.9 that is equal to $a_h(t-1)$, where a_h represents an activation of a hidden neuron.

The activation of the input neurons is determined externally by the sensors. Hence, their outputs are defined as:

$$a_i(t) = in_i(t) \tag{3.10}$$

Dynamics of the RNN

The dynamics of an RNN are defined as the changes of the neural activations. Each type of recurrent neural network has its associated dynamics. It is important to understand how the dynamics of the variables in internal state work and how they depend on parameters and inputs. Understanding a network's dynamics will shed light on how a specific recurrent neural network can change its input-output mapping. In subsection 3.2.2 the dynamics will be discussed in further detail.

The possible dynamics of a type of recurrent neural network determine its capacities as a controller. The discussion of the other four recurrent neural networks in the remainder of the chapter will clarify that the RNN is a special case of those networks. Therefore, the RNN is only able to represent a subset of the dynamical systems that the other networks can realise.

3.2.2 Neural inertia

The second mechanism to be discussed is neural inertia. In nature, the activation of a neuron is influenced by past activations. For example, after a neuron has fired, there is always a refractory period during which the neuron is unable to fire [8]. A neuron has inertia, if the activation of a neuron does not only depend on its neural connections and external input but also on its past activations. Below, we discuss two recurrent neural networks with neural inertia: the recurrent neural network with a delta (RNND) [23] and the continuous-time recurrent neural network (CTRNN) [3].

Recurrent Neural Network with Delta: RNND

The Recurrent Neural Network with a Delta-value (RNND) is the first network that possesses a neural inertia mechanism. It was proposed by Nolfi [23]. In this section, we first discuss the structure of the network and then we demonstrate that the RNND applies the mechanism of neural inertia.

The RNND has the same structure as the standard recurrent neural network shown in figure 3.4. However, the neural activations are calculated in a different way. The activation function of the RNND will clarify that the neural activations of the RNND do not only depend on its neural connections and external input but also directly on its past activation, i.e., that the RNND employs the mechanism of neural inertia. Every hidden and input neuron has its own 'delta'-value d_i and the following activation function.

$$a_i(t+1) = d_i a_i(t) + (1 - d_i)\sigma\left(netinput_i(t+1) + bias_i + in_i(t+1)\right), \tag{3.11}$$

where $netinput_i(t+1)$ is defined as in equation 3.9. From $d_i \in [0,1]$ follows $a_i \in [0,1]$ (see [23]). The value d_i determines to what extent the new activation is determined by the old one, i.e., it is proportional to the degree of neural inertia. All other variables are defined as for the RNN. in_i is an external input to the neuron, for example a value provided by the sensors.

The RNND applies the mechanism of neural inertia, since the activation $a_i(t+1)$ does not only depend on $netinput_i(t+1)$, $bias_i$ and $in_i(t+1)$, but also on the last activation, $a_i(t)$. If the $netinput_i$ and in_i are fixed, the activation of the neuron can still change by means of the neural inertia. If the activation function is not expressed for individual neurons, but for all neurons in the network, then the activation function is of the form of equation 3.5.

The internal state of the RNND consists of the activations of the hidden neurons and all activations of the neurons that apply activation function 3.11. The activations influence future input-output mappings in two ways: through recurrency and through neural inertia.

Continuous-Time Recurrent Neural Network: CTRNN

The continuous-time recurrent neural network is the second network that incorporates the mechanism of neural inertia. Also for the CTRNN, we first discuss the structure of the network and then we demonstrate that the CTRNN applies the mechanism of neural inertia.

The CTRNN has the same network structure as the RNN, but it has other activation functions than the RNN. The activation functions of the CTRNN will clarify that the neural activations of the CTRNN do not only depend on its neural connections and external input but also directly on its past activations.

As its name indicates, the Continuous Time Recurrent Neural Network (CTRNN) operates in continuous time. Beer [3] defined the activation function of neurons in a CTRNN as follows.

$$a_i(t) = \sigma(p_i(t) + bias) \tag{3.12}$$

$$\frac{dp_i(t)}{dt} = \frac{1}{tc_i}(-p_i(t) + netinput_i(t) + in_i(t)) \tag{3.13}$$

$$netinput_i(t) = \sum_{j=1}^{N} w_{ji} a_j(t) \tag{3.14}$$

The activation $a_i(t)$ of neuron i at time t depends on variable $p_i(t)$, the activation potential of that neuron. Differential equation 3.13 defines the change in the activation potential of neuron i at t. tc_i is the time constant of neuron i ($tc_i > 1$, [6]). If $tc_i \in \Re$, then $\frac{1}{tc_i} \in [0,1]$.

A continuous-time recurrent neural network is simulated on a computer by a discrete approximation. For the purpose of obtaining a discrete activation function that clarifies that the CTRNN employs the mechanism of neural inertia, we transform equation 3.12 into 3.17. If we assume $dt = 1$, equation 3.13 becomes:

$$p_i(t+1) - p_i(t) = \frac{1}{tc_i}(-p_i(t) + netinput_i(t) + in_i(t)) \tag{3.15}$$

With a few algebraic operations we transform equation 3.15 in equation 3.16.

$$p_i(t+1) = (1 - \frac{1}{tc_i})pi(t) + \frac{1}{tc_i}(netinput_i(t) + in_i(t)) \tag{3.16}$$

When substituting $t+1$ for t in equation 3.12 and then inserting the right hand side of equation 3.16 for $p_i(t+1)$, equation 3.12 becomes:

$$a_i(t+1) = \sigma\left(\left((1 - \frac{1}{tc_i})p_i(t) + \frac{1}{tc_i}(netinput_i(t) + in_i(t))\right) + bias_i\right) \tag{3.17}$$

The discretisation of the activation function makes clear that to determine the activation of a hidden neuron for time $t+1$, the activations of the input layer of time step t are used ($netinput_i(t)$). So at every layer an additional delay of 1 time step is introduced. One way to deal with the delay is to update the activations of the network with a higher frequency than the sensing frequency of the agent. However, we have chosen to use the neural inputs at time step $t+1$. This implies that the inputs have an immediate effect on the outputs and an agent can immediately react to signals from the environment. In our experiments we use the following activation function to update the neurons with neural inertia.

$$a_i(t+1) = \sigma\left(\left((1 - \frac{1}{tc_i})p_i(t) + \frac{1}{tc_i}(netinput_i(t+1) + in_i(t+1))\right) + bias_i\right) \tag{3.18}$$

We demonstrate that the CTRNN complies with the definition of a neural network that employs the mechanism of neural inertia (definition 3.3). Equation 3.18 illustrates that the activation $a_i(t+1)$ does not only depend on $netinput_i(t+1)$, $bias_i$, and $in_i(t+1)$, but also on the activation potential of last time step, $p_i(t)$. By means of the activation potential that is determined according to equation 3.16, the neural activation depends on the neural activations of past time steps.

We end our demonstration that the CTRNN employs the mechanism of neural inertia, by showing that the function that determines the activations $\mathbf{a}$ of the CTRNN is not only parameterised by the weights, but also by a matrix $\mathbf{D}$ containing the values corresponding to the neural inertia of individual neurons (see equation 3.4). If the activation function is expressed for the vector of all neural activations $\mathbf{a}$, the activation function is as follows.

$$\mathbf{a}(t) = \sigma(\mathbf{p}(t) + \mathbf{bias}) \tag{3.19}$$

$$\mathbf{p}(t) = \mathbf{D}(\mathbf{p}(t-1)) + (\mathbf{I} - \mathbf{D})f_{\mathbf{w}}(\mathbf{i}(t), \mathbf{s}(t)), \tag{3.20}$$

where **D** is a diagonal matrix whose diagonal elements all have values in the interval $[0, 1]$. These elements represent the proportion of neural inertia of the individual neurons, $1 - \frac{1}{tc_i}$. The time constant tc_i determines the way in which the activation potential p_i changes as a result of the past activation potential and new neural inputs. Therefore, the value of the time constant is proportional to the degree of neural inertia.

To resume, the internal state of a CTRNN consists of the activations of the hidden units and the activation potentials of the neurons that apply activation function 3.12. The internal state influences the future input-output mapping by both the neural inertia and the recurrent activations of the hidden layer.

Dynamics of RNND and CTRNN

What are the consequences of the neural inertia for the capabilities of an agent? As stated before, these capabilities depend on the dynamical systems an agent is able to represent. We first discuss the dynamics of the CTRNN (since it is well documented in the literature) to illustrate the effect of neural inertia on these dynamics. Thereafter, we demonstrate that the RNND is very similar in its dynamics to a CTRNN.

What type of dynamics can CTRNNs accomplish and what influence do the parameters, and in particular the time constants, have on their dynamics? In [3], Beer gives a qualitative description of the dynamics of small CTRNNs, relying on dynamical systems theory. He investigates how the parameters affect the equilibrium points in a dynamical system and the dynamics associated with them. An equilibrium point is a point in which the neural activity does not change. He demonstrates that changing the weights or inputs of a CTRNN can change the number and place of the equilibrium points in the dynamical system of neural activations.

The effect of neural inertia on the dynamics of a CTRNN depends on the time constant, since it is proportional to the degree of neural inertia. Beer demonstrates that changing the time constants does not change the number of equilibriums. However, varying the ratio of the different time constants can change the stability and type of equilibriums exhibited by the dynamical system. This increases the types of dynamical systems that can be represented. To clarify this, let us take an example where $p_1(t_0) = 3.5$ and the equilibrium point the dynamical system evolves to is at $p_1 = 4.5$. If the time constant is small, then the trajectory towards 4.5 might take for example 2 time steps. However, a very large time constant might result in a long duration, e.g., 25 time steps.

The dynamics of our implementations of the RNND and CTRNN differ in their behaviour around equilibrium points only. The reason for this is that the equilibrium points of the RNND and CTRNN do not depend on the neural inertia parameters d_i and tc_i, respectively. Beer shows this in [3] for the CTRNN. We demonstrate the independence of the equilibrium points of d_i in appendix A.

Since in the RNND the value d_i does not affect the number of equilibrium points, we can assume $d_i = 0$ to find the equilibrium points. Assuming $d_i = 0$

makes the activation function of the RNND the same as for the RNN. If $d_i = 0$:

$$\begin{aligned} a_i(t+1) &= d_i a_i(t) + (1-d_i)\sigma(netinput_i(t+1) + bias_i + in_i(t+1)) \\ &= \sigma(netinput_i(t+1) + bias_i + in_i(t+1)) \end{aligned} \tag{3.21}$$

Equation 3.21 is equivalent to the activation function of the RNN (3.8, 3.10), provided that the inputs of the RNN are defined as the logistic of the inputs given to the RNND. By setting all d_i to 0, an RNND is equal to a standard recurrent neural network with the same structure. If the deltas in the network have other values, the stability and type of the equilibriums will be different for the same weights and inputs.

If $tc_i = 1$, our implementation of the activation function of the CTRNN becomes:

$$\begin{aligned} a_i(t+1) &= \sigma\left((1-\frac{1}{tc_i})p_i(t) + \frac{1}{tc_i}(netinput_i(t+1) + in_i(t+1)) + bias_i\right) \\ &= \sigma(netinput_i(t+1) + bias_i + in_i(t+1)) \end{aligned} \tag{3.22}$$

By setting all time constants to 1 the CTRNN also equals the RNN described in section 3.2.1. Therefore, the places of the equilibrium points of the activations for our implementations of the CTRNN and the RNND are the same as for the RNN. The RNN is a special case of both the RNND as the CTRNN, so it can only represent a subset of their possible dynamical systems.

The effect of neural inertia on the capabilities of the agent cannot be predicted with this analysis of the network dynamics. However, the RNND and CTRNN only differ in their behaviour around equilibrium points. Since both networks incorporate a parameter that sets the degree of neural inertia, they are likely to share similar dynamics and capabilities.

3.2.3 Varying time delays

Every recurrent neural connection delays an activation for a number of time steps until it is presented as neural input. The delays on the recurrent connections of the networks that have been discussed in section 3.2.1 and 3.2.2 are all one time step. In this section we discuss the mechanism that varies the duration of delays. The NARX-network has more recurrent connections than an RNN and has recurrent connections that have delays of varying duration. The TDRNN has an adaptable delay for every neural connection in its network. Both networks are discussed below.

Nonlinear autoregressive model with exogeneous inputs: NARX

The first recurrent neural network that employs the mechanism of varying delays on the connections that we will discuss, is the Nonlinear AutoRegressive model with eXogeneous inputs (NARX-network). In this section, we first discuss the structure of the NARX-network and then we demonstrate that the NARX-network applies the mechanism of varying time delays. Thereafter, we discuss the reason of the introduction of the varying time delays. Finally, we explain what variables constitute the internal state in a NARX-network and how they influence future input-output mappings.

The structure of the NARX-network is displayed in figure 3.5. The figure illustrates that it has an input-memory and an output-memory. Both memories have a certain 'capacity', c_{in} and c_{out}, respectively. If the output memory has a capacity of 3, then the activations of all output neurons of t-1, t-2 and t-3 serve as input for the hidden neurons. The NARX-network employs the same activation function as the RNN.

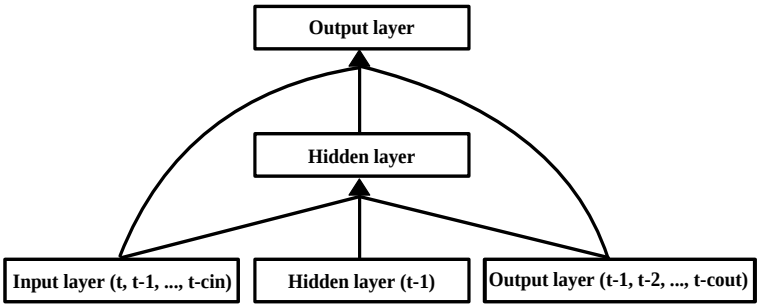

Figure 3.5: A NARX-network

We demonstrate that the NARX-network applies the mechanism of varying time delays, by showing that the internal state contains activations that are delayed more than one time step. The internal state of a NARX-network can be expressed as:

$$S(t) = \{\mathbf{i}(t), \mathbf{i}(t-1), ..., \mathbf{i}(t-c_{in}), \mathbf{o}(t-1), \mathbf{o}(t-2), ..., \mathbf{o}(t-c_{out}), \mathbf{h}(t-1)\}, \quad (3.23)$$

where $\mathbf{i}$ is the vector of activations of the input neurons, $\mathbf{o}$ is the vector of activations of the output neurons, and $\mathbf{h}$ is the vector of activations of the hidden neurons. An activation vector that is delayed more than one time step contains at least one activation variable $a_i(t-d)$ with $d > 1$. Therefore, equation 3.23 illustrates that the NARX-network applies the mechanism of varying time delays, if either $c_{in} > 1$ or $c_{out} > 1$, and the relevant weights are not equal to zero.

The reason that the NARX-network employs varying time delays is that it renders the network suitable for learning long-term dependencies, as shown in [17]. A task displays long-term dependencies when the output at time t_2 depends on an input received long before, at time $t_1 << t_2$. Learning techniques such as back-propagation for standard recurrent neural networks fail on

learning long-term dependencies, [5]. For training methods based on an error gradient, long-term dependencies constitute a problem because the error gradient vanishes over time. NARX-networks do not suffer from this limitation.

An explanation of the improved learning of long-term dependencies in NARX-networks is given in [17]. According to the explanation, the connections in the ouput memory with a longer delay make it possible to use information far back in time. These connections might turn out to be useful as well for pro-active agents that do not adapt their weights during lifetime but throughout evolution.

To resume, the internal state of a NARX-network consists of the activations of the hidden layer and activations of multiple time steps of all input and output neurons. All these activations influence future input-output mappings by serving as neural inputs to the hidden and output layers. The NARX-network incorporates two mechanisms to realise an internal state: recurrent neural connections and delays of multiple time steps on neural connections.

Time Delay Recurrent Neural Network: TDRNN

The second recurrent neural network that we discuss is the time-delay recurrent neural network. In this section, we first discuss the background and structure of the TDRNN. Then we demonstrate that the TDRNN employs the mechanism of varying time delays. Finally, we introduce a restricted version of the TDRNN which is used in our experiments.

The TDRNN has a biological background. Most recurrent neural networks ignore the conduction time of neural signals. Since biological neurons can differ a lot physically, communication time between neurons can also differ greatly. It is generally acknowledged that time delays play a role in neural computation, [30]. Introducing adjustable time delays for every neural connection enhances the capacities of a recurrent neural network. Figure 3.6 shows the structure of a time-delay recurrent neural network (TDRNN). The structure is the same as that of a standard recurrent neural network. However, in the TDRNN all connections have a separate delay. The delays are extra parameters that are adjusted by an evolutionary algorithm. The values of the delays range from 0 to a fixed maximum of time steps. The activation function is the same as that of the RNN.

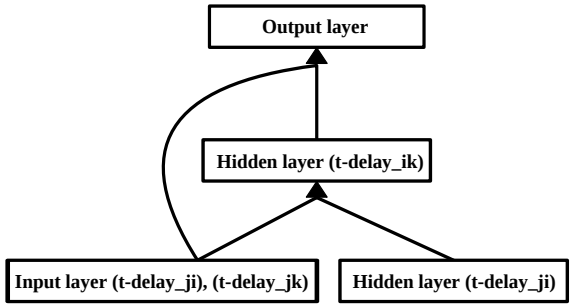

Figure 3.6: A time-delay recurrent neural network

The TDRNN employs the mechanism of varying time delays, if there is a connection from a neuron j to a neuron i, $w_{ji} \neq 0$, and $delay_{ji} > 1$. Then, all past activations of neuron j until $delay_{ji}$ have to be remembered to determine future network activations, as expressed in equation 3.24.

$$\{a_j(t-1), a_j(t-2), ..., a_j(t-delay_{ji})\} \subseteq S(t) \tag{3.24}$$

Equation 3.24 demonstrates that there is at least one $a_i(t-d) \in S(t)$ with $d > 1$. The above mentioned activations of neuron j are conducted along the connection between j and i. The internal state of a TDRNN consists of all instantaneous and stored activations that are conducted along the connections between the neurons. The internal state affects future input-output mappings, because after the delay the stored activations serve as neural inputs.

A TDRNN of moderate size can already have many neural connections and hence many delay parameters. Since a large number of parameters implies a large search space for the evolutionary algorithm, we use a restricted form of the TDRNN for our experiments in which the number of parameters is restricted by using a single delay for all the inputs of a neuron. The restriction also simplifies the analysis of evolved agents.

Dynamics of NARX and TDRNN

The NARX-network and the TDRNN are able to represent more complex dynamical systems than an RNN. The RNN is a special case of both networks. For a NARX-network, setting all weights coming from the input- and output-memory to 0 turns it into an RNN. If we set all time delays of the connections between neurons of the hidden layer of a TDRNN to 1 and the rest of the time delays to 0, the TDRNN equals an RNN as well.

3.3 The robotic tasks

To determine what properties of recurrent neural controllers are important for an agent's capabilities, we apply the five types of recurrent neural networks discussed in 3.2 to three different robotic tasks. The three tasks require an internal state to be solved. In addition, they are set up as to reveal the different capabilities of pro-active agents. For each task we relate the results achieved by the various agents to the three mechanisms underlying an internal state.

Chapter 4 discusses the 'Nest finding task' in which the recurrent neural controllers must realise a simple memory to succeed. Chapter 5 deals with the 'Driving task'. In that task it is necessary for the recurrent neural networks to generate adequate internal dynamics for driving. So the focus is on the internal dynamics that can be generated by the five recurrent neural controllers. In chapter 6 we explain the 'Self-localisation task', in which the recurrent neural controllers have to use their internal dynamics in accordance with sensory signals to determine the location of the agent in the environment.

Chapter 4

Nest-finding task

In the nest-finding task, agents use their internal state to be able to move to the proper location. The five types of recurrent neural networks that have been discussed in chapter 3 serve as controllers for the agents. A detailed description of the task is given in section 4.1. In section 4.2 the results of the experiments are analysed. In section 4.3 preliminary conclusions are drawn about the tight relation between the mechanisms used to realise an internal state and the solutions found by the agents.

4.1 Description

In [16] a robot model is introduced that incorporates the mechanisms that are assumed to underlie the ability of the Cataglyphis, a desert ant, to find its nest. The ant uses the polarization of the light to determine its direction when it is far away from the nest. However, polarization only offers approximate information. Hence, the Cataglyphis uses landmarks to find the precise location of the nest. Landmarks are objects in the environment that mark the position of a certain object or site. In the real world the landmark configuration around the nest, such as a group of little stones, can change during the lifetime of the ant. Since the ant is able to cope with this, it seems to be able to 'learn' or 'remember' these landmarks when it leaves the nest.

Returning to a certain place can be important for robots as well. If the robot cannot sense the target location directly and the landmarks around the target can change during the robot's lifetime, some information has to be retained about the target location to enable the robot to return there. That is why the nest-finding task requires an agent to have an internal state to be able to perform it. The task demonstrates how the recurrent neural networks under investigation can use their internal state to return to a target location. The internal state serves as a type of memory of the robot's experiences at the target location.

The experiment has been devised to necessitate the use of an internal state. The agent is a simulated kephera robot [29] that has to return to a target location marked by one or two light bulbs. Figure 4.2 shows the placement of the ambient light sensors on the Kephera robot. One attempt to perform the

task is called an "epoch". The agent is equipped with light sensors and starts an epoch at one of the target locations whose position in the environment is marked by the stars in figure 4.1. The agent gets some time to interact with the landmarks around the nest. After the interaction the agent is relocated to one of three fixed restart-positions with an accompanying fixed direction, represented by the arrows in figure 4.1. From there it has to return to the target it came from. The only way for the agent to disambiguate the situation after replacement is to store some information about the target location using the internal state. After a fixed time limit, the epoch ends.

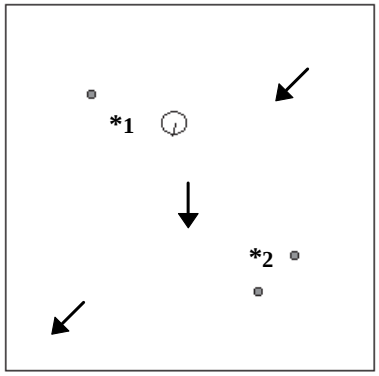

Figure 4.1: The environment of the nest-finding task

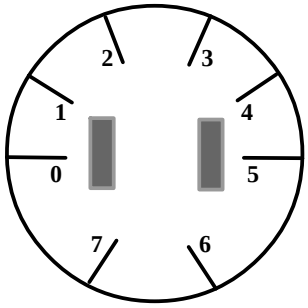

Figure 4.2: Placement of the ambient light sensors on the Kephera

The definition of the fitness function is:

$$F = \begin{cases} \frac{1}{1+(dis_{rt}/100.0)^2} & \text{, if } dis_{rt} < 250 \\ 0 & \text{, if } dis_{rt} >= 250 \end{cases} \tag{4.1}$$

In which dis_{rt} is the distance between the robot and the target location in mm. The maximum fitness per epoch is 1, since the minimum distance is 0. The fitness function is a non-linear function of distance as to increasingly reward individuals that get closer to the nest.

The parameters of the recurrent neural controllers are optimised using an evolutionary algorithm. The number of generations used for evolution is 500 and the mutation rate is 2%. Per network configuration we run the algorithm 10 times with a randomly-initialised population of controllers. All networks had three hidden neurons. Agents controlled by a NARX-network, i.e. NARX-agents, receive the sensory inputs of time step t and $t-1$, and the outputs of $t-1$ and $t-2$. In other words, the network has an input-memory of one time step and an output-memory of two time steps. The input and hidden neurons of the RNND-agents and the CTRNN-agents have neural inertia, the output neurons do not. The TDRNN-controller was tried out in both its normal and restricted form. The maximum delay on its neural connections is 50 time steps.

4.2 Results

The results are expressed in two performance measures: $\overline{F}_{best}$ and $\#r, \overline{F}_r > 0.50$. $\overline{F}_{best}$ is the sample average fitness per epoch of the best agent out of all runs of the evolutionary algorithm. $\#r, \overline{F}_r > 0.50$ is the number of runs of the evolutionary algorithm that resulted in an agent that has a sample average fitness per epoch that is higher than 0.50.[1] A fitness <= 0.50 corresponds to an agent that is not able to perform the task. To obtain the values of these measures, we have determined the average fitness per epoch of the best agent of every run of the evolutionary algorithm. We have executed every best agent for 100 epochs and taken its sample mean fitness per epoch.

Table 4.1 shows the results of the experiments. In the table, the configurations of the recurrent neural networks are represented as follows. The label '$c_{in} = 1$, $c_{out} = 2$' indicates the capacities of the in- and output-memories of the NARX-agents. The label 'I, H' represents the layers applying the mechanism of inertia for the RNND-agents and the CTRNN-agents, and the label 'R' stands for the restricted version of the TDRNN.

	Reactive	RNN	NARX	RNND	CTRNN	TDRNN	TDRNN
Type	-	-	$c_{in} = 1$, $c_{out} = 2$	I, H	I, H	-	R
$\overline{F}_{best}$	0.49	0.73	0.74	0.78	0.74	0.78	0.70
$\#r, \overline{F}_r > 0.50$	0	3	5	9	10	10	8

Table 4.1: Results of the nest-finding experiment

Since $\overline{F}_{best} > 0.50$ for all pro-active agents, all recurrent neural controllers have found solutions to the problem. Less successful RNN-agents and NARX-agents have been evolved, as illustrated by the table: $\#r, \overline{F}_r > 0.50$ has the lowest value for the RNN-controller and the NARX-controller. We analyse the agents in terms of the mechanisms on which their solutions are based. Table 4.2 shows what mechanisms form the basis of the strategies of the different controllers with the label 'B'. The label 'X' indicates that the mechanism is present in the neural network, but not used for the task when the neura network is applied as a neural controller. In contrast with the theoretical discussion on the networks in chapter 3, the NARX-network is placed immediately under the RNN. The reason for this is that, as we will demonstrate, the strategies of the NARX-agents are based on recurrency and not on its varying time delays. The change also has consequences for the way we discuss the results shown in table 4.1. The strategies of the RNND-agents and the CTRNN-agents are based on recurrency and neural inertia. The strategy of the best TDRNN-agent depends only on the mechanism of varying time delays, but TDRNN-agents have been evolved that based their strategy on both recurrency and varying time delays.

In section 4.2.1 we analyse the agents that only use their recurrent connections to solve the task, viz. the RNN-agents and the NARX-agents. After that, in section 4.2.2, we analyse the agents that also use neural inertia to solve the

[1] The total number of evolutionary runs performed per controller is 10.

Neural controller	Recurrency	Neural inertia	Varying time delays
RNN	B	-	-
NARX	B	-	X
RNND	B	B	-
CTRNN	B	B	-
TDRNN	B	-	B

Table 4.2: Mechanisms that form the basis of the strategies of the neural controllers

task, namely the RNND-agents and the CTRNN-agents. Finally the agents relying on the delays on the neural connections are discussed in section 4.2.3, viz. the TDRNN-agents.

4.2.1 Recurrency

For the RNN and the NARX-network the mechanism of recurrency is the basis of the strategy to discern between the two target locations. The experiences at a target location lead to an internal state that is retained after replacement. In other words, the activations of a subset of the recurrent neurons evolve towards an equilibrium point in the first part of the epoch. After replacement their values remain the same, regulating the behaviour of the agent as to reach the right target by adjusting the input-output mapping of the agent. The analyses of the two types of agents elucidate the strategy employed by the RNN-agents and the NARX-agents.

RNN

The RNN-agent uses its hidden neurons to make a difference between the target locations. The first two hidden neurons of the best agent function as a memory of the start location. First we introduce figures that clarify the strategy used by the RNN-agent and then we explain the strategy.

The screen shots of Evorobot clarify the strategy of the RNN-agent. Figure 4.3 and 4.5 show the trajectory of the agent in the environment. The long straight lines in the figures represent the replacement of the agent. Figure 4.4 and 4.6 display the activity plots for all activations. Per neuron the vertical axis represents the activation and the horizontal axis the time. The top two grey plots show the activations of the two outputs, $M0$ and $M1$, as a function of time. A value of 1.0 corresponds to the maximum forward speed of the motor and 0.0 to the maximum backward speed. The three black plots correspond to the activations of the hidden neurons $H0$, $H1$ and $H2$. The remaining grey plots at the bottom show the activations of the inputs $I0$ to $I7$. The vertical black line in the middle of the figures indicates the time step in which the agent is replaced to one of the replacement positions.

The RNN-agent uses its hidden neurons to make a difference between the target locations. At the moment of replacement there is a clear difference in

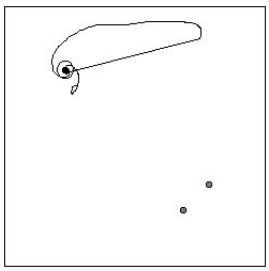

Figure 4.3: The trajectory of the RNN-agent, target 1

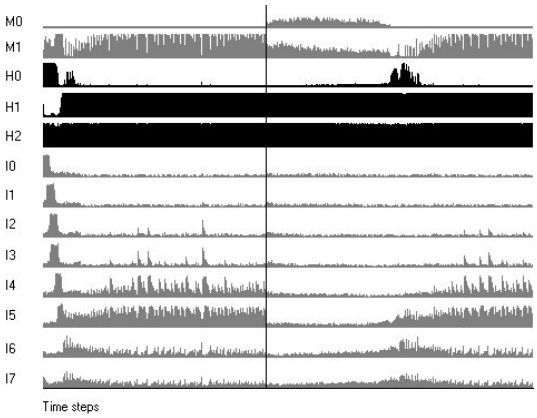

Figure 4.4: Activity plot for the RNN-agent starting at target 1

the hidden neurons of the network between when the agent has started at the first or at the second target. If the agent starts at the first target location, then $(H0, H1) = (0.0, 1.0)$, figure 4.4. Starting at the target location with the two light bulbs results in activations of $(H0, H1) = (1.0, 0.0)$, figure 4.6. The activation of the third hidden neuron, $H2$, is on average 0.98 when coming from the first target against 0.70 from the second. When the inputs are fixed at the time of replacement, the dynamical system has equilibrium points at these coordinates.

Depending on the place of the equilibrium point, the agent exhibits different behaviours after replacement. To clarify this, we show the weights of the connections between the hidden and the output neurons in table 4.3. The activations of the hidden layer result in a higher $M0$ and a lower $M1$ when the agent starts at the first target location in comparison with a start at the second target. The consequence is a straight backwards movement towards the target. In case of the second target location the agent makes circles, closing in on the nest. The internal state determines the way in which the inputs are mapped to the outputs. In this way it regulates the behaviour differently when coming from the different target locations.

	M0	M1
H0	-4.41	-2.54
H1	2.23	-4.02
H2	3.44	0.08

Table 4.3: Weights from hidden to output neurons

The sensory signals at the replacement positions also influence the behaviour of the agent after replacement and play a crucial role to return to the target from the three different replacement positions. An example of the general in-

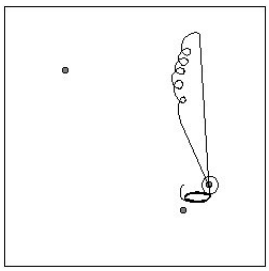

Figure 4.5: The trajectory of the RNN-agent, target 2

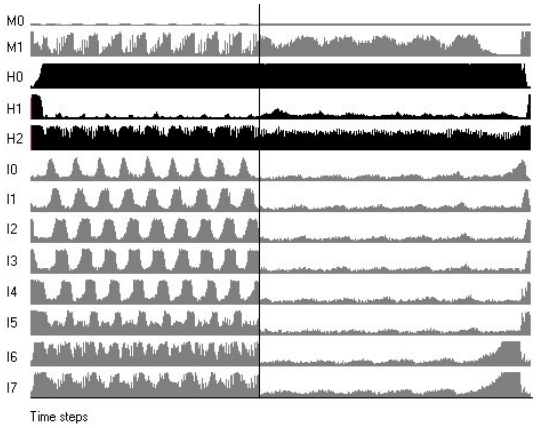

Figure 4.6: Activity plot for the RNN-agent starting at target 2

fluence of the sensory inputs on the behaviour is that the weaker sensory inputs coming from the light bulbs at every replacement position result in an activation of the second output neuron. Sensors such as $I4$ and $I5$ inhibit $M0$ and stimulate $M1$. So when those sensors fall back in activation, the activation of $M0$ increases and of $M1$ decreases. The use of sensory signals specific of the replacement positions is necessary, since every beginning of an epoch at a specific target location is equal. As a consequence, the internal state only contains the information about what target location the agent has started at. An example of the differentiation between replacement positions is that at the position in the centre of the environment, the sensors are more activated than in the right top of the environment. As a result, the motor outputs are somewhat lower which implies a faster backward movement in case of the first target, and a straighter movement in case of the second target.

No other RNN-agent established such a trustworthy strategy to return to the right target location. Two other starts of the evolutionary algorithm resulted in an agent with an average fitness slightly higher than 0.50. The agents were able to make a distinction between the targets only in a few cases.

NARX

We demonstrate that the NARX-agent uses the activations of the hidden and output neurons to remember the target location. The best NARX-agent exemplifies the type of strategy that all evolved NARX-agents use. Figure 4.7 and 4.8 are the activity plots of the NARX-agent when it starts at the first and second target location, respectively. We do not display the input- and output-memory of the network in the activity plots, since they are small translations of the input and output neurons that are already displayed.

The NARX-agent uses the activations of the hidden and output neurons to remember the target location. The weights in the dynamical system have as a

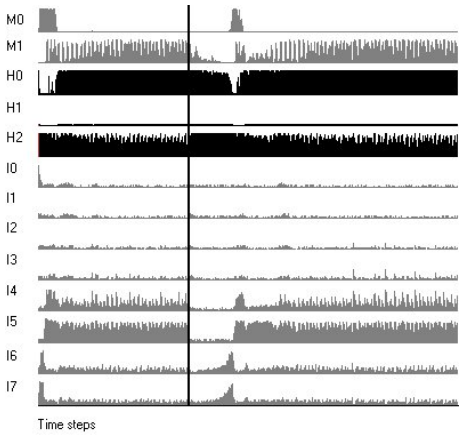

Figure 4.7: Activity plot of the NARX-agent starting at target 1

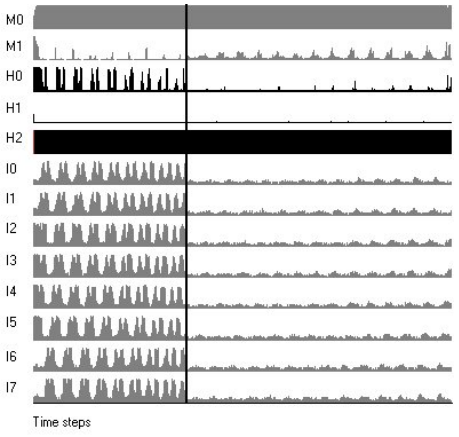

Figure 4.8: Activity plot of the NARX-agent starting at target 2

result that either $H0$ has a high activation and $M0$ does not, or vice versa. The experiences of the agent at the target location determine to which equilibrium the agent activities evolve. Figure 4.7 shows that $(M0, H0) = (0.0, 1.0)$ in case of the first target and figure 4.8 shows that $(M0, H0) = (1.0, 0.0)$ in case of the second.

Just after replacement, the sensory inputs in both figures are almost equal, but the activations of $M0$ and $H0$ remain the same as before replacement. Because of the difference in the hidden and output neuron, the agent goes forward in case of the second target location and backwards in case of the first target location.

The NARX-agent uses the recurrent connections from the hidden and the output neurons to solve the task. When we set $(M0, H0) = (0.0, 1.0)$ for time steps 201 and 202, the agent always goes to the first target or crashes. This demonstrates that the delays on the input neurons are not sufficient to reach the nest. Their delays play even no role at all, since perturbing the input memory in a similar way, does not have any influence on the performance. Only perturbing the inputs long enough for the hidden and output layer to change activations has an effect on performance. So the most important part of the strategy is the equilibrium point of neural activations that is maintained by the recurrent connections.

The delays on the recurrent connections offer some robustness to the internal state. Inciting the agent to go to the first target for only one time step does not result in a deterioration of performance. If the output memory is larger, more time steps are necessary to perturb the internal state. So the delays play a role in the maintenance of the internal state, but are not the basis of the strategy.

Other NARX-agents also use the recurrency of their output or hidden neurons or both as a memory of the target location.

4.2.2 Neural inertia

In this section we discuss how the RNND-agents and the CTRNN-agents use both the mechanisms of recurrency and neural inertia to return to the right target location. We first analyse the best RNND-agent and then the best CTRNN-agent to explain their strategies.

RNND

For the best RNND-agent we demonstrate that it uses neural inertia instead of recurrency to solve the task.

The agent does not use its recurrency to remember the target location, since the activations of the hidden neurons at the time of replacement are not very different for the two different target locations. Table 4.4 displays the sample average activations of $H1$ and $H2$ at $t = 200$ based on 100 epochs. The standard deviations of the average activation of $H1$ are quite small, 0.06 when starting at target 1 and 0.008 when starting at target 2. The small difference in the second hidden neuron ($H1$) at $t = 200$ disappears quickly after the replacement. It lasts too short to explain how the RNND-agent solves the task.

	Target 1	Target 2
$\overline{H1}$	0.73	0.96
$\overline{H2}$	0.33	0.33

Table 4.4: Average activations at $t = 200$

To understand how the agent does achieve to make a difference between the target locations, it is useful to look at the activity plot of its controller when it starts at the two different targets. Figure 4.9 and 4.10 are the activity plots when the agent starts at the first and the second target, respectively. The vertical black line indicates the time step in which the agent is replaced. The agent uses neural inertia in the sensor neuron $I1$ to serve as a memory of the starting location. Figure 4.9 and 4.10 show that $I1$ has a different value at the time of replacement when the agent comes from target 1 or from target 2.

The activation of $I1$ at the time of replacement is related to the target location and it influences the motor outputs long enough to reach the target. The sensor $I1$ is hardly stimulated by the first target location, since the agent keeps the left side of its body away from the single light bulb. In the case of the second target location, it turns around on its axis between the two lights, activating all sensors, including the left one. That its activation lasts long enough after replacement, is a consequence of the very high delta value of 0.97. New input values only slowly influence the activation of the sensory input. $I1$ inhibits $H2$ and $M1$, while $H2$ stimulates $M1$. So $I1$ inhibits the right motor, while it excites the left motor. As a consequence, the agent turns around to the right before it goes straight ahead in case it has started at the second target location. To test if neural inertia was required for the strategy, at $t = 200$ the

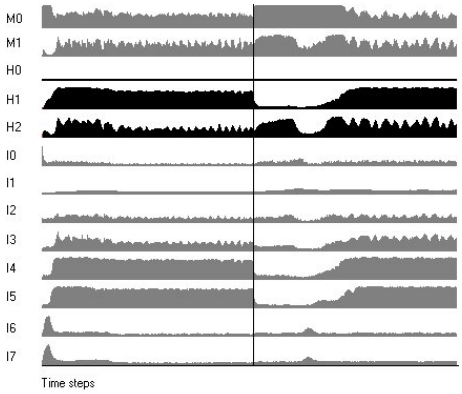

Figure 4.9: Activity plot of the RNND-agent starting at target 1

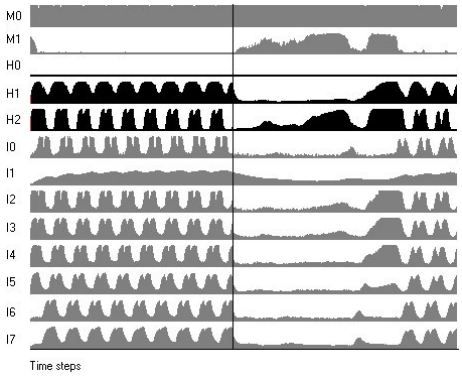

Figure 4.10: Activity plot of the RNND-agent starting at target 2

internal state was disturbed. When $I1$ is put to 0.0 at $t = 200$, the agent always turns towards the first nest and in two out of three cases it succeeds to reach it. Evidently, the RNND uses the neural inertia of $I1$ to remember the target location.

Another evolved agent applies a 'hybrid strategy': it uses both the recurrency of the hidden neurons and the neural inertia to remember the target location. The sensory signals associated with the first target location result in a high activation of the second hidden neuron which excites itself. The sensory signals related with the agent's behaviour around the second target location however, result in a low activation of the second hidden neuron. So the hidden layer makes a difference between the two target locations and with the help of its recurrent connections it stays the same after replacement. One of the sensors involved in the activation of the hidden neuron is $I6$, which keeps inhibiting $H1$ even after replacement because of its high neural inertia.

CTRNN

We demonstrate that the best CTRNN-agent uses a hybrid strategy to return to the target location. Both the neural inertia of the input neurons and the recurrency of the hidden neurons are involved in the strategy. Figure 4.11 and 4.12 are the activity plots of the CTRNN-agent when it starts respectively at the first and the second target location. The figures show that $H1$, $H2$ and $I2$ have values correlated to the target location at the time of replacement.

To investigate the importance of the two parts that constitute the hybrid strategy, we perturb either the relevant input units or hidden units at t = 200 and measure the drop in performance that this perturbance causes. Table 4.5 shows the sample average activations of $H1$, $H2$ and $I2$ at the time of replacement. Separately for the hidden neurons and the input neuron, we set the activations to the values related to the second target location.

For the hidden units, we set $(H1, H2) = (1.0, 0.0)$. Table 4.5 shows that by doing this, we incite the agent to go to the second target. The resulting average

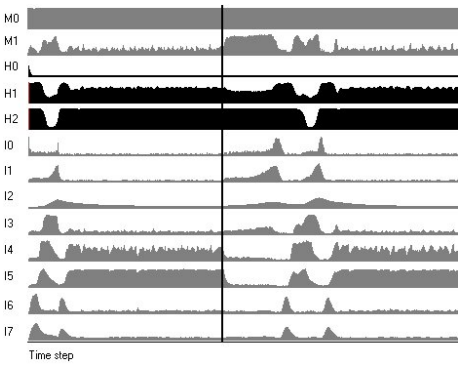

Figure 4.11: Activity plot of the CTRNN-agent starting at target 1

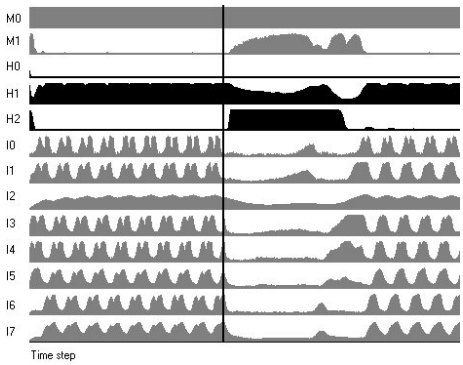

Figure 4.12: Activity plot of the CTRNN-agent starting at target 2

	Target 1	Target 2
$\overline{H1}$	0.6894	1.0
$\overline{H2}$	0.9892	0.0
$\overline{I2}$	0.0755	0.556

Table 4.5: Average activations at $t = 200$

fitness per epoch is 0.57, based on 100 runs. The agent makes mistakes when it is replaced to the bottom left of the environment. To verify the importance of the neural inertia, we set $I2$ at time t = 200 to 0.556 to stimulate the agent to go to the second target. Here the average fitness per epoch based on 100 runs is 0.43. The agent never succeeds in returning to the first target. The influence of $I2$ on the hidden layer does not seem to cause the difference in average performance, since the weights of the connections from $I2$ to $H1$ and $H2$ are small compared to the other weights. Hence it seems that $I2$ is crucial to the strategy, while $H1$ and $H2$ are needed to find the target location from the bottom left replacement position.

Other agents use hybrid strategies, or just use their hidden layer or neural inertia to regulate their behavior. Although one might suppose that a hybrid strategy is more robust, this is not supported by the fitnesses of these agents.

4.2.3 Varying time delays

In this section we discuss how the delays on the neural connections can be used to return to the proper target location. The only agent that followed this strategy is the TDRNN-agent, whose delays are determined by evolution.

TDRNN

We show that the mechanism of recurrency hardly plays a role in the strategy of the TDRNN-agents by analysing the restricted TDRNN-agent with the highest fitness during evolution. The agent uses the delays on the neural connections to perform the task.

The activations of the hidden neurons at time of replacement differ between two successive epochs. However, the TDRNN-agent does not use the activations of the hidden layer to remember the target location, since the activations of the hidden layer depend mostly on the input layer. First, $H1$ is unused. Secondly, the weights of the connections between $H0$ and $H2$ and their self-recurrent connections are all smaller than 1.0 while their connections coming from sensory inputs are larger.

When disturbing the recurrent activations of the TDRNN at the time of replacement, the agent still returns to the right target location.

For the RNND, we verified the importance of the hidden neurons for the strategy by disturbing them at $t = 200$. However, the output neurons of the TDRNN at $t = 200$ are not influenced by the hidden neurons at $t = 200$, but by those of a number of time steps before. The largest time delay between the hidden and the output layer is 37 time steps. That is why we replace the normal activations of the hidden layer from time step 163 to 200 of $(H0, H1, H2)$ by a fixed $(0.05, 0.05, 0.7)$. These are their average activations when starting at the first target. If the hidden layer is crucial to the strategy, the agent will go to the first target. By following this procedure the influence from the inputs on the hidden layer is restored at $t = 200$. These inputs still correspond for a while to a normal start, since the outputs will only be influenced by the fixation of the hidden layer after $t = 200$. The influence of the inputs on the outputs stays the same, although the outputs and thus the inputs will be different after $t = 200$ because of the fixation.

Figure 4.14 shows the activations of the hidden layer from 163 to 200 in three different cases: beginning at target 2, fixing the hidden neurons and beginning at target 1. Figure 4.13 illustrates the effect these activations have on the hidden layer and the outputs after $t = 163$. The activity plot with the fixed hidden layer in the figure are from the agent starting at target 2.

There is some similarity just after replacement between the activations when starting from target 1 and when starting from target 2 with fixed hidden neurons. The activity plots in figure 4.14 show that in both cases $M1$ is higher and $M0$ lower than in a normal epoch started at the second target. But this lasts only for 37 timesteps, exactly the delay from the hidden layer to the second output unit.

Overall however, there is a bigger resemblance between the activations when fixing the hidden units and the usual activations when starting at target location 2. More importantly, this is the target location the agent goes to. So the TDRNN uses the delays on the neural connections to remember the target location.

The best TDRNN-agent of the restricted form succeeds in changing the behaviour of the robot after replacement with a delay of transformed sensory

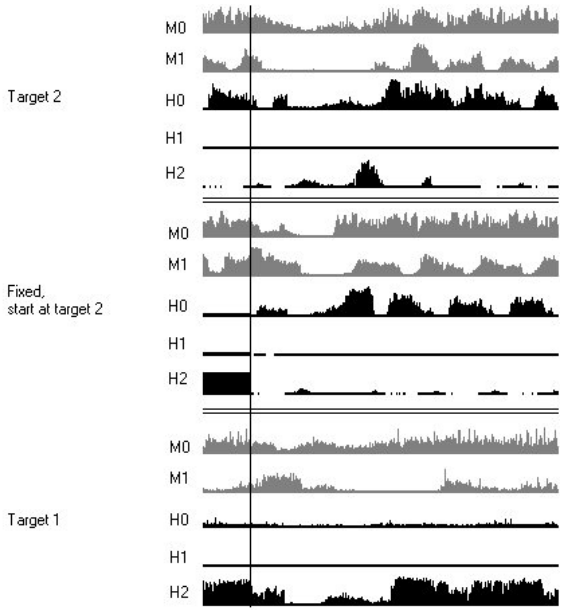

Figure 4.13: The output and hidden neurons $t >= 163$

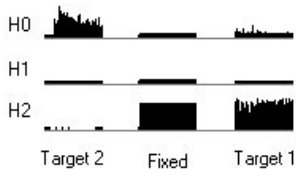

Figure 4.14: $h0$, $h1$ and $h2$, $t \in [163, 200]$

inputs for a period between 8 and 64 time steps. In a minority of cases however, agents base their strategy on the mechanism of recurrency, since they use their hidden neurons to store the target location.

4.3 Discussion

The nest-finding task clarifies how the different recurrent neural controllers can use their internal state as a simple memory. The internal state enables returning to the target location. The values of the internal-state variables at the time of replacement have to be adequate to regulate the agent's behaviour from all three replacement positions. The required differences in behaviours for different replacement positions can only be achieved by exploiting the sensory inputs received at those positions.

Our main finding is that there is a close relation between the different strategies of the five types of pro-active agents and the mechanisms they use to realise an internal state: recurrency, neural inertia or varying time delays. We briefly discuss how the five types of recurrent neural networks differ in their strategies to deal with the nest-finding task.

The internal state of the RNN corresponds to the hidden activations at the last time step. Recurrency is at the basis of the agent's strategy, since it is the only mechanism the network has to influence future input-output mappings. The best agent uses its recurrency to create an equilibrium point at $(H0, H1)$ = (0.0, 1.0) when the agent starts at the first target and $(H0, H1)$ = (1.0, 0.0) when it starts at the second target.

Although the NARX-network has delays that exceed a single time step, the agents' strategies are based on recurrency. The input memory is not a part of the agent's strategy. Also the NARX-agents use equilibrium points of their dynamical system to remember the target. Because of the output memory, the NARX-network can use both the hidden and the output neurons to return to the proper target location. The network uses outputs of multiple time steps, which offers more robustness to the internal state.

RNND-agents use both recurrency and neural inertia to return to the target location. The agent analysed used neural inertia to regulate the agent's behaviour after replacement. The sensory inputs at the target locations lead to different activations in an input neuron with a high neural inertia. The high inertia allows the agent to remember the target long enough to find the direction towards the right target. The agents controlled by a CTRNN follow similar strategies.

The TDRNN-agents solve the nest-finding task by delaying the activations of the input neurons. The internal state of the TDRNN contains all activations that are conducted through the neural connections. Hence, the size of the internal state is large compared to the other networks and disturbing the internal state is hard.

The nest-finding task does not provide a satisfactory answer to our first research question, since it does not result in significant differences in the capabilities of the agents controlled by the five types of recurrent neural networks. However, it does clarify that the strategies of the agents are tightly related to the mechanisms applied to realise an internal state. The RNN-agents and NARX-agents rely only on the mechanism of recurrency. The mechanisms of neural inertia and varying time delays offer the RNND-agents, CTRNN-agents, and TDRNN-agents the possibility of applying alternative strategies for performing the task. In the mechanisms of neural inertia and varying time delays, the internal state variables have more ways to influence future input-output mappings. The results shown in table 4.1 suggest that when the strategy is based on the additional mechanisms of neural inertia or varying time delays, it leads to a higher evolvability: the performance measure $\#r, \overline{F}_r > 0.50$ is lower for the RNN-agents and the NARX-agents.

Chapter 5

Driving task

In chapter 4 the focus was on how the five types of neural controllers use the internal state to memorise past experiences. In the driving task the emphasis is on the dynamical nature of the five recurrent neural controllers. The recurrent neural controllers have to generate activations in the internal state to enable the agent to continue driving without sensory inputs. The driving task addresses the following two questions. In what ways do the different neural controllers achieve this and what is the relation between their solutions and the mechanisms employed to realise an internal state? We describe the task in further detail in section 5.1. In section 5.2 the evolved agents are analysed. In section 5.3 a variant of the task named "Total blackout" is introduced and we briefly analyse the results of the related experiments. In section 5.4 we discuss the analyses of the driving task.

5.1 Description

A reactive agent executes actions based on the sensory inputs it receives from the environment. The actions of the agent have an effect on the agent-environment system, for example on the position of the agent. By means of this effect, the actions of a reactive agent co-determine the inputs that the agent will receive. A reactive agent can use the external feedback loop concerning actions and inputs to simplify problems [22].

A pro-active agent has an internal feedback loop in addition to the external one. It has an internal state that influences future actions and internal states. Interpreted as a memory, the internal feedback loop serves to retain information received from the environment and regulate the behaviour accordingly. For example, the RNN-agent evolved for the nest-finding task has two different equilibrium points that are reached when starting at the two different target locations. For the driving task the internal feedback loop has to be used as a generator of dynamics rather than as a storage device alone.

What would happen if the external feedback loop of a reactive agent would be cut off by putting all inputs to zero? The answer is simple: since a reactive agent forms a function from the inputs to the outputs (see equation 3.1), it will only be able to perform one action in response to the zero input. In contrast, a

pro-active agent is able to use its internal feedback loop to produce activations that vary in time.

In the driving task, a Kephera robot has to drive along a trajectory in a clockwise direction. The environment is shown in figure 5.1. The circles in figure 5.1 are zones used to determine the fitness of an agent. Every zone an agent crosses in the right direction will increase its fitness by 1. The agent starts an epoch driving around with all of its sensors enabled. At a certain moment the agent is 'blinded', i.e. all infrared sensors are disabled. For a reactive neural controller this results in constant motor outputs. Hence, the reactive agent will hit a wall or turn on its axis. However, an agent with an internal state can use its internal feedback loop to keep on moving.

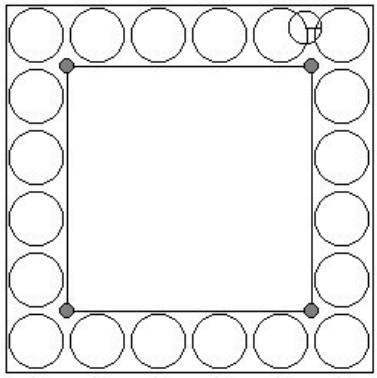

Figure 5.1: Environment of the driving task

In the first experiments one execution of an agent is called an epoch and it consists of a sensing and a blind period. The whole epoch lasts 1600 time steps and at $t = 500$ the agent is blinded. The number of generations used in the evolutionary algorithm was 500 and the mutation rate 1%. All agents also had the motor outputs as inputs to the network, although these were set to zero during the period of blindness. Only the NARX-network can still use the recurrency of the output units after blindness. The number of hidden neurons in controllers were equal to 3 or 5.

5.2 Results

We use two performance measures to express the results, $\overline{F}_{best}$ and $\#r, \overline{F}_r > 36.00$. $\overline{F}_{best}$ is the sample average fitness per epoch of the best agent of all runs, based on 100 epochs. $\#r, \overline{F}_r > 36.00$ is the number of runs of the evolutionary algorithm that resulted in agents that have a sample average fitness higher than 36. This is the highest fitness an agent can obtain when it drives straight ahead when it turns blind. Table 5.1 shows the results. The table shows that the NARX-agent has the highest fitness. The TDRNN-agents have the highest evolvability, i.e. it has the most runs of the evolutionary algorithm resulting in successful agents.

	Reactive	RNN	NARX	RNND	CTRNN	TDRNN	TDRNN
Hiddens	3	5	5	5	3	3	5
Type	-	-	$c_{in} = 2$, $c_{out} = 2$	I, H	I, H	-	R
$\overline{F}_{best}$	34.32	56.65	59.98	36.00	39.86	56.72	54.18
$\#r, \overline{F}_r > 36.00$	0	2	2	0	1	3	8

Table 5.1: Results of the first driving task experiment

When a reactive agent is blinded, it can output only one action for the rest of the epoch. As a result, the reactive agents drive straight on when they become blind and obtain a maximum score of around 34.32 per epoch. The reactive agents only cross zones and gain fitness during the sensing part of the epoch.

When a successful pro-active agent is blinded, the activities in its neural controller do not converge to an equilibrium, even though the sensory inputs are kept constant. We observed that these controllers maintain a limit cycle in their internal dynamics. The internal state results in motor actions that are appropriate to continue to drive and even to make turns at the appropriate locations.

The networks that shared strategies in the nest-finding task, also share strategies in the driving task. Table 5.2 is a copy of table 4.2 and shows on what mechanisms the different types of agents base their strategies. We argue that the RNN-agents and NARX-agents base their strategy for the driving task on the mechanism of recurrency, since both types of agents exhibit limit cycles with short periods for the driving task. The neural controllers relying on neural inertia do not yield effective behaviours. The table shows that no RNND-agent is able to obtain a fitness higher than 36 and that the average fitness per epoch of the best CTRNN-agent is only 39.86. The TDRNN-agents base their strategy on the mechanisms of recurrency and varying time delays and can have limit cycles with either long or short periods.

We discuss the strategies associated with the three mechanisms in sections 5.2.1 to 5.2.3. As for the nest-finding task, the discussion in section 5.2.1 on the mechanism of recurrency deals with the RNN-agents and the NARX-agents. In section 5.2.2 we analyse the agents that rely in addition to neural inertia, viz. the RNND-agents and the CTRNN-agents. Finally, in section 5.2.3 we analyse the agents whose strategies are based on the mechanism of varying time delays, viz. the TDRNN-agents.

5.2.1 Recurrency

The agents that relied on recurrency for the nest-finding task share solution properties for the driving task. The RNN-agents and the NARX-agents generate limit cycles with short periods and have high performances. We first discuss the RNN-agents and then the NARX-agents.

Neural controller	Recurrency	Neural inertia	Varying time delays
RNN	B	-	-
NARX	B	-	X
RNND	B	B	-
CTRNN	B	B	-
TDRNN	B	-	B

Table 5.2: Mechanisms that form the basis of the strategies of the neural controllers

RNN

We demonstrate that the RNN-agents are able to continue to drive because the dynamics in the internal state follow a limit cycle during blindness. The two successful RNN-agents apply the same strategy. Figure 5.2 displays the complete trajectory of the agent of a whole epoch. Figure 5.3 is an activity plot before and after the time step in which the best agent is blinded. The vertical line is drawn at the time step that the agent is blinded.

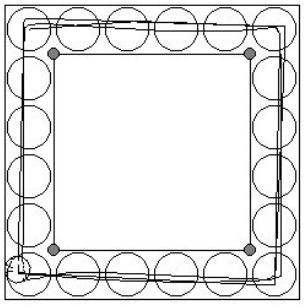

Figure 5.2: Trajectory of the RNN-agent

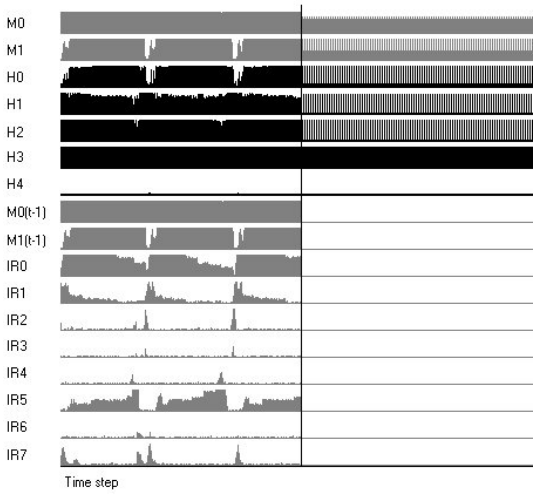

Figure 5.3: Activity plot of the RNN-agent

We first explain what happens to the internal state when the sensory inputs are disabled. Then we clarify how the internal dynamics have as effect that the agent continues to drive. Finally, we show that the internal state does not serve to retain information from the sensing part of the epoch.

The dynamical system realised by the RNN undergoes a bifurcation when the input neurons are set to zero, and a limit cycle emerges. The activations right of the vertical line in figure 5.3 are periodical, while left of it they are not. To demonstrate that a bifurcation occurs when the inputs are set to zero, we take $H2$ as an example. During sensing $H2$ is always close to 1.0, since the four input neurons that are most active are connected to $H2$ with large weights.

When the input neurons are all set to zero, the neural input to $H2$ becomes much smaller and $H2$ becomes more dependent on neural inputs coming from the other hidden neurons. The connections between the hidden neurons cause the limit cycle to arise. $H0$ and $H2$ excite $H1$, but inhibit themselves. For $H1$ the inverse is valid. As a result, the three hidden neurons exhibit an alternating pattern of high and low activations, as shown in figure 5.3. Prolonging the epoch to for example 10000 time steps does not result in the activities dying away; the hidden and output neurons show the same activation pattern during the whole epoch.

How do the internal dynamics cause the agent to drive on during blindness? The motor outputs undergo very sudden changes during blindness: motor output 0 alternates between 0.63 and 0.76, while motor output 1 alternates between 0.99 and 0.44. As an effect, the network alternates continuously between two different motor actions: going forward and left and making a sharp turn to the right. In a corridor this alternation of actions almost results in a straight trajectory. If the agent is close to a wall, the movement forward and to the left results in a collision. When the agent performs the sharp turn to the right however, a small change in direction is achieved. The collisions and changes in direction continue until the agent faces the corridor again.

The only link between the activitions before and during blindness consists of the weights of the network that have to be appropriate both for driving with and without sensory signals. The bifurcation erases the information in the internal state from the first part of the epoch. If random activations are chosen at the time of blinding, the average fitness is 56.30. Disturbing the information in the internal state at the time of blinding does not influence the agent's fitness.

NARX

We discuss the NARX-agents together with the RNN-agents, because they exhibit limit cycles with a short period, just like the RNN-agents. The NARX-agent has the highest average performance. The trajectory of the best NARX-agent is shown in figure 5.4. Figure 5.5 is an activity plot of the hidden and output neurons before and after $t = 500$.

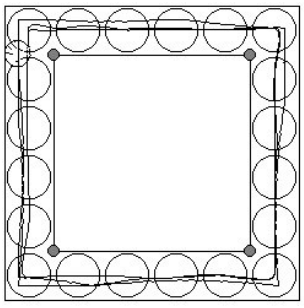

Figure 5.4: Trajectory of the NARX-agent

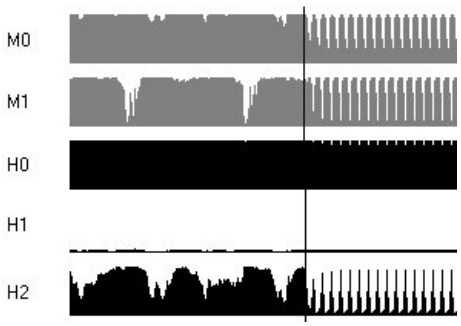

Figure 5.5: Activity plot of the NARX-agent

Within a few time steps after blindness sets in, a limit cycle is reached so that the agent drives on. The agent moves fast forward, while turning to the left and to the right in an alternating fashion. Since $M1$ almost drops to 0 (figure 5.5), the turn to the right is almost a turn around its own axis. This is necessary when the agent encounters a wall and has to turn right towards the corridor. The turns to the left serve to keep the trajectory of the agent reasonably straight in the corridor. The NARX-agent drives closer to the wall than the RNN-agent whose trajectory is shown in figure 5.2.

As illustrated in figure 5.5, during blindness the activations change very quickly and the length of the period consists of a few time steps. Other agents followed the strategies of the RNN-agent. There is one agent with 5 hidden neurons that has interesting activations, since it has periodic activities with a period of 23 time steps. However, the activations still change very abruptly. Figure 5.6 is an activity plot of the hidden and output neurons of that agent when the sensory inputs are switched off.

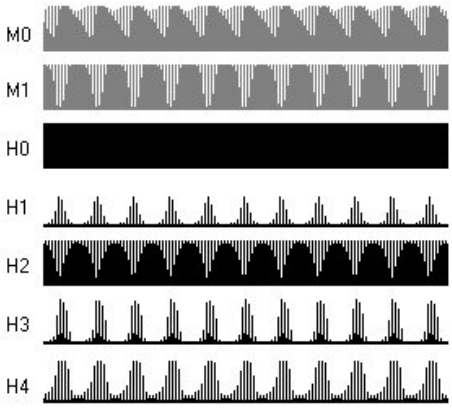

Figure 5.6: Activity plot of the NARX-agent

If the output-memory is enlarged, it enhances the number of limit cycles a NARX-agent can establish. An output memory of 4 results in 5 successful agents for 10 starts of the evolutionary algorithm. Therefore, the delays have an influence on the strategies of the NARX-agents for this task.

5.2.2 Neural inertia

Not many successful agents with neural inertia have been evolved for the driving task. We will discuss first the RNND-agents and then the CTRNN-agents. The RNND-agents are not able to perform the task. The only successful CTRNN-agent exhibits a limit cycle with a long period.

RNND

The activations in the hidden layer of the RNND-agents evolve towards an equilibrium when the sensors are disabled. All agents drive against the wall. The best RNND-agent always drives to the end of the corridor. This is why the

average fitness of the best RNND-agent is a bit higher than that of the reactive agent.

CTRNN

The neural inertia in the CTRNN results in slow internal dynamics. The CTRNN-agent exhibits a limit cycle with a long period. Figure 5.7 displays a trace of the agent and figure 5.8 is the activity plot of the CTRNN-agent. The left part of the activity plot shows the activations of the network when the agent can still use its sensors. The right part of it shows the activations just before and after t = 500, i.e. the moment that the sensory inputs are disabled.

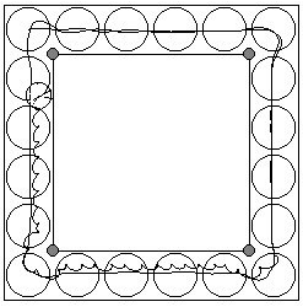

Figure 5.7: Trajectory of the CTRNN-agent

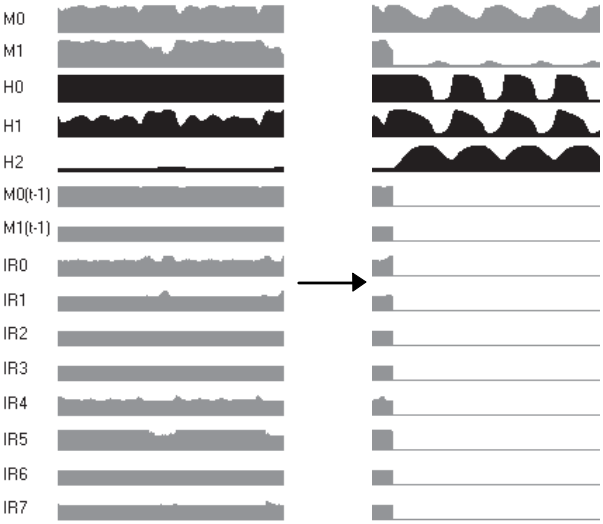

Figure 5.8: Activity plot of the CTRNN-agent

We demonstrate that the strategy of the CTRNN-agent is in principle the same as that of the agents relying on recurrency. The dynamics of the CTRNN-agent cause a limit cycle to arise when the sensors are blinded. The motor actions that result from the limit cycle, enable the agent to continue to drive. The internal state does not retain information from the sensing part of the epoch to enable the limit cycle.

The dynamical system realised by the CTRNN undergoes a bifurcation when the inputs are set to zero. Non-zero inputs inhibit the third hidden unit. So when the agent becomes 'blind', the inhibition from the input units falls away and the activity of the third hidden neuron increases. The third hidden unit inhibits the other two hidden units while they excite him. This is the cause of the limit cycle, since the inhibition of $H2$ of the other two hidden neurons will lead to a decrease of its own activity. This decrease leads to higher activations in the other hidden neurons, that start to activate $H2$ again. Because of the neural inertia, the period of repeating activations is 38 time steps.

The limit cycle that arises during blindness causes the agent to move forward and make turns in the environment. In figure 5.8 it is shown that the left motor

changes between fast forward ($M0 = 1.0$) and not moving ($M0 = 0.5$). The right wheel moves backwards fast. When the left motor moves forward quickly, the agent turns around its own axis. As a consequence, the back of the agent faces the corridor. This enables the backward circular movements the agent makes when the left wheel does not move. These movements are the 'hops' in figure 5.7. The straight lines in the trajectory are related to when the senors function properly.

The CTRNN-agent does not use its internal state to retain information about the environment as sensed when the inputs are enabled. The bifurcation erases all information in the internal state. Replacing the activations of the hidden neurons at the time of blinding by random activations indeed does not change the fitness of the agent, the average obtained with 100 epochs is 40.02. So at the time of blinding, the internal state contains no information necessary to continue to drive.

5.2.3 Varying time delays

We argue that the TDRNN-agents base their strategies both on recurrency and varying time delays, since the TDRNN is the only recurrent neural controller that exhibits many different limit cycles, both with long and with short periods.

TDRNN

TDRNN-agents exhibit limit cycles with both short and long periods. We first show the internal dynamics of multiple TDRNN-agents to demonstrate the diversity of possible dynamics. Then we discuss the long time interval that the best TDRNN-agent needs to establish the limit cycle. Finally, we show that the TDRNN-agents do not use information from the sensing part of the epoch to enable the agent to drive during blindness.

The TDRNN-agents exhibit various dynamical systems that satisfy the requirements of the task. Two strategies of unrestricted TDRNN-agents show similarities with the CTRNN-agent, with as a difference that they move along the outer wall. Figure 5.9 displays the activity plots of other TDRNN-agents during blindness. All agents are of the restricted form of the TDRNN. The TDRNN-agents with 5 hidden neurons that are not displayed, have dynamics similar to those of the agents with three hidden neurons.

TDRNN-agents can generate slowly changing activations. Figure 5.9 shows that the dynamics of the agents with three hidden neurons are all smooth. This is surprising, since the TDRNN has the same activation function as the Elman and NARX-network that did not result in any smooth changes in activations in the experiments.

Because of the long delays, the transition of the internal dynamics from sensing to blindness lasts a long time. Figure 5.10 shows the activity plot of the best TDRNN-agent. The black vertical line in figure 5.10 indicates the time step at which the agent is blinded. The largest delay from the input layer to the hidden layer is 49 timesteps, and the largest delay from a hidden neuron to another one is also 49. Therefore, after blinding the inputs still have an influence

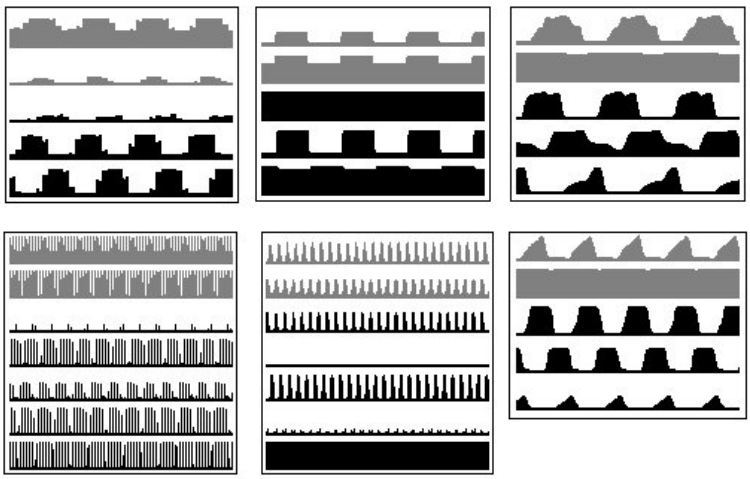

Figure 5.9: Activity plots of restricted TDRNN-agents showing hidden and output layers during blindness

on the hidden neurons and outputs. Figure 5.10 demonstrates this influence, since the periodical activities are only established after many time steps. This supports the results of the nest-finding task; it takes more time to change the internal dynamics of a TDRNN. The internal state of a TDRNN-agent is harder to disturb, but it also takes longer to control.

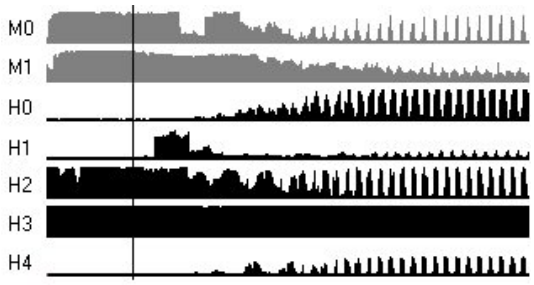

Figure 5.10: Activity plot of the TDRNN-agent

The TDRNN-agent does not use information from the sensing part of the epoch to be able to drive on during blindness. Disturbing the internal state from t = 450 to t = 500 does not have an effect on the activations after the transition. Replacing the internal state by a random one, results in almost the same average fitness based on 100 epochs as without the disturbance: 56.38.

5.3 Total blackout

The first experiment demonstrates that an agent can use its internal state in co-ordinance with the sensory inputs it receives and that it can drive on when these sensory inputs cease to be provided. In the first part of the epoch, the agent can use its sensory inputs. As a consequence, the parameters of the neural network (i.e. the weights, delays and time constants) are adapted to exploit sensory information. A subset of these parameters also have to be adapted to driving during blindness. This double adaptation restricts the possible parameter-configurations for the blindness part.

Here we introduce a second experiment that investigates what would happen if the parameters are only used in a blind period. The driving task would then become a sequence-generation task and the dynamics of the internal state would not depend on the sensory inputs at all. So in the new experiment there are no sensory inputs during the whole epoch. For the experiment we express the results in the performance measures $\overline{F}_{best}$ and $\#r, \overline{F}_r > 6.00$. $\overline{F}_{best}$ is the average fitness per epoch of the best agent of all evolutionary runs. $\#r, \overline{F}_r > 6.00$ indicates the number of successful agents out of the ten runs of the evolutionary algorithm. There are six zones in a corridor, so an agent that is not able to generate adequate dynamics will have a fitness per epoch $<= 6.00$. Table 5.3 contains the results of this experiment.

	RNN	NARX	RNND	CTRNN	TDRNN
Hiddens	3	5	5	5	3
Type	-	$c_{in} = 1$, $c_{out} = 3$	I, H	I, H	R
$\overline{F}_{best}$	36.00	51.33	22.00	3.49	43.64
$\#r, \overline{F}_r > 6.00$	1	7	1	0	3

Table 5.3: Results of the second driving task experiment

The table clarifies that the best RNND-agent is able to perform the task, since $\overline{F}_{best} = 22.00$. The RNN-agent and the RNND-agent follow the same strategy as the RNN-agent in the first experiment of the driving task. The hidden neurons of the RNND-agent that are involved in the limit cycle have almost no neural inertia ($d_i \approx 0$). Since the output neurons do not have inertia, the RNND-agent is almost equal to an RNN-agent. Both networks resulted in only one agent that was able to drive around in the environment, $\#r, \overline{F}_r > 6.00$ in the table is one, while the number of evolutionary runs was ten. There were no successful CTRNN-agents for this experiment.

The results for the TDRNN and the NARX-network demonstrate that the double adaptation indeed restricts the possible parameter configurations and dynamics during blindness. The time-delay recurrent neural network generated sufficient solutions. Without the sensory inputs, the restricted form of the TDRNN with 3 hidden neurons produces quickly changing inputs. It was only able to do this with 5 hidden neurons for the first experiment. The NARX-network resulted in many more solutions to the task, 7 out of 10 runs resulted in successful agents. The length of the periods of activations are still short, all

shorter than that of the agent whose activity plot is shown in figure 5.6.

5.4 Discussion

All types of recurrent neural networks are able to make the agent drive in the absence of sensory inputs. To keep on driving, all networks generate a limit cycle to produce periodic activations. The internal feedback loop is used to generate the appropriate motor outputs for the task. In the first experiment only the parameters of the networks form a link between the sensory and blind parts of the epochs. Disturbing internal states when the sensory inputs are set to zero does not deteriorate the performance of any of the evolved agents.

Our main result is that the agents that use the same strategies for the nest-finding task, share solution properties for the driving task. The recurrent neural controllers relying on recurrency have limit cycles with short periods. The controllers relying on neural inertia have not evolved many capable agents. The controller with adjustable delays can have limit cycles with either short or long periods. We briefly discuss how the five types of recurrent neural networks differ in their strategies to deal with the driving task.

The activations of the RNN during blindness oscillate between two values which results in a high fitness. However, the best agents are controlled by a NARX-network. If the agent is evolved in total blindness, the number of successful agents is very large. Two out of four agents in the first experiment follow the same strategy as the RNN-agent does. For all agents the length of the periods is small, and neural activations change very quickly. The output memory seems to establish richer dynamics that are absent in an RNN. By enabling richer dynamics, the varying time delays do play a role for the strategies of the NARX-agents.

There are not many agents relying on neural inertia that are able to drive during blindness. The only RNND-agent able to perform the task is almost an RNN-agent because of its low delta-values. The CTRNN-agent has a limit cycle with a large period, which results in a suboptimal fitness. The slower dynamics make the agent loose time in the corridor. The core of the problem for the networks with neural inertia is that agents with short-term dynamics perform better on the driving task. Although the deltas and time constants can result in quickly changing activations, the agents with neural inertia appear to have a bias towards slowly changing activations.

The only network whose delays are adjusted by evolution, the TDRNN, has the highest evolvability measured in successful agents for the first experiment. This result supports the claim in [9] that the TDRNN has a higher capacity to model dynamical systems than a standard recurrent neural network. Even though the TDRNN applies the same activation function as the Elman and NARX-network, it is able to generate smooth periodic activities, as shown in figure 5.9. In addition, it can generate limit cycles with either small or large periods. However, when the agent is blinded in the first experiment, it takes a long time to reach the limit cycle. So a large internal state might be hard to disturb, but it is also hard to control.

The driving task offers insight into the first research question, since it demonstrates that the RNND-agents and the CTRNN-agents have lower evolvability and performances on the task than the other networks. The reason of the lower performances concerns the second research question, since the lower performances are related to the use of the mechanism of neural inertia. The effect of the neural inertia is that both the RNND-agents and the CTRNN-agents are biased towards slowly changing activations, while agents with short-term dynamics are more successful at performing the driving task.

Chapter 6

Self-localisation task

The self-localisation task requires that the recurrent neural controllers use the internal state together with the sensory signals to determine the location of the agent in the environment. In this task agents are required to be able to use sensory signals on different time scales. This ability is related to the mechanisms of recurrency, neural inertia and varying time delays. In section 6.1 we describe the self-localisation task in more detail. In section 6.2 the evolved agents are analysed and in section 6.3 the results and their consequences are discussed.

6.1 Description

One of the possible uses of an internal state is to differentiate between situations in which the sensory information is the same, but different actions are necessary. A case in which the same sensory inputs should lead to different actions is called perceptual aliasing [22]. For reactive agents perceptual aliasing poses a problem, since they can only react in one way to one type of input. Sensory-motor coordination can sometimes alleviate or solve this problem [27], but there are situations in which perceptual aliasing is too hard to solve by sensory-motor coordination.

A task that exemplifies the utility of the internal state to solve perceptual aliasing is the task of self-localisation in an environment [23]. An agent has to drive around in a loopy corridor and to indicate with an output neuron in what part of the environment it is located. Figure 6.1 is a drawing of the environment for this task. The arrows indicate the direction in which the agents are forced to drive. The different rooms are painted in different shades of grey. If the agent is in the top room (light grey), the localisation output neuron has to have a value in the interval $[0, 0.5]$ to be correct. In the bottom room (dark grey) this value has to be in $\langle 0.5, 1]$. Figure 6.2 is also a drawing of the environment, displaying the zones in the environment that are used during evolution to stimulate agents to drive around in the environment.

This task is too difficult for a reactive agent, since the two different parts of the environment are largely the same from the viewpoint of the agent. For example, for the agent there is no difference between being in the top or bottom horizontal corridor. Moreover, the agent has to drive around rather quickly, so

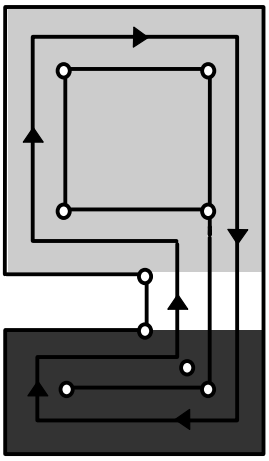

Figure 6.1: Environment and forced driving direction

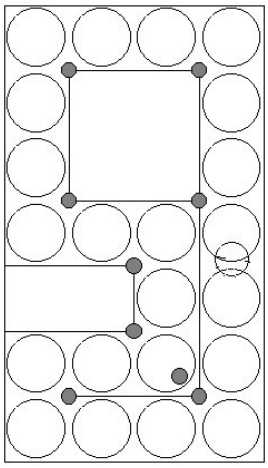

Figure 6.2: Zones in the environment

it can not stop in a location of the environment which is easy to recognise. One additional difficulty is that the agent has to drive in a narrow corridor which limits its possibilities to exploit sensory-motor coordination to solve the perceptual aliasing. In other words, it cannot adopt a strategy in which it drives in a peculiar way to be able to differentiate between the two rooms. Experiments performed with reactive agents did not lead to successful individuals [23].

So the agents need to use their internal state to solve the perceptual aliasing and the sensory signals have to be exploited to cause changes in the internal state and ultimately, in the self-localisation neuron.

We repeat part of the experiments done in [23] with all recurrent neural network types. The fitness is determined by the same function as in [23]:

$$F = \begin{cases} \frac{z_c}{z_t} & \text{, if } z_c < z_t \\ 1 + (bottom * top) & \text{, if } z_c >= z_t \end{cases} \tag{6.1}$$

In which z_c is the number of zones crossed and z_t is the zone threshold. *bottom* and *top* are the percentages of good localisations in the bottom and top room respectively. The maximum fitness is 2.

The agent is executed for multiple epochs. An agent starts an epoch at a specific position in the environment and drives for a fixed amount of time steps. In the original evolution every agent performed the task for eight epochs of 2500 time steps and $z_t = 1000$. The threshold is used in the first part of the evolutionary process to force the agent to drive around in the environment, see figure 6.2. The self-localisation output of the agent is measured continuously when it is inside one of the two rooms, but is only considered if the agent crosses the driving threshold during the epochs that it is executed. For our experiments, we use four epochs and we use a driving threshold expressed in the number of

rounds an agent has to drive. A round in the environment consists of 22 zones.

We have executed three experiments[1] per type of agent at three different driving thresholds. For the experiments the agents have to perform the task at a driving threshold of 20, 23, and 25 rounds, corresponding to $z_t = 440$, $z_t = 506$, and $z_t = 550$, respectively. As a result, the agents drive slightly slower or faster than in the original experiment. As in the original experiments, the different networks have the activations of the motors of last time step as inputs to the network.

6.2 Results

We first discuss the results of the experiments performed at the lowest driving threshold. Then we relate the results for the experiment at the lowest driving threshold to the experiments at the higher driving thresholds. In sections 6.2.1 to 6.2.3, we discuss the three mechanisms and the strategies of the corresponding agents for that task at different driving thresholds.

The results of the experiments are expressed in two performance measures: $\overline{F}_{best}$ and $\#r, \overline{F}_r > 1.56$. $\overline{F}_{best}$ is the average fitness of the best agent of all evolutionary runs for a specific recurrent neural controller. $\#r, \overline{F}_r > 1.56$ indicates the number of runs of the evolutionary algorithm that resulted in an agent that has an average fitness higher than 1,56. A fitness of 1,56 implies that the agent is able to self-localise correctly for 75% of the time. To obtain the results, we have executed the best agent of every evolutionary run for 100 epochs. Table 6.1 shows the results of the experiments with a driving threshold of 20 rounds.

	RNN	NARX	RNND	CTRNN	TDRNN
Hiddens	5	5	5	5	5
Type	-	$c_{in} = 1$, $c_{out} = 4$	I, H	I, H	-
Rounds	20	20	20	20	20
$\overline{F}_{best}$	1.85	1.75	1.95	1.75	1.68
$\#r, \overline{F}_r > 1.56$	2	4	7	4	3

Table 6.1: Results of the self-localisation experiment

Since $\overline{F}_{best} > 1.56$ for all recurrent neural controllers, they are all able to solve the task at the driving threshold of 20 rounds. All controllers use the recurrency of the internal state to escape the perceptual aliasing in the environment.

However, the different types of agents differ in their strategies to change their internal state when the agent changes room. Table 6.2 is a copy of table 4.2 and shows what mechanisms form the basis of the strategies to change the localisation of the different types of agents. The agents with controllers depending only on recurrency, i.e., the RNN-agents and the NARX-agents, change their internal state by means of sensory-motor coordination. By moving

[1] Each experiment consists of 10 evolutionary runs.

in a particular fashion, they recognise when they enter another room and they change the internal state accordingly. On the contrary, agents with controllers that depend on either neural inertia or varying time delays do not apply sensory motor coordination to notice the change of room. They use the mechanisms of neural inertia or varying time delays for exploiting a long-term regularity in the agent-environment system to change their localisation output.

When the agents have to attain the driving threshold of 25 rounds, the agents basing their strategy on the mechanism of recurrency are not able to perform the self-localisation task anymore. Higher driving thresholds, i.e. higher speeds, restrict the possible movements of the agent in the environment. Therefore, they restrict the possibility to exploit sensory-motor coordination. Table 6.3 shows at what driving thresholds the different neural controllers are still able to perform well on the localisation task. A successful performance of at least one evolved agent ($\overline{F}_{best} > 1.56$) is indicated by the label 'S'. A failure ($\overline{F}_{best} <= 1.56$) is indicated by the label 'F'. Both the RNN-agents and the NARX-agents base their strategy on the mechanism of recurrency (see table 6.2) and are not able to perform the task for all driving thresholds.

Neural controller	Recurrency	Neural inertia	Varying time delays
RNN	B	-	-
NARX	B	-	X
RNND	B	B	-
CTRNN	B	B	-
TDRNN	B	-	B

Table 6.2: Mechanisms that form the basis of the strategies of the neural controllers

Neural controller	20 rounds	23 rounds	25 rounds
RNN	S	F	F
NARX	S	S	F
RNND	S	S	S
CTRNN	S	S	S
TDRNN	S	S	S

Table 6.3: The driving thresholds for which the different controllers are able to perform well on the localisation task, i.e. for which $\overline{F}_{best} > 1.56$

In sections 6.2.1 to 6.2.3, we discuss the three mechanisms and the strategies of the corresponding agents. In section 6.2.1 we demonstrate that the agents relying on the mechanism of recurrency only, viz. the RNN-agents and the NARX-agents, change their internal state by means of sensory-motor coordination. In section 6.2.2 we show that the agents also employing the mechanism of neural inertia, viz. the RNND-agents and the CTRNN-agents, change their internal state based on a long-term regularity in the agent-environment system. Finally, in section 6.2.3 we demonstrate that the agents relying on the mecha-

nism of varying time delays, viz. the TDRNN-agents, also change their internal state based on a long-term regularity in the agent-environment system.

6.2.1 Recurrency

We first discuss the strategies of the agents that depend only on recurrency. The RNN- and NARX-agents apply similar strategies to solve the self-localisation task. They use their recurrent connections to form equilibrium points of a subset of their neurons to indicate the room in which the agent is situated. In addition, they apply sensory-motor coordination to recognise the locations where the internal state has to be changed. We first discuss the strategies of the RNN-agents and then the strategies of the NARX-agents.

RNN

We demonstrate that the RNN-agents use recurrency to solve the perceptual aliasing. In addition, we show that they use sensory-motor coordination to change the internal state. We analyse the best RNN-agent. It has the second-best average fitness for the task at a driving threshold of 20 rounds; 1.85 (table 6.1). Its performance is surprising, since in [23] RNN-agents were not able to perform the task. The RNN-agent we analyse only makes mistakes in the transition areas between the two rooms.

The agent uses recurrency in the hidden layer to solve the perceptual aliasing, since there are two equilibrium points in the internal state between which the agent switches. These are, on average[2], $(H0, H2, H4) = (0.95, 0.02, 0.07)$ and $(H0, H2, H4) = (0.03, 1.00, 0.84)$. The changes between the equilibrium points are very sudden, usually within a few time steps. The activity plot of the agent in figure 6.4 shows these quick changes of equilibrium. From top to bottom, this activity plot shows the following neural activations. The top two grey plots represent the activations of the motor outputs and the third one represents the activation of the self-localisation output. The black plots that follow represent the hidden units. They are followed by the input neurons in grey that have the values of the two motor outputs of last time step. After that, there are eight infrared inputs, also shown in grey. The bottom black plot represents the performance of localisation. A localisation of 0.5 means that the area between the rooms is crossed, while a 1.0 and 0.0 correspond to a correct and a wrong localisation, respectively.

The agent uses sensory-motor coordination to achieve changes in the internal state. We show that the behaviour of the agent in the environment serves to obtain specific sensory information that indicates when the agent changes room.

The RNN-agent does not change the interval of its localisation output on time, which is shown by the error graph in figure 6.4 and by the grey parts of the trajectory in figure 6.3. A black trace indicates good self-localisation. Grey indicates a mistake or an area in which the localisation is not measured. The areas in which the agent structurally makes errors are indicated by dotted circles. The screenshot of the trajectory of the agent has been taken at the time

[2] Averages based on 100 time steps

step at the end of the activity plot. The numbers from 1 to 5 on top of the activity plot indicate the similarly-labelled locations in the trajectory in figure 6.3.

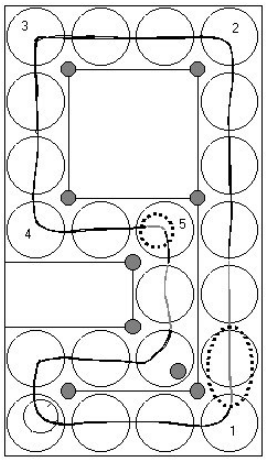

Figure 6.3: Trajectory of the RNN-agent

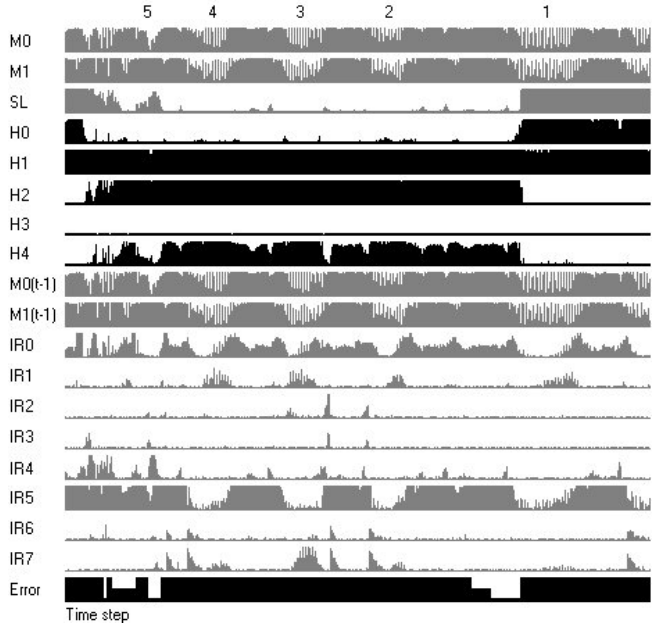

Figure 6.4: Activity plot of the RNN-agent

The localisation error of the RNN-agent is due to its use of sensory-motor coordination to recognise the bottom room. Evolution did not find a solution to recognise the middle of the hall way and the correct output is given only after the agent encounters the corner in the bottom right of the environment. The agent always slightly approaches the right wall in a corridor. At the end of the long corridor the agent is a little closer to the right wall than in the other corners. As a result, the agent negotiates this corner in a different way than the other ones. Consequentially, the two infrared sensors on the back ($IR6$ and $IR7$) are almost not activated when it turns around the corner. Figure 6.4 shows the difference between the activations of $IR6$ and $IR7$ for the turns to the right, labelled '4','3','2' and '1'. The turn labelled '1' is the turn associated with the bottom right corner in which the back sensors are not activated. The placement of the IR-sensors is equal to that of the ambient light sensors in the nest-finding task. Figure 6.5 is a duplication of figure 4.2 and shows the placement of these sensors. To test if the back-sensors IR6 and IR7 are crucial to recognise the bottom room, we fixed their neural activations to 0.05 for the whole epoch. The sensor readings in all turns then resemble those in the bottom right corner. As a result, the agent always indicates that it is in the bottom room.

Sensory-motor coordination is also used to change the internal state to indicate the top room. The agent can perceive the change to the top room by

means of the sensors $IR3$ and $IR4$. They are activated by the object in the hallway and by the movements towards the wall on the right of the agent. The activation of $IR3$ and $IR4$ leads to a localisation output corresponding to the top room, since they inhibit the first hidden neuron.

The use of sensory-motor coordination also leads to the error in the transition to the top room. The cause of this is that the agent is still in the bottom room when it senses the object in the hall. To prevent the agent to immediately indicate the top room, both $IR3$ and $IR4$ inhibit $H4$ whose activation is associated with the top room. However, in the turn to the left in the top room $IR4$ is again high. The result is an error just after the transition to the top room, indicated by the dotted circle in figure 6.3.

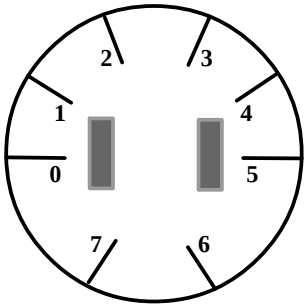

Figure 6.5: Placement of the infrared sensors on a Kephera

The other agent with a reasonable performance follows a similar strategy. It makes the same mistake in the bottom right corner. There is a slightly different error in the transition from the bottom to the top room, since it uses other sensors to notice this transition.

Although the performance is not perfect, the results obtained disagree with those reported in [23] in which RNN-agents were not able to perform the task at all. Hence it would be interesting to see the effects of a driving threshold of 23 rounds, or 506 zones. This higher driving threshold is almost equal to the one used in [23] and results in a higher speed of the agent. The information about the experiment is shown in table 6.4. 'Rounds' indicates of how many rounds the driving threshold consists. Instead of the normal ten times, we restarted the evolutionary algorithm twenty times with a randomly-initialised population.

	RNN
Hiddens	5
Rounds	23
$\overline{F}_{best}$	1.55
$\#r, \overline{F}_r >= 1.56$	0

Table 6.4: Results of the RNN-agent at higher speed

The fitness drops drastically at the higher speed. Changing the weight range to $w_{ji} \in [-10, 10]$ or the number of hidden neurons to 10 did not result

in successful individuals. These results do agree with those in [23]. Forcing the agent to drive faster limit the agent's possibilities to use sensory-motor coordination to solve the task. It has less time to recognise specific corners, so it cannot recognise the turn in the bottom right of the environment in the way it did at a lower speed. As a consequence, it is not able to perform the task anymore.

NARX

Like the RNN-agents, the NARX-agents use sensory-motor coordination to change the internal state. Since an obvious influence on the performance of the NARX-network is the capacity of the input- and output-memory, multiple experiments have been performed to establish the contribution of both memories. Table 6.5 displays the results of these experiments. Averages are based on 100 epochs.

	NARX	NARX	NARX	NARX
Hiddens	5	5	5	5
Type	in 0, out 5	$c_{in} = 1$, $c_{out} = 4$	$c_{in} = 3$, $c_{out} = 2$	$c_{in} = 4$, $c_{out} = 1$
Rounds	20	20	20	20
$\overline{F}_{best}$	1.35	1.75	1.61	1.52

Table 6.5: Results of NARX-agents

We do not give an in-depth analysis of the best agent evolved with a driving threshold of 20 rounds, since its strategy is in principle the same as that of the RNN-agent. Also the NARX-controller does not move slowly towards equilibrium points. One of the reasons for this is the fact that both neural controllers have the same activation function. The only difference between the agents concerns the error when going from the bottom to the top room. The NARX-agent makes the permanent error in the bottom room, instead of the top one. So exactly the opposite choice has been made by the evolutionary algorithm, accepting the error in the bottom room and not in the top one.

As the following results show, NARX-agents are also able to self-localise when they are evolved at the higher driving threshold of 23 rounds. The results are displayed in table 6.6.

	NARX	NARX	NARX	NARX
Hiddens	5	5	5	5
Type	$c_{in} = 0$, $c_{out} = 5$	$c_{in} = 1$, $c_{out} = 4$	$c_{in} = 3$, $c_{out} = 2$	$c_{in} = 4$, $c_{out} = 1$
Rounds	23	23	23	23
$\overline{F}_{best}$	1.72	1.80	1.83	1.61

Table 6.6: Results of NARX-agents, $z_t = 506$

The values of $\overline{F}_{best}$ in table 6.5 and 6.6 show that at this higher speed all NARX-agents perform even better than at the lower speed. The best agents

use another strategy than the agents evolved at a lower speed. The reason for this might be that the NARX-agents evolved at the lower speed threshold, converged to simpler strategies that constituted local minima in the search space. Even though the agents are able to perform the task at this higher speed, the explanation of their strategy to change the internal state clarifies that it still depends on sensory-motor coordination.

To illustrate that the NARX-agents use recurrency to solve the perceptual aliasing and use sensory-motor coordination to change their internal state, we analyse the agent with an input-memory of 1 and an output-memory of 4. It attained the highest fitness during evolution. We discuss both the transition to the top and to the bottom room. Figure 6.6 displays the trajectory of the agent. The areas in which it changes its localisation are indicated by a circle and a rectangle. The figure shows that the agent makes structural errors at the beginning and the end of the top room. Figure 6.7 is the activity plot of the agent in the environment. It displays the neural activations, except for the input and output memory, during the two transitions. The first vertical black line has been drawn at the time step where the agent changes its third output unit to indicate the bottom room. The second line has been drawn at the time step where the agent starts indicating the top room. The circles and the squares in the activity plot highlight some of the important sensors for the change of interval of the third output neuron.

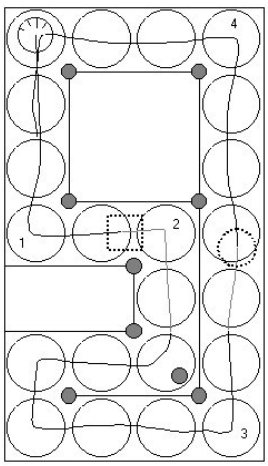

Figure 6.6: Trajectory of the NARX-agent

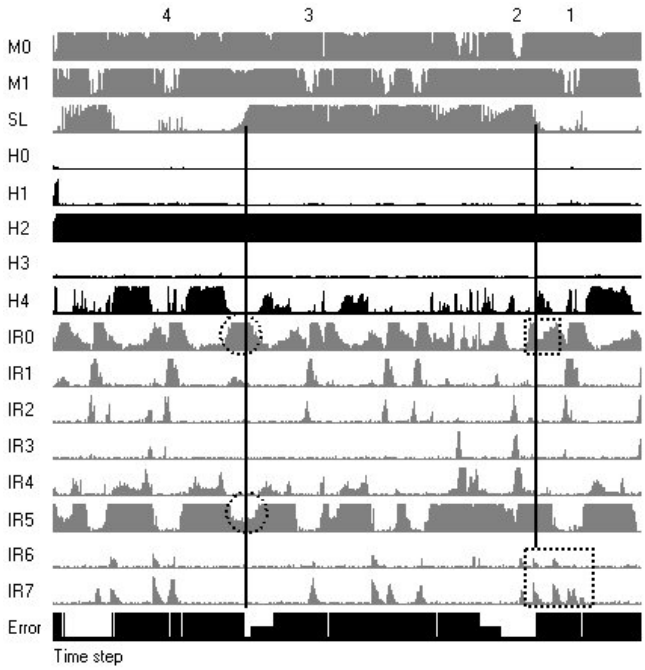

Figure 6.7: Activity plot of the NARX-agent

The NARX-agent uses recurrency to solve the perceptual aliasing in the

environment. None of the hidden neurons is used to indicate the current room. Figure 6.7 demonstrates that there are no hidden neurons whose activations are different between the vertical lines and outside of them. Instead, the third output neuron maintains its own activation by having two large positive weights amongst its 4 self-connections. It depends on the inputs at what equilibrium point it is situated: at 0.0 or at 1.0.

The NARX-agent uses its movements in the environment to obtain unambiguous sensory signals about where the transitions between rooms are situated, i.e. its strategy to change the localisation output is based on sensory-motor coordination.

The agent uses sensory-motor coordination to change the activation of the self-localisation neuron from $[0.0, 0.5]$ to $\langle 0.5, 1.0]$ in the middle of the long corridor. The reason for this is that the agent approaches the left wall in every corridor. Normally, when the agent is close to the wall, it is situated in a turn such as the one labelled '4'. However, there is one time that the agent is not situated in a turn; in the middle of the long corridor. The moment in which the agent comes close to the wall in the long corridor, is indicated by the dotted circles in figures 6.6 and 6.7. This has an effect on the sensors: $IR0$ augments in activation, $IR5$ decreases and $IR2$, $IR3$, and $IR7$ are not activated as is the case in a turn. As can be deduced from the weights shown in table 6.7, this all helps to activate the localisation output. Other well performing agents with the same or another type of in- and output memory follow similar strategies, although in some cases the hidden neurons are used to indicate the room.

Because of the use of sensory-motor coordination, the agent localises the top room too late. The NARX-agent uses sensory signals that are specific of the transition to the top room to change its internal state. The sensors involved are $IR6$, $IR7$ and $IR0$. $IR6$ and $IR7$ are activated after the second turn to the left, which is indicated in figure 6.7 with the dotted box. $IR0$ has a decrease of activity in a turn to the left. Table 6.7 contains the weights of the three infrared neurons to the self-localisation neuron. To prove that these sensors are crucial to the strategy to indicate the top room, we set $IR6$ and $IR7$ to 0.05 and provided $IR0$ a minimum activation of 0.25. As a result, the self-localisation output always indicates the bottom room. The particular combination of signals that causes the agent to change its localisation occurs too late, when the agent has already entered the top room.

From / To	SL
IR 0	1.25
IR 2	-2.46
IR 3	-0.43
IR 5	-3.36
IR 6	-0.7
IR 7	-4.18

Table 6.7: Weights from the input memory to the self-localisation output

If the agent has to drive faster, then its possibilities to use sensory-motor

coordination become even smaller: the agent cannot exploit special movements to recognise the transition to the bottom room on a short time scale. No good NARX-agents evolved at a speed threshold of 25 rounds, the highest average fitness being 1.56. The trajectory of the NARX-agent in that experiment is smoother. Because of the higher required speed, it cannot 'wiggle' anymore in the corridor.

The NARX-agents can still perform well at the driving threshold of 23 rounds, while at the same threshold no successful RNN-agents are evolved. There are a number of differences between NARX-networks and RNNs that could explain the difference in their success. However, an RNN-agent might be able to use the same strategy as the NARX-agent, because the strategy of the NARX-agents is based on sensory-motor coordination, and the changes in the internal state happen on a short time scale. The NARX-network at least seems to ease the search for a successful controller, since almost all runs used by the evolutionary algorithm result in good individuals for the driving threshold of 23 rounds.

6.2.2 Neural inertia

The agents discussed in 6.2.1 use sensory-motor coordination to recognise when the agent enters a room. The strategy of the agents with neural inertia depends less on sensory-motor coordination. Instead of using sensory-motor coordination to receive unambiguous signals from the environment, the internal state is used to exploit a long-term regularity in the environment. In particular, the best RNND- and CTRNN-agents exploit the fact that in the long corridor the sensory inputs are very similar for an extensive period of time. In this section we explain the role of neural inertia for the strategies of both the RNND- and the CTRNN-agents.

RNND

The RNND-agents use recurrency and neural inertia to solve the perceptual aliasing. They use neural inertia to change the internal state when the agent enters another room. We analyse the agent with the highest average fitness. It uses its recurrent hidden layer to localise itself. The first hidden neuron strongly inhibits the self-localisation neuron. Its activation is 1.0 in the top room and 0.0 in the bottom room. Figure 6.8 and figure 6.9 are respectively the trajectory and the activity plot of the agent when it enters the bottom room. The numbers on top of the activity plot are related to the corners encountered and they form a link between the two figures.

We demonstrate that the RNND-agent uses the sensory inputs, recurrency and neural inertia to change the localisation output. By doing so, the RNND-agent is able to exploit regularities on a long time scale. We first discuss how the RNND-agent uses the neural inertia to notice that it is traversing the long corridor. Then we explain the agent's strategy concerning the transition to the top room.

Neural inertia forms the basis of the strategy of the RNND-agent to notice

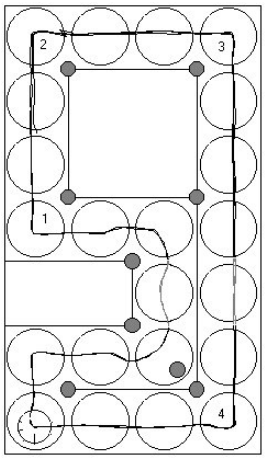

Figure 6.8: Trajectory of the RNND-agent

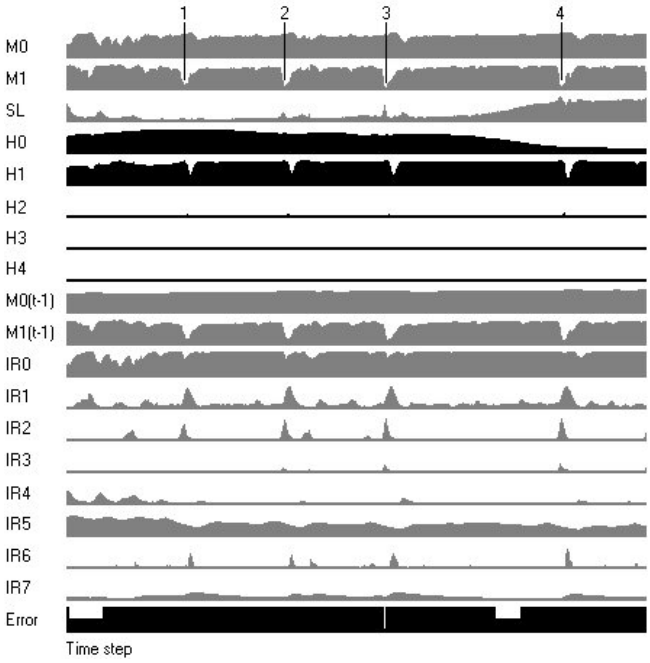

Figure 6.9: Activity plot of the RNND-agent showing the transition to the bottom room

that the agent enters the bottom room. If the agent turns to the right, it excites $H0$. In a corridor the high neural inertia of $H0$ ($d_{H0} = 0.96$) results in a slow movement towards an equilibrium point of 0.0. Everywhere in the top room the agent encounters a corner before the neuron can reach the equilibrium point. Only in the long corridor the related sensory inputs last long enough for the activation to reach the equilibrium point of 0.0. Figure 6.9 illustrates this, since the activation of $H0$ decreases between the time steps labelled '3' and '4'. The other successful agents also use the neural inertia to make a smooth transition from the top to the bottom room. The back sensors sometimes have a high delta as well to support the strategy.

The RNND-agent uses recurrency to prevent the agent from wrong localisations in the turns to the right in the bottom room. The self-connection of $H0$ has a large positive weight. As a consequence, the equilibrium value of $H0$ equals 1.0 in a turn to the right, only if $H0$ is already activated. In the bottom room $H0 = 0.0$ and therefore the turns to the right do not activate $H0$. In figure 6.9, the sensory inputs at the time step labelled '4' resemble those at '1', '2' and '3'. However, the activation of $H0$ remains low.

For the transition to the top room, the agent uses sensory signals that are specific to turns to the left. Figure 6.10 displays the trajectory of the agent and the position of the agent just after it has entered the top room. Figure 6.11 is the activity plot of the agent. The labels '1' and '2' in the figures are

placed at the time steps that the agent makes a turn to the left. The sensory inputs $IR0$, $IR4$ and $IR5$, excite the first hidden neuron in turns to the left. $IR0$ is the input to the left side of the Kephera. The figures 6.11 and 6.10 show that in a turn to the left $IR0$ has a low value since there is no wall in its immediate surroundings. Since $w_{IR0H0} = -1.99$, it inhibits $H0$ less in turns to the left. In addition, the turn to the left has its influence on the sensors $IR4$ and $IR5$ because of objects to the front-right during the turn and because of the movement towards the wall afterwards. Their activities increase, while $w_{IR4H0} = 2.89$ and $w_{IR5H0} = 2.46$. When these two sensors are fixed to their average values in the rest of the environment, the agent does not immediately localise itself correctly in the top room.

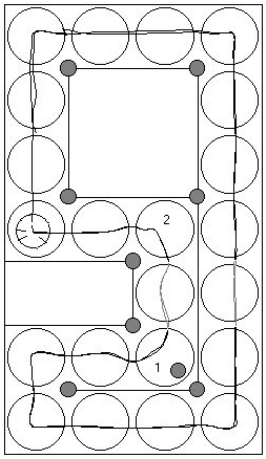

Figure 6.10: Trajectory of the RNND-agent

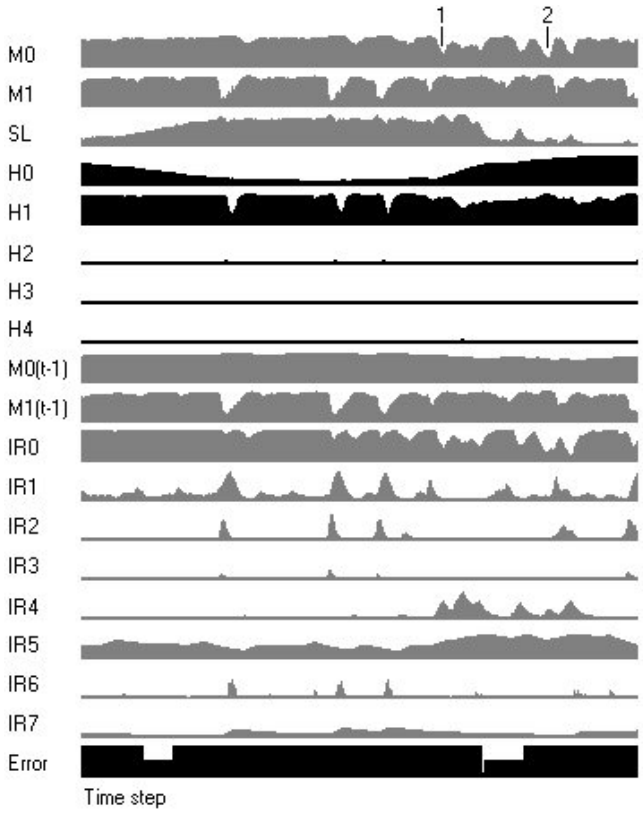

Figure 6.11: Activity plot of the RNND-agent showing the transition to the top room

The strategy of the RNND-agent is based on the neural inertia of the hidden neurons. Since the neural inertia can be adjusted by the evolutionary algorithm to different time scales, higher speeds do not hamper the performance of RNND-agents. RNND-agents still perform well at a driving threshold of 25 rounds, applying the same basic strategy as the analysed agent with a different d-value to match the different time scale on which the internal state has to change in the long corridor.

CTRNN

CTRNN-agents apply almost the same strategies as RNND-agents. At a speed threshold of 25 rounds ($z_t = 550$), the CTRNN still performs well on the task. It functions even better than at a driving threshold of 20 rounds: the best performance is 1.91 and the average fitness is 1.82 (based on 100 runs).

We demonstrate that the strategy used by the best agent is very similar to that of the RNND-agent. The activity plot and trajectory of the agent in figure 6.13 and figure 6.12 respectively, clarify the similarity. In figure 6.12 a circle indicates the error the agent structurally makes.

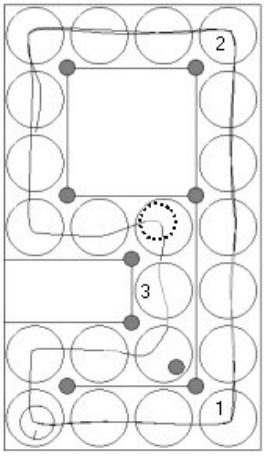

Figure 6.12: Trajectory of the CTRNN-agent

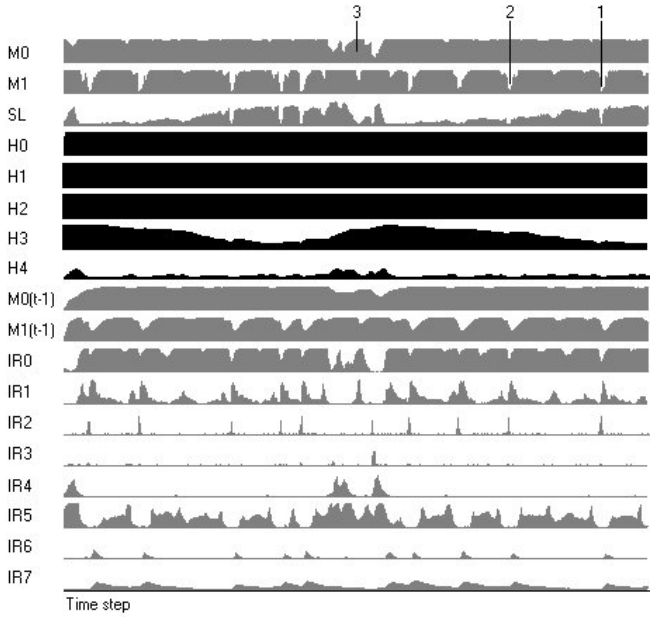

Figure 6.13: Activity plot of the CTRNN-agent

As in the case of the RNND-agent, neural inertia is used to localise the bottom room. A hidden neuron, $H3$, is used that gets excited by the sensory inputs related to turns to the right. When no turns to the right are encountered, the activation of the neuron drops. In figure 6.13 the activation of $H3$ decreases to 0.0 between the upper right and bottom right corner, labelled respectively '2' and '1'.

The way of the CTRNN-agent to change the localisation to the top room has similarities with the RNND-agent. The CTRNN-agent also uses signals that are specific for the two turns to the left to change the localisation output. In the turns to the left, $M0(t-1)$ and $IR0$ decrease in activation. Since they normally inhibit $H3$, it gains activation. The only difference with the RNND-agent is the structural error in the transition from the bottom to the top room.

The error of the CTRNN-agent is similar to the error of the RNN-agent. To prevent the agent to indicate the top room too early, a direct connection between the sensors and the localisation output is used. $IR4$ has a strong

connection with the localisation output, so that the localisation does not change immediately together with $H3$. However, when $IR4$ has a second peak in the second turn to the left, it excites the localisation output which then indicates the wrong room. In figure 6.13 $IR4$ and SL have a peak at the same time in the second turn to the left, just after the time step labelled '3'. It is remarkable that a direct excitation of the localisation output by sensory neurons is used, while $H3$ only slowly augments to indicate the top room.

Other agents followed a similar strategy to the agent we have discussed. One agent was able to perform a smooth transition to the top room. However, it made errors in the transition to the bottom room.

6.2.3 Varying time delays

The TDRNN-agents use strategies that are less dependent on sensory-motor coordination than the strategies of the NARX- and RNN-agents. They do not make special movements to obtain unambiguous sensory information about the location of the agent. Instead, evolution selects the delays on the connections as to exploit long-term regularities of the agent's behaviour. We will demonstrate this by explaining the strategy of the best TDRNN-agent.

TDRNN

We demonstrate that the TDRNN-agents use recurrency and varying time delays to recognise the transitions between the rooms. The adaptable delays enable the TDRNN-agents to exploit long-term regularities in the environment. The performance of the best TDRNN-agent at a speed threshold of 20 rounds is just sufficient. However, the performance of the restricted form of the TDRNN at a speed threshold of 25 rounds is more than reasonable. The average fitness based on 100 runs of the best agent is 1.80 and the highest fitness obtained during evolution was 1.85.

To explain the strategy of the TDRNN-agent, we focus on the hidden neurons. The reason for this is that they determine almost solely the activation of the self-localisation neuron. Figure 6.14 shows the trajectory of the agent and figure 6.15 is its activity plot. The vertical lines represent the areas in between the two rooms. The horizontal line is the line of the equation $SL = 0.5$. The right end of the activity plot corresponds to where the agent is in figure 6.14. There is a certain period in which the activations of the hidden neurons 2 and 3 are very high. The delay from the hidden layer to the self-localisation output neuron is 35. As a consequence, the localisation neuron receives the high activations from the hidden layer almost only when the agent is in the top room. The hidden layer inhibits the localisation output very strongly. In contrast, the weights of the input neurons towards the localisation output are quite small.

We demonstrate that the TDRNN-agent uses information that might be considered ambiguous to determine its location: all turns in the top and the bottom room lead to an indication of the top room. We discuss the two factors that prevent wrong localisations in the bottom room. First we explain why the bottom right corner does not result in a localisation of the top room. Second,

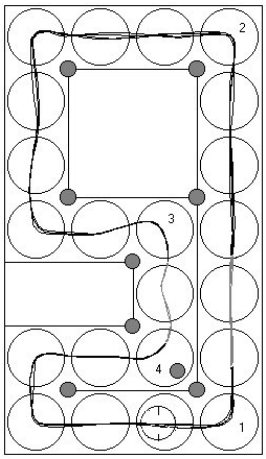

Figure 6.14: Trajectory of the TDRNN-agent

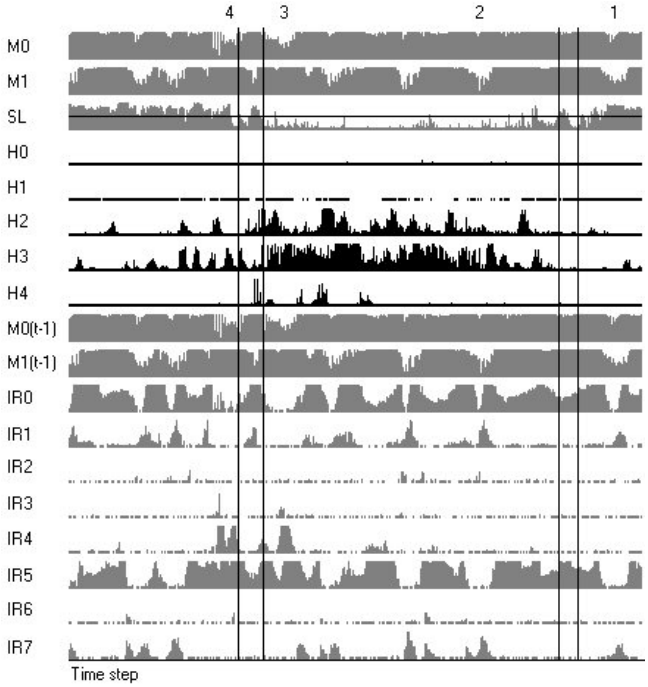

Figure 6.15: Activity plot of the TDRNN-agent

we show that evolution adapted the delays on the connections to a suitable time scale on which sensory information has its effect on the localisation output. The delays have been adjusted to the speed of the agent in the environment, so that the excitation of the turns in the bottom room are adequately delayed until the agent enters the top room. The sensory information related to turning is ambiguous, since both rooms contain turns. But the agent uses its internal state to disambiguate this information.

All turns in the top and the bottom room lead to an indication of the top room. The turns to the left or right excite the third and fourth hidden neuron that strongly inhibit the localisation output. Both $H2$ as $H3$ are stimulated by $IR7$, the sensor on the left back of the Kephera. Figure 6.15 shows that the activation of $IR7$ increases during turns to the right. $IR1$ is usually active in the same time and has a large excitatory connection with $H3$. $H2$ is inhibited very strongly by $M0(t-1)$ and $IR0$. These two neurons decrease in activity and inhibit $H2$ less during turns to the left.

There are two causes that prevent the TDRNN-agent to localise itself wrongly in the bottom room.

The first cause is the use of sensory-motor coordination regarding the bottom right corner and the limited use of recurrency. By using sensory-motor coordination, the corner in the bottom right of the environment does not stimulate the hidden neurons as much as in other turns. In the bottom right corner the activation of IR7 and IR1 are less high during the turn. So the effect of

these activations is smaller than in other curves. In addition, the recurrency of the hidden layer is not large enough to sustain the activity in the hidden layer in the long corridor. Only when multiple turns are encountered after each other, the hidden neurons start to retain a certain activity high enough to have its effects on each other and on the localisation output. Consequentially, encountering the bottom corner does not result in a low activation of the localisation output.

The second cause is that the TDRNN-agent uses its delays to avoid wrong localisations caused by the turns in the bottom left of the environment. When the agent encounters those turns, it is already close enough to the top room to use its delays to delay the activation of the hidden layer until the agent enters the top room. Table 6.8 shows the delays involved. In the turn to the right in the bottom left of the environment, $H3$ is excited after 5 timesteps. This does not lead to a total decrease to 0.0 of the self-localisation output. This decrease only comes when the signals caused by the turn have travelled from the input layer to $H2$ and then to SL, so after 63 time steps. At almost the same time, part of the signals coming from the second turn reaches the output, those going through $H3$. But the hidden neurons are most excited due to the turns to the left, labelled '4' and '3' in figures 6.15 and 6.14. The left motor slows down, which activates $H2$ which in turn activates $H3$. After the turns to the left, the recurrent connections and the new turns encountered keep the activations of the hidden neurons high and the self-localisation output low. Despite the large positive connections in the recurrency of the hidden layer, the activations decrease when the agent is travelling along the long corridor. The other agents show strategies very similar to the discussed one, but they are less successful.

From / To	H2	H3	SL
Inputs	28	5	45
Hidden neurons	47	9	35

Table 6.8: Delays on the connections in the network

Although the TDRNN-agents have a reasonable performance on all speeds, they seem to be biased towards certain time scales. The TDRNN-agents perform worse than RNN-agents on the self-localisation task with a driving threshold of 20 rounds (see table 6.1), while setting all delays to 1 would make a TDRNN-agent equal to an RNN-agent. The random initialisation of the populations of TDRNN-agents results in a bias of the agent towards certain speeds. For the driving threshold of 20 rounds we have evolved agents with a maximum time delay of 100 time steps, but it did not result in agents with a higher fitness.

6.3 Discussion

All types of agents use the recurrency of their internal state to localise themselves in the environment. In addition, they are all able to change their internal state. However, the way in which they are able to change it depends on the

mechanisms with which an internal state is realised.

Our main finding for the self-localisation task answers the first research question and the explanation of it answers the second research question. Our main finding is that the agents relying on neural inertia or varying time delays have the ability to exploit regularities on variable time scales, while agents relying only on recurrency do not have this ability. We first discuss why agents applying neural inertia and adaptable delays are able to use short- and long-term regularities in time. Then we discuss why the agents relying solely on the mechanism of recurrency do not have this ability.

The mechanisms of neural inertia and varying time delays offer agents the ability to exploit both short- and long-term regularities in time, since both mechanisms include parameters that determine when perceived sensory signals have an effect on the outputs. The RNND-agents and CTRNN-agents are able to determine the time scale on which sensory inputs have an effect on the outputs, since the activations can move either slowly or quickly towards an equilibrium. As a result, they can exploit either short or long durations of similar sensory inputs. In the localisation task, the agents with neural inertia can use a long-term regularity to determine their location, i.e. the long duration of similar sensory inputs in the long corridor. In TDRNN-agents the adaptable delays determine on what time scale the inputs have an effect on the outputs. As a result, an agent can either respond quickly or slowly to sensory inputs. In the localisation task, the delays are adapted to the time the agent needs to drive from the first corner in the bottom left of the environment to the top room.

The agents relying on recurrency, i.e. the RNN-agents and NARX-agents, are not able to determine the time scale on which inputs have an effect on the outputs. What is the reason of this? Clearly, evolution cannot adjust the delays on the connections of their networks. But theoretically, an RNN or a NARX-network can represent dynamical systems in which the activations slowly evolve towards an equilibrium point. In chapter 3 we explained that the behaviour around the equilibriums of a CTRNN or a RNND can be changed with the time constant or delta respectively. These parameters cause the activations to move more slowly towards equilibriums. We conjuncture that it is easier for evolution to establish slow changes in the internal state of an RNND or a CTRNN than in an RNN or a NARX-network.

By means of the activation functions of the RNND and the RNN, we clarify that the change in activation of a neuron in an RNND is always smaller or equal to the change of activation of a neuron in an RNN, if $d \in [0, 1]$ and their situation[3] is the same. First we express the neural activation function of the RNND (equation 6.3) in terms of the activation function of the RNN (equation 6.2), as shown in equation 6.4.

$$a_{rnn}(t+1) = \sigma(netinput(t+1) + bias + in(t+1)) \tag{6.2}$$

$$a_{rnnd}(t+1) = da(t) + (1-d)\sigma(netinput(t+1) + bias + in(t+1)) \tag{6.3}$$

[3]I.e., the bias weight, neural inputs, and past activations of both types of neurons have the same values.

$$a_{rnnd}(t+1) = da(t) + (1-d)a_{rnn}(t+1) \tag{6.4}$$

If $d \in [0,1]$ we can conclude equation 6.5 and 6.6 from equation 6.4.

$$a_{rnnd}(t+1) \in [min\{a_{rnn}(t+1), a(t)\}, max\{a_{rnn}(t+1), a(t)\}], \tag{6.5}$$

$$|a_{rnnd}(t+1) - a(t)| <= |a_{rnn}(t+1) - a(t)| \quad \Rightarrow \quad |a'_{rnnd}(t)| <= |a'_{rnn}(t)| \tag{6.6}$$

Equation 6.6 implies that the change in activation of a neuron in an RNND is always smaller than that of a neuron in an RNN, for the same neural input, bias, past input, and external input. The internal state of an RNN-agent or a NARX-agent moves quicker towards an equilibrium than that of an RNND-agent. Although the networks can in theory move slowly towards an equilibrium point, all weights of their connections have to be adjusted to establish it. In an RNND adjusting the delta-value suffices.

The RNN-agents and the NARX-agents cannot determine the time scale on which sensory inputs have an effect on the outputs, since they have difficulties to realise a slowly changing internal state. This results in lower agent capabilities, because all sensory inputs have an effect on the internal state and the outputs on a short time scale. As a result, the agent can only change its internal state if it receives unambiguous sensory information that indicates the necessity of the change. The agents obtain such information by employing sensory-motor coordination, which is harder to apply at higher speeds.

The agents with neural inertia or adaptable delays are less dependent on sensory-motor coordination. They are able to change the localisation output in the middle of the corridor by exploiting long-term regularities in time. The results obtained in the self-localisation task agree with those in [23], in which it was demonstrated that the difference in results between the RNN-agents and RNND-agents is due to the ability of the RNND-agents to exploit short- and long-term regularities in time.

Chapter 7

General Discussion

In this thesis, we addressed the following problem statement: *How do the mechanisms that realise an internal state influence the agent capabilities?*

We reiterate some background information. The capabilities of a pro-active agent do not only depend on its reactive abilities, but also on the properties of its internal state. Each type of recurrent neural network has a different internal state. The properties of the internal state depend on the mechanisms used to establish it. We have investigated three such mechanisms: recurrency, neural inertia and varying time delays. We found the relation between the three mechanisms and the capabilities of agents controlled by recurrent neural networks employing them. To solve the problem statement, our study focussed on two research questions:

1. What are the capabilities of the agents controlled by the different types of recurrent neural networks?

2. How are these capabilities related to the mechanisms?

In this chapter, we answer both of these questions as follows.

1. We answer the first question by dividing the agents into two classes with different capabilities: single time-scale agents and variable time-scale agents. Single time-scale agents cannot exploit regularities in sensory information on different time scales, while variable time-scale agents can. If single time-scale agents operate on a short time scale, then they are more dependent on sensory-motor coordination. The results indicate that RNN-agents and NARX-agents are single time-scale agents, while RNND-agents, CTRNN-agents, and TDRNN-agents are variable time-scale agents.

2. The answer to the second question is that the networks that use the mechanisms of neural inertia or varying time delays result in variable time-scale agents. The reason for this is that the recurrent neural networks that use the mechanism of neural inertia or the mechanism of varying time delays (in the form of adaptable delays), have parameters that determine the time scales on which sensory signals affect the outputs of the controller.

The mechanism of recurrency alone is not a sufficient condition to obtain a variable time-scale agent.

The organisation of this chapter is as follows. In section 7.1 and 7.2, we define the two classes of agents. In addition, we support the classification into variable and single time-scale agents with the results from our experiments. Then we discuss the limitations of our research in section 7.3. We discuss the practical implications of our work in section 7.4. Finally, we discuss the relation of our study to other studies in section 7.5.

7.1 Single time-scale agents

In this section, we define and discuss single time-scale agents.

Definition 7.1 *Single time-scale agents are agents that cannot determine the time scale on which sensory signals affect the outputs*

The recurrent neural controllers that apply strategies primarily based on the recurrency of activations lead to single time-scale agents. Below, we explain why we regard RNN-agents and NARX-agents as single time-scale agents. Then we discuss the limitations of single time-scale agents.

The RNN is the best example of a single time-scale controller. Theoretically, an RNN can model a dynamical system that moves slowly towards an equilibrium. However, in all three experiments presented in this thesis, the internal state of the evolved agents changes very quickly between different values. This is the case for the limit cycle used to solve the driving task, but also for the changes of equilibrium point in the nest-finding and self-localisation task. The internal state of an RNN acts as a light switch, due to the sudden changes in the activations of the hidden neurons. As a consequence, a sensory signal always has an effect on both the hidden and the output layer on a very small time scale.

NARX-agents are also single time-scale agents, because in all experiments the neural activations change on short time scales. The delays influence the agent's strategy only in the driving task, allowing dynamics that extend over a slightly longer period of time than those of the RNN-agents. The use of the input and output memory results in a higher performance on the tasks than the RNN-agents, but the dynamics and strategies of both agents are very similar.

What are the limitations to the agent capabilities that result from the quick changes in the internal state? The time scale on which both RNN-agents and NARX-agents operate in the performed experiments is short: the agents cannot slowly change their internal state and thus will only change it when there is an appropriate sensory signal from the environment. Therefore, sensory-motor coordination remains very important for both types of agents and it is impossible for them to exploit regularities in time on other time scales. The main support for this line of reasoning are the strategies applied by the abovementioned recurrent neural controllers for the self-localisation task. The transition of the internal state from the top to the bottom room is either done by recognising

the bottom right corner, or by driving in a peculiar way in the corridor. When a higher speed prohibits the use of these strategies, neither controller is able to perform well on the task. Although 'short' is a relative concept, the two agents can also be called 'short time-scale agents'. Agents that can only operate on 'long' time scales will have different, but still limited capabilities.

7.2 Variable time-scale agents

In this section, we define and discuss variable time-scale agents.

Definition 7.2 *Variable time-scale agents are agents that are able to determine the time scale on which sensory signals affect the outputs*

Agents controlled by recurrent neural networks that use the mechanisms of neural inertia or varying time delays are variable time-scale agents. We first explain why the agents with adaptable neural inertia, the RNND-agents and the CTRNN-agents, are variable time-scale agents. Then we do the same for the agents with adaptable time delays, the TDRNN-agents. Finally, we argue that the variable time-scale agents investigated have biases towards certain time scales.

The RNND-agents and the CTRNN-agents use the mechanism of neural inertia and belong to the class of variable time-scale agents. The neural inertia allows their neural activitions to change at different speeds in time. As a result, the neural activations evolve either quickly or slowly towards equilibriums. In all tasks, the agents possess neurons with high inertia and neurons with low neural inertia. Since an equilibrium point is (partly) determined by the inputs, the neural inertia provides the agent the ability to exploit the short or long duration of similar sensory inputs. The most convincing evidence for this are the strategies that both the RNND-agents and the CTRNN-agents used for the self-localisation task. In that task, a hidden neuron with a high neural inertia exploits the long duration of sensory inputs associated with a corridor to localise the bottom room. The result of the neural inertia is that both neural controllers applied the same strategies in the nest-finding and the self-localisation task.

The TDRNN-agents use the mechanism of varying time delays and belong to the variable time-scale agents. The TDRNN-agent has the same neural activation function as the RNN-agent and the NARX-agent, but the adaptable delays offer the TDRNN the possibility to vary the time scale on which sensory inputs affect the outputs. In the nest-finding task, the TDRNN-agent uses the delays on the neural connections to return to the proper target location. With the help of the delays the neural controller is even able to generate slowly changing activations for the driving task. In the self-localisation task the agent is able to self-localise itself even at higher speeds. Since the internal state can be used to exploit regularities in time on a larger time scale, a TDRNN-agent is less dependent on sensory-motor coordination. For example, in the self-localisation task the TDRNN-agent uses sensory inputs related to the turns in the environment to change its internal state. These signals might be considered ambiguous since they are the same in the bottom and the top room, but the

internal state is able to exploit a regularity in time of the agent's behaviour: the delays are adapted to the time the agent takes to enter the top room after encountering the turns in the bottom room.

The variable time-scale agents discussed have biases towards certain time scales. The TDRNN-agents perform only satisfactory on the slower version of the self-localisation task. In addition, the performance and evolvability for the driving task of RNND-agents and CTRNN-agents are very poor. The variable time-scale agents have a bias towards time scales larger than those on which the investigated single time-scale agents operate.

7.3 Limitations

The method that we have used to determine the two classes of recurrent neural controllers has some limitations that stem from three factors:

1. The limitations inherent to evolutionary robotics
2. The number and types of tasks
3. The number and types of recurrent neural networks

The first limitation we discuss is the limitation inherent to evolutionary robotics. We have chosen to adopt the methodology of evolutionary robotics, in which a controller is evolved artificially. Although we have repeated the experiments for the driving task and the self-localisation task with different parameter settings, no guarantee can be given that other parameter settings would not give different results. Such a result might be, for example, that a NARX-agent is able to self-localise well on higher speeds. Although multiple experiments have been performed and the results have been related to the underlying mechanisms, it is a possibility that cannot be entirely excluded. In addition, the classification of the networks does not necessarily have to be valid for other learning strategies than evolutionary algorithms. It should be noted however, that individuals that learn by adjusting their weights during life time have a different internal state: the weights then are also part of the internal state.

The second limitation concerns the use of only three tasks. It is possible that all three tasks share a property that influences the evolution of the Elman and NARX-agents in such a way that evolved agents only change their internal states on a short time scale. Only one of the three tasks, i.e. the self-localisation task, really demonstrates the need to determine the time scale on which sensory inputs affect the outputs of the agent.

The third limitation concerns the limited number and types of recurrent neural networks. That is why we do not argue that the use of recurrency alone always results in single time-scale agents. The single time-scale agents investigated apply the activation function of the RNN (equation 3.8). Other activation functions might allow slow changes in the internal state and thus lead to variable time-scale agents.

7.4 Practical implications

In principle, the classification of single and variable time-scale agents could be of interest for every application of a recurrent neural network as the controller of a robot. A robot is usually constructed to perform a certain task and it is useful to be able to estimate if a controller is adequate for the task at hand. So the capabilities of recurrent neural controllers are of interest for research in which recurrent neural networks are applied to robotic tasks [12, 4, 18]. We illustrate what kind of questions can be answered by taking the capabilities of single and variable time-scale agents into account. We discuss two experiments.

The first experiment we discuss, involves an agent with a standard recurrent neural network that has to drive around in an environment and has to return to a base station when its (virtual) battery is low [12]. The agent perceives its own battery level and can sense the base station from a distance, since it is marked by a light. The agent also senses if it is on the base station, because the floor in that area is painted black and the robot has a light sensor pointed at the floor. Evolved agents succeed to get back in time to the base station, in order to continue to function. Figure 7.1 is a photograph of the experimental environment and robot, taken from [12].

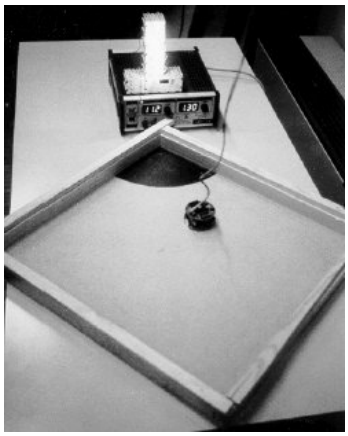

Figure 7.1: Experimental setup in [12]

What would happen if the agent would not be able to perceive its battery level? The other sensory inputs, the infrared sensors, give no information about the battery level, so the agent might have to rely on its internal state to go back to the base station in time. A CTRNN- or RNND-agent will be able to slowly change an activation that triggers homing actions after a certain time. However, we have seen that the internal state of RNN-agents changes on very short time scales only. Therefore, it will not be able to register internally how much time has past since the last visit to the base station. This does not necessarily mean that an RNN-agent or even a reactive agent will not be able to perform the task. Sensory-motor coordination (possibly in combination with the internal state) might help such agents to cope with the absence of the battery sensor.

In [11] Floreano elaborates on the homing experiment by performing some tests to get more insight into the behaviour of the evolved agents. In one test, the light is replaced to the other corner of the environment. The replacement results in a failure of the agent to find the base station. However, the agent has not been evolved to cope with this environmental change. Would an agent with a standard recurrent neural network still be able to perform the task if it would be evolved in an environment in which the light can be in two different corners? The results on short time-scale agents suggest that it is able to do that. If the agent is very close to the light but it does not yet sense the base station, the signal from the environment can be used to change the internal state. The agent can use its internal state to remember that it has to drive away from the light. In this case a short time-scale agent might use its internal state to still perform well on the task.

The second experiment we discuss involves a CTRNN-agent. In [3], Beer states that one of the arguments to use CTRNNs is that they "are arguably the simplest nonlinear, continuous dynamical neural network model". However, the results of the experiments in this thesis seem to suggest that a continuous form of the RNN is a simpler continuous dynamical neural network model than the CTRNN in the sense that it has less capabilities. In [4], Beer develops an agent that has to catch circles and avoid diamonds that fall downwards in a two-dimensional world. He uses this task to exemplify the analysis of the agent's behaviour according to dynamical systems theory. A practical question could be if it is necessary to use a CTRNN to perform the task. Would an RNN- or even a reactive agent not suffice to execute the task? If so, it can be doubted if the internal state of the agent evolved in [4] really contributes to the fitness of the agent. Van Dartel deals with a similar task of active categorical perception in [32], applying both reactive and pro-active agents to the task.

To summarize, the classes of pro-active agents can be used to estimate the possible strategies that an agent can use to solve a certain task. However, it is already notoriously difficult to determine a priori if a task can be solved by a reactive agent or if a pro-active agent is necessary [22]. In fact, the same seems to be true for single and variable time-scale agents. So the necessity of using a variable time-scale agent can only be determined by performing experiments.

7.5 Related work

In the introduction we mentioned that embodied cognitive science favours a bottom-up approach to artificial intelligence. In this approach the step from reactive to pro-active agents represents a step towards increasing complexity. However, pro-active agents do not necessarily apply a completely different task-solving strategy than reactive agents [22]. They still rely partly on sensory-motor coordination to solve a task and often exploit the same properties of the environment as a reactive agent. Nolfi [22] suggests that there is a whole spectrum of pro-active agents that rely on sensory-motor coordination to different extents:

"Between pure reactive agents and pure representational agents a

large variety of intermediate cases exists."

We have tried to gain a deeper understanding of the spectrum of pro-active agents in order to better understand the step from reactive to pro-active agents. In [23], Nolfi already sheds some light on the structure of this spectrum by demonstrating that it consists of agents that are or are not able to *"integrate sensory-motor information over time at different time scales"*. Using slightly different definitions than in [23], the classes introduced in this thesis divide the spectrum of pro-active agents into two groups of agents; agents that are or are not able to determine on what time scale sensory inputs have an effect on their outputs. In addition, we have demonstrated that short time-scale agents are more dependent on sensory-motor coordination than variable time-scale agents. Hence, they are more similar to reactive agents than variable time-scale agents. The class to which an agent belongs, depends on the mechanism its recurrent neural controller employs to establish an internal state. The recurrent neural networks with neural inertia and adaptable time delays studied in this thesis are able to determine on what time scale sensory inputs have an effect on the outputs, while those relying only on recurrency are not.

The spectrum of pro-active agents does not only exist of agents that have recurrent neural connections. According to its definition (section 3.1), an internal state can for example also be effectuated by using a neural network whose weights change in time. The weights then belong to the variables instead of the parameters of the network. Some researches focus on this type of internal state, such as [18, 34, 31].

In [35], Ziemke states the following:

> *" . . . the problem with conventional ANNs is that their connection weights are rather static, with the result that the controlled agent, apart from the influence of the internal feedback, always maps sensory input to motor output the same way. In biological neural networks, however, connection weights are known to fluctuate."*

According to Ziemke, the internal state formed by changing weights does not have the problem that sensory inputs are always mapped to the same motor outputs. In particular, in [34] he introduces self-adapting recurrent networks that combine changing weights with recurrent connections. These networks might indeed have more capabilities than standard recurrent neural networks. However, both past activations as changing weights co-determine the input-output mapping of an agent, i.e. constitute an internal state. In addition, there are many types of internal states in neural networks that have plastic weights. It is still unclear what place in the spectrum of pro-active agents is occupied by those different types and if they offer a better control on the input-output mapping of an agent than the 'conventional ANNs' that we have studied in this thesis.

Chapter 8

Conclusions

We conclude that the answer to the problem statement is, that the mechanisms that realise an internal state in an agent, influence its capabilities by determining whether the agent is a single time-scale agent or a variable time-scale agent. From our results we may draw the following four additional conclusions. First, single time-scale agents cannot use regularities in sensory information on multiple time scales and if they operate on a short time scale, they are more dependent on sensory-motor coordination than variable time-scale agents. Second, the agents whose strategies depend on the recurrency of the connections, the RNN- and NARX-agents, are single time-scale agents. Third, agents using neural inertia or varying time delays are part of the variable time-scale agents. Hence, CTRNN-, RNND-, and TDRNN-agents are variable time-scale agents. Fourth, the reason that the mechanisms of neural inertia and varying time delays in those networks lead to variable time-scale agents, is that they are based on parameters that explicitly determine the time scale on which sensory inputs affect the outputs.

Future research

We hope that the present thesis makes a modest step towards a better understanding of the step up in cognitive complexity from reactive to pro-active agents. However, many more experiments and analyses are needed to gain significant insight in the capabilities of pro-active agents with different internal states. We discern two directions in which future research can develop.

The first direction of future research can involve the verification of the current results and the analysis of other mechanisms realising an internal state. For example, the number of tasks can be extended beyond the three studied in this thesis to verify the validity of our binary classification of the recurrent neural controllers. In addition, other recurrent neural controllers might be included. An example of such a controller is the LSTM-network [14, 1]. Moreover, also agents that implement other mechanisms to obtain an internal state, such as agents with plastic weights, could be studied to determine their capabilities. Furthermore, in this thesis we have dealt with the combination of the mechanisms of neural inertia and varying time delays with the mechanism of recurrency. It would be interesting to study all mechanisms in isolation as

well.

The second direction of further research can involve studying the finer structure of the spectrum of pro-active agents. We could for example determine the differences between TDRNN-agents on one side and CTRNN- and RNND-agents on the other side. And would a TDRNN with neural inertia be even more capable than these neural controllers alone to solve certain tasks?

References

[1] B. Bakker, V. Zhumatiy, G. Gruener, and J. Schmidhuber. A robot that reinforcement-learns to identify and memorize important previous observations. *in Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2003.

[2] R.D. Beer. A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence 72*, pages 173–215, 1995.

[3] R.D. Beer. On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior 3, (4)*, pages 469–509, 1995.

[4] R.D. Beer. The dynamics of active categorical perception in an evolved model agent. *Behavioural and Brain Sciences, submitted*, 2001.

[5] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 1994.

[6] J. Blynel and D. Floreano. Levels of dynamics and adaptive behavior in evolutionary neural controllers. *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 on Adaptive Behavior*, 2002.

[7] R. A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 1990.

[8] G. Bugmann. Biologically plausible neural computation. *Biosystems, 40*, pages 11–19, 1997.

[9] B. Cohen. Training synaptic time delays in a recurrent neural network. *Master of Science Thesis, Tel-Aviv University*, 1994.

[10] J. L. Elman. Finding structure in time. *Cognitive Science 14*, pages 179–211, 1990.

[11] D. Floreano. Ago ergo sum. *Mulhauser, G., editor, Evolving Consciousness*, 1997.

[12] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics, 26(3)*, pages 396–407, 1996.

[13] I. Harvey, P. Husbands, D. Cliff, A. Thompson, and N. Jakobi. Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 1996.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation 9(8)*, pages 1735–1780, 1997.

[15] R. Kortmann. Embodied cognitive science. *W. de Back, T. van der Zant, and L. Zwanepol, editors. Proceedings of Robo Sapiens - the first Dutch symposium on embodied intelligence. Artificial intelligence preprint series vol. 24, Universiteit Utrecht, Utrecht, The Netherlands.*, 2001.

[16] D. Lambrinos, R. Moller, T. Labhart, R. Pfeifer, and R. Wehner. A mobile robot employing insect strategies for navigation. *Robotics and Autonomous Systems, special issue on Biomimetic Robots, Vol. 30*, pages 39–64, 2000.

[17] T. Lin, B. G. Horne, P. Tino, and C. Lee Giles. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks*, 1996.

[18] L. A. Meeden. An incremental approach to intelligent neural network controllers for robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Volume 26, Number 3*, pages 474–485, 1996.

[19] O. Miglino, H. Hautop Lund, and S. Nolfi. Evolving mobile robots in simulated and real environments. *Artificial Life*, 1996.

[20] S. Nolfi. Evorobot 1.1 user manual. *http://gral.ip.rm.cnr.it/evorobot/simulator.html*.

[21] S. Nolfi. Evolving non-trivial behaviors on real robots: A garbage collecting robot. *Robotics and Autonomous System, 22*, pages 187–198, 1997.

[22] S. Nolfi. Power and the limits of reactive agents. *Neurocomputing, 49*, pages 119–145, 2002.

[23] S. Nolfi, G. Baldassare, and D. Marocco. Evolving robots able to self-localize in the environment: The importance of viewing cognition as the result of processes occurring at different time scales. *T. Asakura and K. Murase, editors. Proceedings of the Third International Symposium on Homan and Artificial Intelligence Systems. Fukui, Japan*, 2002.

[24] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines.* Cambridge, MA, MIT Press/Bradford Books, 2000.

[25] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: different approaches in evolutionary robotics. *Proceedings on the Artificial Life IV Conference, July 6-8, Cambridge, MA, U.S.A.*, 1994.

[26] S. Nolfi and D. Marocco. Evolving robots able to integrate sensory-motor information over time. *Biologically Inspired Robot Behavior Engineering, Berlin*, 2002.

[27] S. Nolfi and D. Marocco. Evolving robots able to visually discriminate between objects with different size. *International Journal of Robotics and Automation (17) 4*, pages 163–170, 2002.

[28] R. Pfeifer and C. Scheier. *Understanding Intelligence*. MIT Press, Cambridge, MA, 1999.

[29] K-Team S.A. Kephera robot. *http://www.k-team.com*.

[30] M. Schmitt. On the implications of delay adaptability for learning in pulsed neural networks. *J. Demiris and G. Westermann, editors. Proceedings of the Workshop on Biologically-Inspired Machine Learning*, pages 28–37, 1999.

[31] J. Urzelai and D. Floreano. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolutionary Computation 9(4)*, pages 495–524, 2001.

[32] M.F. van Dartel, I.G. Sprinkhuizen-Kuyper, E.O. Postma, and H.J. van den Herik. Reactive agents and perceptual ambiguity. *submitted for publication in Adaptive Behavior.*, 2003.

[33] B.M. Yamauchi and R.D. Beer. Sequential behavior and learning in evolved dynamical neural networks. *Technical Report CES-93-25*.

[34] T. Ziemke. Towards autonomous robot control via self-adapting recurrent networks. *von der Malsburg, von Seelen, Vorbrueggen, Sendhoff, editors. Artificial Neural Networks - ICANN'96*, 1996.

[35] T. Ziemke. Adaptive behavior in autonomous agents. *Presence - Teleoperators and Virtual Environments, 7(6), special issue on Autonomous Agents*, 1998.

Appendix A: Network implementations

The implementation details of the networks that have not been mentioned in chapter 3, are included in this appendix. The weight range used in all experiments is $w_{ij} \in [-5, 5]$.

RNN

We have implemented the Elman network as described in [10].

RNND

The original activation function introduced in [23] for the hidden and output neurons has the following form:

$$a_i(t+1) = \begin{cases} d_i a_i(t) + (1 - d_i)\sigma\left(netinput_i(t+1) + bias_i\right) & \text{, if } netinput_i(t+1) + bias_i >= 0 \\ 0 & \text{, otherwise} \end{cases} \tag{1}$$

and the activation function for the input neurons is:

$$a_i(t+1) = d_i a_i(t) + (1 - d_i) in_i(t+1) \tag{2}$$

We have used activation function 3.11 in our experiments, since we wanted to focus only on the neural inertia in the RNND. Using activation function 1 for the hidden and output neurons offers the controller the possibility to immediately reset the neuron based on the neural inputs. The possibility to reset a neuron might facilitate the use of a neuron as a type of internal clock. The influence of resetting the activation on the capabilities of an RNND-agent would be interesting to investigate. We did use activation function 2 in our experiments, to prevent the inputs from being too uninformative.

Equilibrium points

The equilibrium points of our implementation of the RNND are independent of the value of d_i. In [3], Beer sets the change in activation potential of the

CTRNN to 0 to find the equilibrium points. This corresponds to setting the activation change to 0.

To demonstrate that the equilibrium points of the RNND are independent of d_i, we look at the activation change of an RNND. The change in activation over time of an RNND can be expressed as follows.

$$a_i(t+1) - a_i(t) = \tag{3}$$
$$d_i a_i(t) + (1-d_i)\sigma(netinput_i(t+1) + bias_i + in_i(t+1)) - a_i(t) =$$
$$(d_i - 1)a_i(t) + (1-d_i)\sigma(netinput_i(t+1) + bias_i + in_i(t+1))$$

For the RNND, the change in activation is zero if:

$$a_i(t+1) - a_i(t) = 0 \tag{4}$$

$$(d_i - 1)a_i(t) + (1-d_i)\sigma(netinput_i(t+1) + bias_i + in_i(t+1)) = 0$$

$$(d_i - 1)a_i(t) = -(1-d_i)\sigma(netinput_i(t+1) + bias_i + in_i(t+1))$$

$$\frac{(d_i - 1)a_i(t)}{-(1-d_i)} = \sigma(netinput_i(t+1) + bias_i + in_i(t+1))$$

Since $(d_i - 1) = -(1 - d_i)$, equation 4 simplifies to:

$$a_i(t) = \sigma(netinput_i(t+1) + bias_i + in_i(t+1)) \tag{5}$$

The equilibrium points are independent of the value of d_i, so that we can choose it to be zero to find the equilibrium points. In section 3.2.2 it is shown that setting d_i to zero and tc_i to one, amounts to the same equation. Therefore, our implementations of the RNND and CTRNN only differ in their behaviour around equilibrium points.

CTRNN

For the implementation of a CTRNN we have made some choices that might be slightly different from those made in other researches. We have chosen to use the values of the inputs and outputs of neurons at time t + 1. In addition, we have determined that $\frac{1}{tc_i} \in [0, 1]$. Next to the domain of the time constant and the method of discretisation we have chosen not to use a gain; a value that

is multiplied with the activation of the input neurons. The activation function of the input neurons with a gain-variable g is as follows.

$$a_i(t+1) = \sigma\left(g\left((1-\frac{1}{tc_i})p_i(t) + \frac{1}{tc_i}(in_i(t+1)) + bias_i\right)\right) \quad (6)$$

Without the gain, the logistic function can make the inputs too 'flat' to be informative. In experiments done with a logistic on the activations but without a gain, the agents have very poor performances.

In evorobot the external input values are already in the interval $[0, 1]$. Therefore, we have chosen to use the following function in our experiments to determine the activations of the input neurons:

$$a_i(t+1) = p_i(t+1) = (1-\frac{1}{tc_i})p_i(t) + \frac{1}{tc_i}in_i(t+1) \quad (7)$$

To cross-check results, we have performed some experiments applying activation function 6. Examples are the experiments for the driving task and the self-localisation task. The application of activation function 6 did not lead to better results. The CTRNN-agent analysed for the driving task applied function 6 with a fixed gain of 1.

In addition, we mention in the result-tables of the three tasks that in the implementation only the input- and hidden neurons use the CTRNN-function to update their activations. The output-neurons do not apply this function, because of the better results obtained. Having neural inertia on the output neurons hampers the reactive capabilities of the pro-active agent controlled by the CTRNN.

NARX

The input memory is only connected to the hidden neurons. So only the current inputs and the output memory are connected directly to the output neurons. This does not influence the jump-ahead connections that are advantageous for learning long-term dependencies.

TDRNN

In our implementation of a TDRNN, there is only one connection between two neurons. As a consequence, there is an initial period in the epoch, that the inputs do not yet reach the outputs. If the time step is smaller than the delay on the connection, the current inputs form the neural inputs to the outputs. So at the beginning of the epoch there is a short time interval in which the delays are ignored.