EXPERIMENTS WITH RESERVOIR COMPUTING ON THE ROAD SIGN PROBLEM

Eric Aislan Antonelo*, Benjamin Schrauwen*, Dirk Stroobandt*

**Electronics and Information Systems Department*
Ghent University
Ghent, Belgium

Emails: `Eric.Antonelo@elis.ugent.be, Benjamin.Schrauwen@elis.ugent.be,`

Abstract— The road sign problem is tackled in this work with Reservoir Computing (RC) networks. These networks are made of a fixed recurrent neural network where only a readout layer is trained. In the road sign problem, an agent has to decide at some point in time which action to take given relevant information gathered in the past. We show that RC can handle simple and complex T-maze tasks (which are a subdomain of the road sign problem).

Keywords— Reservoir Computing, road sign problem, T-maze, long-term memory.

1 Introduction

Most autonomous systems are merely reactive systems. They act according to a predefined sensory-motor mapping which is usually learned and stored in artificial neural networks (bottom-up approach). On the other hand, deliberative systems are usually made of (symbolic) cognitive layers which can deal with reasoning and planning of robot trajectories, for example (the top-down approach). The artificial intelligence community has been raising its attention to deliberative systems which might be built using a bottom-up approach (Rylatt and Czarnecki, 2000; Ziemke and Thieme, 2002). Actually, this new focus of AI is very attractive, since it is known that biological models can cope with very complex problems.

In this work, we follow that *new* approach to artificial intelligence and tackle the road sign problem. In this problem, an artificial agent (robot) which is driving along a corridor receives a temporary sign at some specified time which must be remembered at a later moment in order to take the correct decision (e.g., to turn right or left) (Ulbricht, 1996). The road sign problem is a delayed response task which is usually tackled in the form of a T-maze task (Fig. 2). The T-maze is an environment with a T-shape in which a robot is positioned initially at the end of longest arm (corridor). The goal for the robot is located at the right arm if it has seen a sign (usually) at the same side when driving along the main corridor. Accordingly, the goal is at the left part if the sign was seen at the left side previously. The difficulty of this task relies on the time gap existing between the sign and the decision at the T-junction. The robot has to navigate up to the T-junction and hold the information (the sign) gathered in the past. This is a problem with a temporal nature which can not be handled with traditional feedforward neural networks (Rylatt and Czarnecki, 2000).

Several approaches to the road sign problem have been proposed. In (Kim, 2004), evolutionary multi-objective optimization is used to evolve finite state controllers with two objectives: behavior performance and memory size. Noisy environments are considered in that work and considerable performance degradation is perceived. The work in (Ziemke and Thieme, 2002) is based on neuromodulation of synaptic weights in higher-order Recurrent Neural Networks (RNNs) to solve the T-maze task. This means that the sensory-motor mapping (synapses) can be modified while the robot is navigating as a mechanism of short-term memory. This synaptic plasticity is evolved by a standard genetic algorithm in their work. However, for simple T-mazes the resulting controller became purely reactive and followed the left wall as soon as the light sign appeared at the left. In (Linaker and Jacobsson, 2001), event extraction is used to pre-process the input from the robot in order to facilitate the training of a recurrent neural network with backpropagation through time. In that work, the outputs of the network are three handcrafted behaviors (wall following behaviors) instead of more low-level motor output. The resulting controller works for arbitrary delay periods between the stimulus and the cue for response. Reinforcement learning with Long Short-Term Memory (LSTM) is the approach used in (Bakker, 2002) to solve non-Markovian tasks with long-term dependencies between relevant events (such as the T-maze task). A specific RNN architecture is used to approximate the value function of a reinforcement learning algorithm. The environment of the agent is discrete (made up of connected squares) and it can execute one out of 4 actions: move North, East, South or West.

The current work uses Reservoir Computing (RC) on the road sign problem. RC has been introduced in (Verstraeten et al., 2007) as an unifying term for three other computing techniques independently discovered which share similar characteristics: Echo State Networks (Jaeger and Haas, 2004), Liquid State Machines (Maass et al., 2002), and BackPropagation DeCorrelation (Steil, 2004). RC is a RNN with very efficient and fast learning. This is because the architecture of RC networks consists of a reservoir (the RNN itself) and a readout layer (Fig. 1). The reservoir itself is left fixed (the weights are created randomly at the beginning) whereas the readout layer is trained by standard linear regression methods.

This work aims at showing that RC is very well suited to solve problems with long-term temporal dependencies between relevant events. It is also a first experiment on using RC on the context of adaptive behavior research (as far as we know). The experimental setup is divided in two phases. First, we use an autonomous robot simulator and an existent reactive robot controller (Antonelo et al., 2006) in a T-maze environment for automated generation of a dataset containing samples of the robot sensory inputs and actuator. The second phase consists of training a reservoir in a Matlab environment using the previously generated data. We show that RC is able to solve the T-maze task by learning the relantionship between the sign and the correct turning at the T-junction. Furthermore, experiments with simple and more complex T-mazes are accomplished as well as an analysis of memory requirements for these experiments.

2 Methods

2.1 Reservoir Computing

The current work uses the Echo State Network approach as a learning system for the road sign problem. The random, recurrent neural network (or reservoir) is composed of sigmoidal neurons and is modeled by the following state update equation:

$$\mathbf{x}(t+1) = f(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t)), \qquad (1)$$

where: $\mathbf{W}_{\text{in}}$ is the connection matrix from input to reservoir; $\mathbf{W}$ is the weight matrix for the recurrent connections between internal nodes; f is the hyperbolic tangent function; and $\mathbf{u}(t)$ is the input vector at time t. The initial state is $\mathbf{x}(0) = \mathbf{0}$.

The output $\mathbf{y}(t)$ of the network at time t is given by

$$\mathbf{y}(t) = \mathbf{W}_{\text{out}} \begin{bmatrix} \mathbf{x}(t) \\ 1 \end{bmatrix}, \qquad (2)$$

where $\mathbf{W}_{\text{out}}$ is the readout matrix.

The readout matrix is created by solving (in the mean square sense) the following equation:

$$\mathbf{M}\mathbf{W}_{\text{out}} = \hat{\mathbf{Y}}, \qquad (3)$$

where $\mathbf{M}$ is the matrix containing the internal states $\mathbf{x}(n)$ for $n = 1, 2, \ldots, n_s$ (which are collected after stimulating the network with input

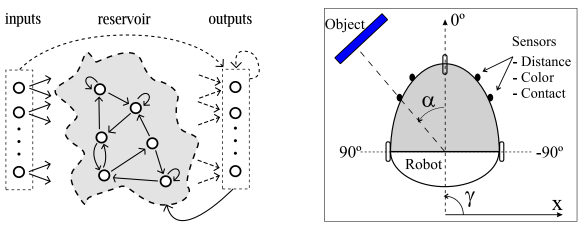

Figure 1: Reservoir Computing network (left) and Robot model (right).

data); $\hat{\mathbf{Y}}$ contains the corresponding teacher outputs; n_s denotes the total number of time samples.

2.2 Robot model

We use an autonomous robot navigation simulator (SINAR) and a particular reactive robot controller for automated generation of the dataset (of sensory inputs and actuator) which is used to train a RC neural network in the T-maze task. The robot simulator and controller are the same as used in (Antonelo et al., 2006). Several environments with a T-maze shape are created with this simulator for the experiments in the following sections (see Fig. 2). Next the environment and robot controller are described briefly.

The environment of the robot is composed of repulsive and attractive objects. Each object has a particular color, denoting its respective class. Obstacles are considered repulsive objects while targets are attractive objects (Antonelo et al., 2006). For the T-maze experiments, the light sign is modeled by an object with a distinct color (see Fig. 2). The robot model is shown in Fig. 1. The robot interacts with the environment by distance, color and contact sensors; and by one actuator that controls the adjustment on the movement direction. Sensor positions are distributed homogenously over the front of the robot (from -90° to +90°). Each position holds three sensors (for distance, color and contact perception) (Antonelo et al., 2006). In this work, the robot model has 17 sensor positions, as in (Antonelo et al., 2007). The velocity of the robot is constant. At each iteration the robot is able to execute a direction adjustment to the left or to the right in the range [0, 15] degrees.

The robot controller (Antonelo et al., 2006) can produce appropriate trajectories for the T-mazes by placing specific attractive objects in the environment. So, the reactive controller does not solve the T-maze task but is only used for generating data (the generation of these trajectories are explained in the following section). The data (from distance and color sensors, and actuator) collected from the robot simulator are saved in files which in turn are used to train and test reservoir networks in a Matlab environment using the

RCT Toolbox[1] (Verstraeten et al., 2007). Note that the RC neural network is not integrated yet with the simulator (the robot controller is used only to generate trajectories for the training of RC networks).

3 Experiments

3.1 Introduction

In this work, we use the Reservoir Computing paradigm to enable a system to learn long-term temporal dependencies in the road sign problem.

In (Rylatt and Czarnecki, 2000), a robot is manually moved (using cursor keys) to generate samples of the robot's sensory inputs and corresponding motor responses. These samples are used to train an Elman recurrent neural network. In the current work, we automate the generation of the training dataset by letting a reactive controller (see previous section) to drive the robot in the environment. The robot knows which side to turn at the T-junction due to the existence of attractive objects (see Fig. 2) which are placed in the environment depending on the current goal. They attract the robot to the correct goal. Naturally the sensory perceptions on attractive objects are not included in the dataset used to train the RC network.

There are two sets of experiments: simple T-maze and complex T-maze experiments. In the first set, usual T-mazes of different sizes are used (see Fig. 2). This is explained in the next section. The following set includes experiments with complex T-mazes which have two signs visible at a time (see Fig. 5), being described in Section 3.3.

The reservoir configuration is as follows for all experiments (except where changes are explicitly stated). The inputs to the network are distance and color sensors suming up 34 inputs which can range from 0 to 1. The sensors' range is limited, so they can saturate (see Fig. 2). The reservoir is composed of 400 nodes, scaled to a spectral radius of $|\lambda_{max}| = 0.9$ (Jaeger, 2001). The readout layer has 1 output unit which corresponds to the robot actuator (current direction adjustment). The input nodes are connected to reservoir nodes by a fraction of 0.3 and are set to -0.2 or 0.2 with equal probabilities.

The original dataset (collected from the simulator) is downsampled by a factor of 100, which is equivalent to slowing down the reservoir timescale (Jaeger et al., 2007; Antonelo et al., 2007) (so, 1 time step in the reservoir corresponds to 100 time steps in the simulator). This is because the robot has a relatively constant low velocity, taking about 1300 time steps to go from the start position until the goal for example in environment E1 (Fig. 2). The same resampling factor is used for every experiment in this paper unless otherwise stated (although a greater resampling can yield better results if the delay period is too large between the light sign and the cue at the T-junction).

The dataset (sensory inputs and motor output) collected from the simulator are divided in three parts, where two parts are used for training and the other one for testing. The current stage of the work does not include the test of the RC network as a controller embedded in the simulator. It is only tested on pre-recorded sensory inputs from the simulator (not real-time). The performance measure is based on the normalized mean square error (NMSE) calculation on a 3-fold cross validation with ridge regression as the learning algorithm. Distance sensors data are 20% noisy while color sensors data are 10% noisy (which are high noise rates when compared to (Rylatt and Czarnecki, 2000) who consider noise-free data). The noise is generated from a Gaussian distribution.

3.2 Simple T-mazes

This section presents three experiments on the most common form of the road sign problem: the T-maze task (see Fig. 2). The T-maze environments were built considering different lengths for the main corridor in order to test the memory capacity of a reservoir neural network. The corridor in environments E2 and E3, shown in Fig. 2, are two and three times bigger than in E1, respectively.

It is important to note that the current approach does not force the robot to go until the T-junction as in (Kim, 2004). Therefore, in our case the robot might hit the wall before reaching the T-junction if driven by the reservoir network. This is a more difficult problem because the reservoir network also has to learn to drive forward until the T-junction and only then turn right or left. Additionaly, this work is not based on event extraction which works on a higher abstraction level as in (Linaker and Jacobsson, 2001) but instead the reservoir receives the continuous raw sensor data.

Results are summarized in Table 1. Each experiment has a dataset with 60 examples (each example is built with sensor and actuator samples from a robot going from the start position to the goal position). Also each experiment is evaluated 30 times (runs) with different stochastically generated reservoirs and the results (error) are averaged over these 30 runs.

Remember that the output (readout) of the reservoir is the actuator of the robot (for direction adjustment) . To evaluate whether the predicted trajectory is *correct* (i.e. whether the reservoir is able to drive the robot through the T-maze

[1]This is an open-source Matlab toolbox for Reservoir Computing which is freely available at http://www.elis.ugent.be/rct

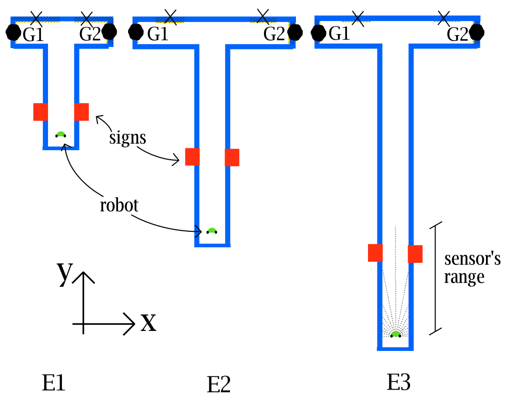

Figure 2: Simple T-mazes of distinct sizes. The robot only *sees* 1 sign at a time. A sign at the left (right) indicates that the goal is at the left arm, in G1 (right arm, in G2). Crosses mark the positions of attractive objects which are visible to the robot controller (which automates the generation of the samples) but not for the RC network.

and solve the task correctly), we plot the real trajectory and the predicted trajectory by moving two points in X and Y coordinates as they were driven by the desired and predicted actuators, respectively. Fig. 3 shows three plots for T-mazes E1, E2 and E3. They show the real trajectory of the robot (each point given by an asterisk) and the trajectory formed by reading the reservoir responses (each point represented by circles). Note that the predicted trajectory was built by stimulating the reservoir with test data not contained in the training data.

It is easily seen that as the size of the corridor increases the difficulty for getting a 100% effective reservoir is also higher. For T-mazes E1 and E2, the test error is very low (Table 1). It is also accomplished the evaluation of the reservoir performance by visual inspection (i.e., cheking the real and predicted trajectories). This is done by training a reservoir network on a dataset with 40 examples, and counting the number of *correct* and *bad* predicted trajectories (which correspond to *Paths A* and *Paths B* in Table 1, respectively) for the training and test (whose size is 20 examples) datasets. The predictions that are not in *Paths A* neither in *Paths B* represent an intermediate class for which we are not sure whether the reservoir would drive a simulated robot in real time to the goal.

The trajectories in Fig. 3 were randomly chosen between the set of *correct* trajectories (Paths A). The set of *bad* predictions (Paths B) have the worst trajectories such that the reservoir would likely not drive the robot to the goal position if the reservoir network was integrated in the simulator in real time. For instance, in Fig. 6 the trajectory built from the reservoir output for T-maze E2 deviates significantly from the desired

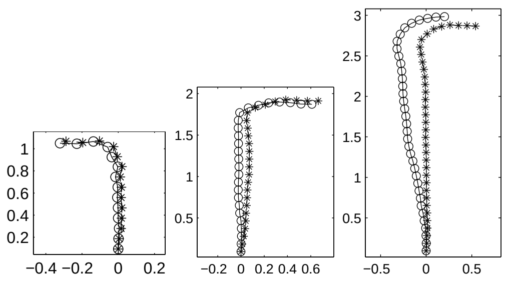

Figure 3: Plots showing reasonable robot path predictions for the simple T-maze experiments. Each point of the desired trajectory is marked by an asterisk while circles represent points of the predicted robot path. From left to right are the plots for T-mazes E1, E2 and E3 respectively.

robot trajectory. Actually, we do not know how this bad prediction would work with data in real time from the simulator. This is because when the predicted robot trajectory first deviates after the first 4 time steps, the reservoir still *thinks* that the robot is going straight ahead once the data that feed the network does not reflects the motor outputs from the reservoir. This will be solved by integrating a RC network into the robot simulator in order to know the real outcome with real-time sensory input (which is left as future work). Nonetheless, the current results clearly show that RC networks can solve the T-maze task.

We calculated the time gap in Fig. 3 between the first time step in which the robot looses the sight of the sign and the first time step for the decision at the T-junction. These values are approximately 6, 10 and 19 time steps for T-mazes E1, E2 and E3, respectively. In (Rylatt and Czarnecki, 2000), the road sign problem was solved by employing Elman networks with BPTT with reliable results up to around 6 time steps for the time gap between stimulus and decision (considering noise-free data). In our approach, we get very effective RC networks even considering time gaps of circa 10 time steps and very noisy data.

In Fig. 4, it is possible to analyse the effect of the reservoir size on the performance on the T-maze tasks. For T-maze E1, a reservoir with 40 nodes is already enough to produce good results (without any bad trajectories). It is also possible to note that a reservoir with 400 nodes offers more memory and increases the performance for T-mazes E2 and E3 (as these environments require the system to hold a history of the received stimuli for longer delays). As a general rule, it is known that as the size of the reservoir increases the memory capacity is also greater (Jaeger, 2001; Verstraeten et al., 2007). On the other hand, changing the downsampling factor to 200 (slowing down the reservoir timescale even more) for T-maze E3, we improve greatly the performance to a error rate (NMSE) of 0.0172 and 0.0419 for training and test

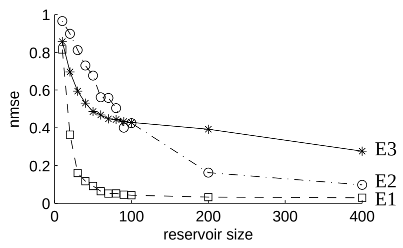

Figure 4: Normalized mean square error on test data x Reservoir size, for T-mazes E1, E2 and E3. A 3-fold cross-validation is executed 10 times and results are the average.

errors, respectively, without any bad trajectories.

3.3 Complex T-mazes

Here we consider T-maze tasks with two visible signs at a time. In Fig. 5, it is possible to see the enviroments used in the experiments. Environment E4 is similar to E2 except that 4 different sign combinations (of 2) exist. The goal is G2 when either both signs at the right are visible or the lower right and upper left signs are both visible. Accordingly, the goal is G1 when either both signs at the left are visible or the lower left and upper right signs are both visible. Environment E5 is a multiple T-maze which has 4 branches. Each single sign combination (of 2) corresponds to a goal: Lower Left and Upper Left (G1), Lower Left and Upper Right (G3), Lower Right and Upper Right (G2), Lower Right and Upper Left (G4). The length of the main corridors of E4 and E5 are equal to the length of the corridor of E2.

Both training and test errors are considerably higher in E4 than in E2, despite the same size of the main corridor. It turns out that for the same

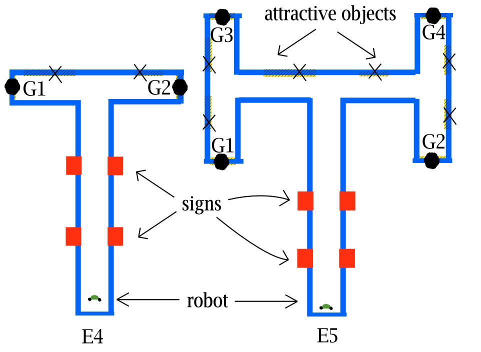

Figure 5: Complex T-mazes. Note that all 4 signs are shown in the figure for visualization purposes (the robot only *sees* two signs). Depending on the current goal, attractive objects (marked by crosses) are placed in the environment for automated generation of the training dataset (naturally these objects are not *seen* by the RC network).

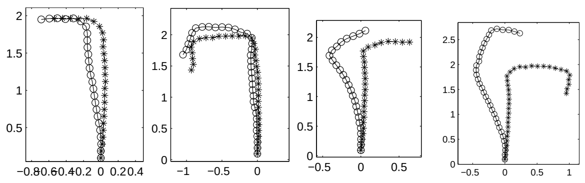

Figure 6: Correct and bad predicted trajectories. Each point of the desired trajectory is marked by an asterisk while circles represent points of the predicted robot path. The first two plots are good predictions whereas the last two are bad ones. From left to right, the environments shown are E4, E5, E2, E5.

Table 1: Summarized results. Results in parenthesis are related to performance on test data. Paths A are the number of correct predicted trajectories (40 and 20 trajectories are the maximum values for the training and test sets respectively). Paths B are the number of bad predictions.

Env.	Error(NMSE)	Paths A	Paths B
E1	0.0191(0.0341)	40(20)	0(0)
E2	0.0559(0.1080)	40(19)	0(0)
E3	0.2122(0.3346)	20(8)	3(4)
E4	0.1952(0.3185)	27(12)	0(2)
E5	0.2072(0.2863)	19(10)	6(5)

reservoir configuration the difficulty for establishing a correct path until the goal is increased when more than one sign are associated with the same goal. Furthermore, in E4 the reservoir has to discriminate between different combinations of signs that are temporally structured. For instance, the Upper Left sign can be associated with either G1 or G2, depending on the side of the lower sign which first appears. This characteristic may be the main reason for the increased error compared to the experiments in E2. Nevertheless, we still get around 27 (12) correct trajectories for the training (test) data in T-maze E4 out of 40 (20) trajectories. The left plot in Fig. 6 shows a correct predicted trajectory for T-maze E4. The performance of RC networks for T-maze E4 and E5 are comparable in terms of the NMSE, but the number of *bad* predicted trajectories are higher for T-maze E5, as expected. In E5, the RC network must discriminate distinct sign combinations as well as to associate each of them to a goal (out of 4). The second plot in Fig. 6 shows a correct prediction for T-maze E5 whereas the last plot in the same figure represents a predicted trajectory with deviates considerably from the desired one for the same T-maze.

4 Conclusion

This work tackles the road sign problem in the form of a T-maze navigation task using generic recurrent neural networks which are processed by a single linear readout layer. We show that RC networks are well suited to deal with long-term temporal dependencies between relevant events. The network holds the information gathered in the past (e.g., the light sign) because of its recurrent connections, being able to produce adequate outputs (e.g., turn right or left) after some delay period. Only a readout layer in the RC network is trained by a fast linear regression algorithm, avoiding the convergence problems of BPTT and the need for unfolding the network in time as in (Rylatt and Czarnecki, 2000). Experiments consider simple T-maze tasks and even more complex T-mazes with different sign combinations.

The next stage of this work is to embed the RC network in the simulator. In this way, we can evaluate the network driving the robot in real-time. Another interesting point is to test the generalization capability of RC networks for arbitrary corridor lengths.

As a general and efficient learning system for temporal problems, RC networks can be combined with reinforcement learning, giving rise to several potential applications (mainly on control). Therefore, an agent could learn by reinforcement to associate a sign perceived in the past with actions that lead to a future reward. This is done in (Bakker, 2002) using LSTM with reinforcement learning (although their agent's world is discrete).

From the RC point of view, it is possible to study further the impact of the reservoir timescale on the T-maze task. By slowing down the reservoir timescale, the RC network can cope with longer delay periods between the relevant events. Additionaly, by creating an unsupervised method which adapts the reservoir timescale for different sizes of the T-maze, we can obtain a system that works for arbitrary delay periods between the sign and the turning decision at the T-junction.

Acknowledgments

This work was partially funded by FWO Flanders project G.0317.05. The authors wish to thank David Verstraeten for his important contributions to this work.

References

Antonelo, E. A., Baerlvedt, A.-J., Rognvaldsson, T. and Figueiredo, M. (2006). Modular neural network and classical reinforcement learning for autonomous robot navigation: Inhibiting undesirable behaviors, *Proc. of IJCNN 2006*, Vancouver, pp. 498– 505.

Antonelo, E. A., Schrauwen, B., Dutoit, X., Stroobandt, D. and Nuttin, M. (2007). Event detection and localization in mobile robot navigation using reservoir computing, *in* J. M. de S´a et al. (ed.), *ICANN, Part II*, Springer-Verlag, pp. 660–669.

Bakker, B. (2002). Reinforcement learning with long short-term memory, *Advances in Neural Information Processing Systems*, Vol. 2, MIT, pp. 1475–1482.

Jaeger, H. (2001). Short term memory in echo state networks, *Technical Report GMD Report 152*, German National Research Center for Information Technology.

Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication, *Science* **308**: 78–80.

Jaeger, H., Lukosevicius, M. and Popovici, D. (2007). Optimization and applications of echo state networks with leaky integrator neurons, *Neural Networks* **20**: 335–352.

Kim, D. (2004). Evolving internal memory for t-maze tasks in noisy environments, *Connection Science* **16**(3): 183–210.

Linaker, F. and Jacobsson, H. (2001). Mobile robot learning of delayed response tasks through event extraction: A solution to the road sign problem and beyond, *Proc. IJCAI'2001*, pp. 777–782.

Maass, W., Natschläger, T. and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations, *Neural Computation* **14**(11): 2531–2560.

Rylatt, R. M. and Czarnecki, C. A. (2000). Embedding connectionist autonomous agents in time: The 'road sign problem', *Neural Processing Letters* **12**: 145–158.

Steil, J. J. (2004). Backpropagation-Decorrelation: Online recurrent learning with O(N) complexity, *Proceedings of IJCNN '04*, Vol. 1, pp. 843–848.

Ulbricht, C. (1996). Handling time-warped sequences with neural networks, *in* e. a. Maes P. (ed.), *From Animals to Animats 4: Proc. Fourth Int. Conf. on Simulation of Adaptive Behaviour*, MIT Press, pp. 180–192.

Verstraeten, D., Schrauwen, B., D'Haene, M. and Stroobandt, D. (2007). A unifying comparison of reservoir computing methods, *Neural Networks* **20**: 391–403.

Ziemke, T. and Thieme, M. (2002). Neuromodulation of reactive sensorimotor mappings as a short-term memory mechanism in delayed response tasks, *Adaptive Behavior* **10(3/4)**: 185–199.