Mobile Robot Control in the Road Sign Problem using Reservoir Computing Networks

Eric Antonelo, Benjamin Schrauwen and Dirk Stroobandt

***Abstract*— In this work we tackle the road sign problem with Reservoir Computing (RC) networks. The T-maze task (a particular form of the road sign problem) consists of a robot in a T-shaped environment that must reach the correct goal (left or right arm of the T-maze) depending on a previously received input sign. It is a control task in which the delay period between the sign received and the required response (e.g., turn right or left) is a crucial factor. Delayed response tasks like this one form a temporal problem that can be handled very well by RC networks. Reservoir Computing is a biologically plausible technique which overcomes the problems of previous algorithms such as Backpropagation Through Time - which exhibits slow (or non-) convergence on training. RC is a new concept that includes a fast and efficient training algorithm. We show that this simple approach can solve the T-maze task efficiently.**

I. INTRODUCTION

An increasing number of research groups have raised their attention to the fields of intelligent autonomous systems and learning robots. These systems are usually designed by computational intelligence techniques which provide a rich ground for achieving learning capabilities as well as robustness to noise and environment changes. The biological foundation of these techniques comes from several areas and includes: computational models of the brain [8], evolutionary-based systems [12], swarm intelligence techniques [5] and reinforcement learning systems [4].

The road sign problem, which is tackled in this work, constitutes a particular temporal task which is defined in [24]. In this problem, an artificial agent (robot) which is driving along a corridor receives a temporal sign that must be remembered for future correct decision making. The T-maze task is the most common form of such problem: the robot drives along an environment whose shape resembles the letter T (see Fig. 3). The robot's task is to drive from the initial position located at the bottom of the longest corridor, reach the T-junction and then turn to the correct goal (left or right). The correct turning decision at the T-junction depends on the previous input sign received while driving along the corridor (usually a sign at the left/right side of the corridor indicates that the goal is at the left/right arm of the T-maze).

Several systems designed to solve such task represent the robot's environment as a discrete world [4] in order to make the task easier to be solved. Sometimes the world's representation is not discrete, but instead the models are designed with event extraction mechanisms which work on the robot sensory cues in order to provide abstract signals to the control module [14]. Recent work has tackled the road sign problem with a continuous world representation [18], [26], [12]. Most approaches to the road sign problem are based on recurrent neural networks [24], [18], [26]. The work in [26] 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. 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 contrast to the current work which does not yield wall following behaviors). In [12], evolutionary multi-objective optimization is used to evolve finite state controllers for the T-maze task, whose control task is simplified by forcing the robot to first reach the T-junction. In that work, a detailed analysis of the required internal memory for the T-maze task is accomplished. Reinforcement learning with Long Short-Term Memory (LSTM) is the approach used in [4] 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.

New computational models of the brain have been proposed in the literature [16], [8] under the name of Liquid State Machines (LSMs). This technique uses a reservoir of spiking neurons that has fixed random weights and is sparsely connected. The reservoir is a RNN which produces rich dynamics of temporal nature (because of the recurrent connections in the reservoir). The states of the reservoir are mapped onto a readout output layer (see Fig. 1). The training is only accomplished for the connection weights in the readout output layer with standard linear regression techniques. The engineering counterpart of the LSM is the Echo State Network (ESN) , which was created independently by [11]. In [25], both LSM and ESN (as well as BackPropagation DeCorrelation [22]) are compared and termed jointly as Reservoir Computing (RC) because of their great similarities.

Reservoir Computing has been applied successfully in several applications, such as: mobile robot control [19], [6], robot localization and event detection [2], time series

This document is a draft version of the paper to appear in the Proceedings of IEEE ICRA 2008.

This work is partially supported by FWO Flanders project G.0317.05. Eric Antonelo is sponsored by a BOF grant.

E. Antonelo, B. Schrauwen and D. Stroobandt are with the Electronics and Information Systems Department, Ghent University, 9000 Ghent, Belgium. eric.antonelo@elis.ugent.be

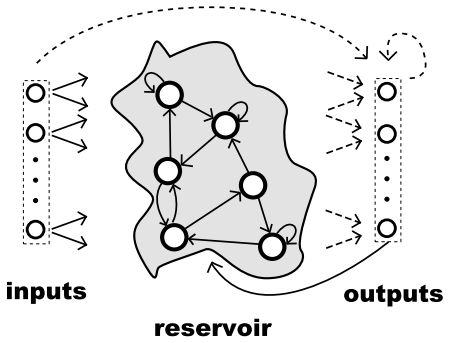

Fig. 1. Reservoir Computing network. The reservoir is a dynamical system of recurrent nodes. Solid lines represent connections which are fixed. Dashed lines are the connections which can be trained.

prediction [13], generation of music sequences [7] and learning grammatical structures [23]. There are also promising applications of ESNs in the domains of system identification or as low-level plant controllers [10].

This work uses Reservoir Computing as an efficient tool for training robot controllers to solve the T-maze task. The main advantages of our approach are threefold: simplicity of the approach; efficient and fast training of controllers; and integration of reactive and sequential behavior in a single control module. The first point is related to the easing of the design task by just employing the RC networks to perform the T-maze task by imitation learning (i.e., it is a black-box approach with high performance which facilitates a complex design task). The next point corresponds to the efficient and fast training of ESNs by standard linear regression methods [25] which is superior to previous methods and avoids convergence related problems like Backpropagation Through Time. Finally, there is no separation between reactive and sequential (deliberative) behavior (i.e., there is only one control module), which forces the RC network to learn the whole task by imitation learning. In this way, the robot has to learn to move in its environment (reactive part) but it also should learn the sequential task of reaching the T-junction and turn to the correct goal depending on the previous cue received (deliberative part).

The next section presents the Reservoir Computing approach used for this work as well as the autonomous robot simulator and robot model. The accomplished experiments and associated results are presented in Section III. Finally, the last section summarizes and concludes this work.

II. METHODS

A. 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:

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

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

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

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

The readout matrix $\mathbf{W}_{\text{out}}$ is created by solving (in the mean square sense) the following equation:

$$\mathbf{M}\mathbf{W}_{\text{out}} = \hat{\mathbf{Y}}, \tag{3}$$

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

We use the Reservoir Computing Toolbox (RCT Toolbox [1] [25]) for training robot controllers in a Matlab environment.

B. Autonomous Robot Simulator

The simulation of the road sign problem in the form of a T-maze task is accomplished using a sophisticated autonomous robot simulator developed in C++ [1]. The environment of the robot is composed of several objects, each one of a particular color. Obstacles are represented by blue objects whereas the light sign in the T-maze is simulated by a red object. The robot model is shown in Fig. 2. The robot interacts with the environment by distance and color sensors; and by two actuators which control the movement direction (turning) and speed. Sensor positions are distributed uniformly over the front of the robot (from -90° to +90°). Each position holds two virtual sensors (for distance and color perception) [1]. The following experiments consider that the robot model has either 3 or 7 sensor positions (see Fig. 2). The distance sensors are limited in range (i.e., they saturate for distances greater than 300 distance units (d.u.)) and are noisy (they exhibit Gaussian noise on their readings). A value of 0 means near some object and a value of 1 means far or nothing detected. The color sensor is calculated as the normalization of the component Hue of the Hue-Saturation-Value (HSV) color system. 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 and the speed is limited to [0, 17] distance units (d.u.).

III. EXPERIMENTS

A. Introduction

In this work, we use the Reservoir Computing (RC) paradigm to enable a simulated mobile robot to solve the T-maze task. The environments used for the experiments are shown in Fig. 3. The task of the robot is to drive from the initial position until the T-junction and then turn left/right if the sign previously appeared at the left/right side of the longest corridor. Environment B has a longer corridor (2x)

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

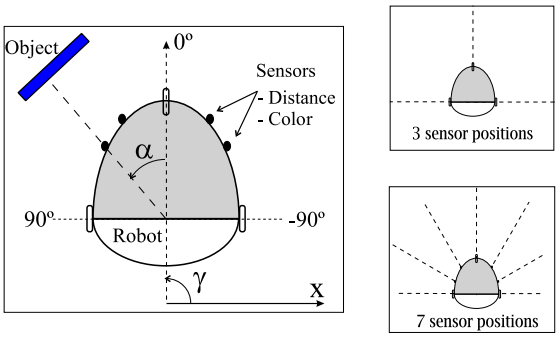

Fig. 2. Robot model.

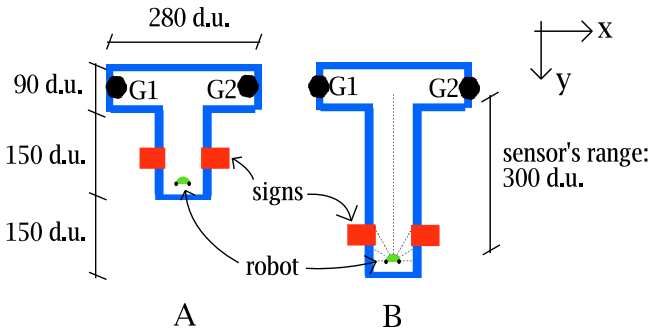

Fig. 3. Environments used for the experiments. The robot only *sees* 1 sign at a time. A sign at the left (right) indicates that the goal is at the left arm, in G1 (right arm, in G2).

than in environment A. This will enable us to see how the task is solved when a longer delay between the cue (light sign) and the subsequent response (turning) is required.

The experiments are divided in three stages: acquisition of the training dataset; training the RC-based robot controller; and testing of the resulting robot controllers. These three steps are executed for each environment (i.e., a controller trained with data from environment A is only tested in the same environment). The first stage consists of using the robot simulator to generate a dataset with samples of the robot's sensory inputs and actuators by driving the robot through the T-maze with a simple set of rules (e.g., go from the initial position until the T-junction, then turn left if the sign was at the left side). The dashed line in Fig. 4 represents an example of a correct trajectory generated by such algorithm. We collect around 50 examples for the training dataset which considers random robot starting positions (in the range [-10,10] d.u. for X and Y coordinates) and robot heading (in the range [-15,15] degrees). After data acquisition, the second stage consists of performing imitation learning with the RC network by using the previous collected examples to train the controller, characterizing a generalization process. The RC network is trained to output the desired turning and speed values for solving the robot task (in the Matlab environment). The last stage is the performance testing of the RC-based controller in the T-maze task, which is based on the real-time communication between the Matlab process and the robot simulator (implemented by TCP/IP sockets). The previous work [3] only considered off-line testing on pre-recorded sensory inputs from the simulator (not real-time).

The average number of timesteps for the realization of the T-maze task by the algorithm which generates the training dataset is 26.3 timesteps for environment A (standard deviation of 1.6) and 34.9 timesteps for environment B (standard deviation of 1.5). In the testing stage, the T-maze task has to be accomplished in 38 and 46 timesteps for environments A and B, respectively (this was arbitrarily set). These values do not include the first 20 timesteps in which the robot stays still in the data acquisition stage as well as in the testing phase. During these first timesteps, the reservoir starts at the initial state $\mathbf{x}(t) = \mathbf{0}$ and follows an undesired transient response to the input. The reservoir reaches a steady state after some transient interval. So the training of the readout output layer discards the first 20 timesteps of the data which are used only for initializing the state of the reservoir (this is called warm-up drop in the literature). At each timestep, Gaussian noise is added to the robot's actuators from the distributions N(0,2) for the robot turning (in degrees) and from N(0,0.5) for the robot speed (in d.u.). This noise on actuators is considered in the data acquisition stage as well as in the testing stage.

The reservoir configuration is as follows for all experiments in this work. The inputs to the network are distance and color sensors totalling either 6 inputs (if the robot model has 3 distance sensors and 3 color sensors) or 14 inputs (if the robot model has 7 distance sensors and 7 color sensors) which can range from 0 to 1. The reservoir is composed of 500 sigmoidal nodes, scaled to a spectral radius[2] of $|\lambda_{max}| = 0.9$ [9], which approximately sets the reservoir at the edge of stability (sometimes referred to as the edge of chaos). The readout layer has 2 output units which correspond to the robot actuators (i.e., robot turning and speed). The connection matrix from input to the reservoir is initialized at the values -0.1, 0.1 and 0 with probabilities 0.1, 0.1 and 0.8, respectively. This parameter setting for weight matrices are not critical for the tasks in this work.

The robot sensors on the training datasets are 5% noisy (Gaussian noise from $N(0, 0.05)$ which means effectively $N(0, 15)$ in distance units) and on color sensors (Gaussian noise from $N(0, 0.05)$) ([18] only considers noise-free data for a particular version of the road sign problem solved with Elman networks). The performance tests consider either 1% or 5% Gaussian noise on the robot sensors (this will be stated accordingly in the text).

B. Results

In this section, we will investigate how the number of sensors in the robot model and the noise level on the sensor readings affect the performance of the RC-based controller on the T-maze task. This analysis is made for both environments A and B.

An example of the robot's trajectory in the T-maze of environment A is shown on Fig. 4(a). The solid line which

[2]The spectral radious λ_{max} is the eigenvalue of the matrix $\mathbf{W}$ with the largest absolute value.

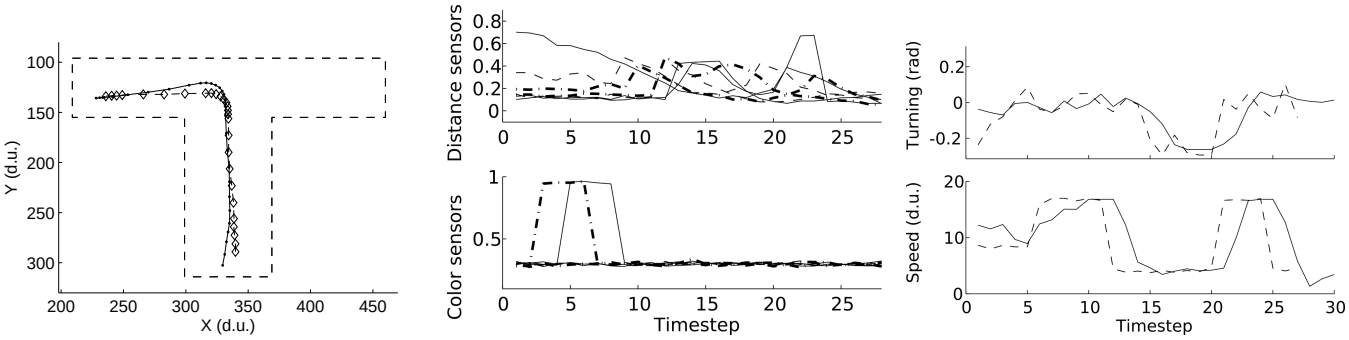

Fig. 4. Plots for the robot trajectory (a), robot sensors (b) and actuators (c) when the sign appears at the left side of the corridor in the T-maze of environment A. The desired path takes 26 timesteps whereas the controlled robot path takes 38 timesteps. The robot model has 7 distance sensors and 7 color sensors which are 1% noisy. (a) The solid line is the real robot trajectory driven by the RC network. The dashed line is an example of desired trajectory. (b) The 14 inputs to the network (i.e., 7 distance sensors and 7 color sensors readings) in the testing stage for 30 timesteps. (c) The robot's actuators given by the RC network (solid line; for 30 timesteps) and given by the desired path (dashed line; for 26 timesteps).

connects the robot positions at each timestep represents the real trajectory of the robot (driven by the output of the RC network) whereas the dashed line which connects small boxes represents an example of a desired trajectory. The corresponding sensory inputs and robot actuators are given in Fig. 4(b) and Fig. 4(c), respectively.

Fig. 4 shows that the control task is smoothly performed by a single control module (i.e., the RC network). In that example, the trajectory shows that both reactive and sequential behaviors are achieved with our simple approach. After training, the RC network can drive the robot exclusively based on sensor data and can hold the past information for posterior decision making. The recurrent pathways in the reservoir yield a fading memory which is crucial for solving the T-maze task. Traditional feedforward neural networks are not capable of this [18]. Furthermore, we can see that it takes at least 5 timesteps between the last perception of the sign in the corridor (timestep 9) and the start of the turning movement (timestep 14) for this short environment.

The robot trajectory given by the trained RC network in the T-maze of environment B is shown in Fig. 5. We can observe that the time gap between the cue received in the corridor and the decision making at the T-junction can be even greater (18 timesteps) while the task is still solved correctly.

In order to know the repeatability of the performance of the experiments, we generate statistics for environments A and B, for different noise levels on sensor readings (1% or 5%), and for distinct robot models (with 3 or 7 sensors). The summarized results for each combination are presented on Tables I and II. The results are calculated after executing the experiment for 10 different reservoirs, each one being evaluated 30 times. So, each combination results from 300 runs on the T-maze (150 runs for each goal). The numbers in the tables show the percentage of examples (trajectories) that are classified as successful for solving the T-maze task. We consider that the run was successful if the robot reached the inner part of the correct arm at the final timestep. For instance, the run on Fig. 5 was successful because the last point of the trajectory has an abscissa which is lower than the abscissa (300) of the left side of the main corridor (so the left and right sides of the corridor are delimiters).

The tables show that a robot model with 7 sensors provides important additional information for solving the T-maze task appropriately when compared to a robot model of 3 sensors. The model with 7 sensors increases performance over the model with 3 sensors by 37% for environment A and by 45% for environment B (figures for the left goal - see Table I). It is also possible to observe that the effect of increasing the noise level on the sensor readings mainly affects the experiments on environment B, specially the ones considering the robot model with 7 sensors. In this case, the degradation in performance is up to 13%. We conclude that

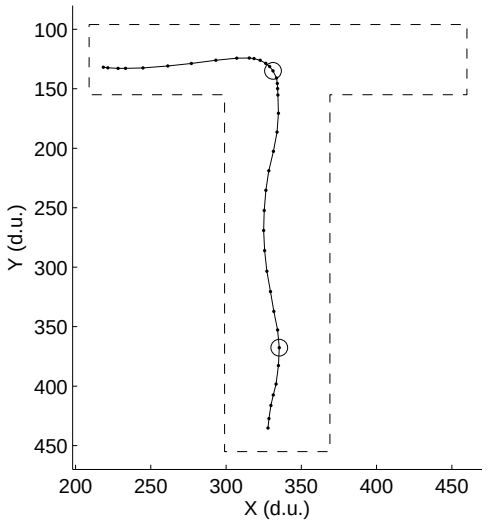

Fig. 5. Plot for robot trajectory driven by RC network in environment B. Circles represent the moment in which the robot loses the sight of the sign and the starting time of the turning at the T-junction. This time gap corresponds to 18 timesteps. The robot model has 7 distance sensors and 7 color sensors which are 1% noisy.

TABLE I

PERCENTAGE OF CORRECT TRAJECTORIES FOR 1% NOISE ON SENSORS

	3 sensors		7 sensors	
	Left	Right	Left	Right
Environment A	**58%**	**62%**	**95%**	**93%**
Environment B	**37%**	**33%**	**82%**	**81%**

TABLE II

PERCENTAGE OF CORRECT TRAJECTORIES FOR 5% NOISE ON SENSORS

	3 sensors		7 sensors	
	Left	Right	Left	Right
Environment A	**67%**	**70%**	**93%**	**87%**
Environment B	**28%**	**26%**	**69%**	**69%**

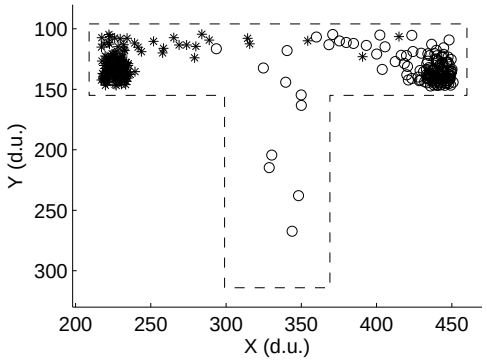

Fig. 6. Distribution of ending points in environment A. Each point represents the final robot position after 38 timesteps in which the RC network drives the robot. There are 300 points (generated by 10 different reservoirs which run 30 examples each). Asterisks and circles represent left and right goals for the current task, respectively.

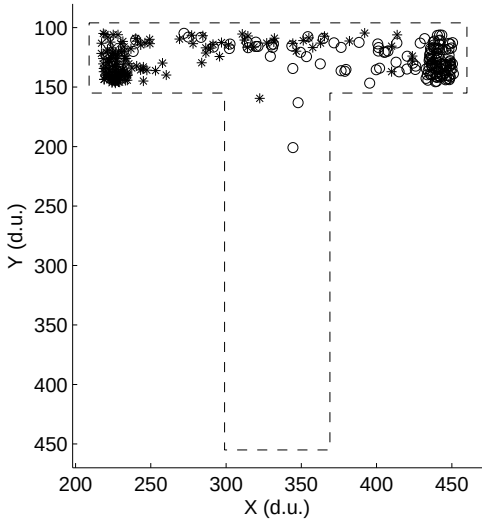

Fig. 7. Distribution of ending points in environment B. Each point represents the final robot position after 46 timesteps in which the RC network drives the robot. There are 300 points (generated by 10 different reservoirs which run 30 examples each). Asterisks and circles represent left and right goals for the current task, respectively.

the experiments in a longer T-maze (environment B) take more advantage of the addition of extra information in the form of more sensors in the robot model. Furthermore, higher noise levels negatively influence difficult T-maze tasks (the difficulty is directly related to the size of the main corridor, i.e., the time gap between cue and required response).

Part of the statistics summarized in the aforementioned tables is available visually in Fig. 6 and Fig. 7. These figures show the coordinates of the robot in its environment at the final timestep of each run. The robot model with 7 sensors and a noise level of 1% are considered here. Observe that we only look at the final position of the robot, while it can sometimes still collide against a wall during the intermediate steps (a collision will cause a step back and a small change on the direction of movement). There are 300 points for each figure which represent distinct runs through the respective T-maze. A circle/asterisk means that the sign appeared in the right/left side of the corridor for the corresponding run. We can note that circles are concentrated on the right arm whereas asterisks are located mainly on the left arm, as expected.

IV. CONCLUSION AND FUTURE WORK

In this work we presented a simple approach to the road sign problem by employing Reservoir Computing (RC) networks in the modeling of the robot controller. RC is a simple technique which provides easy and fast training of recurrent neural networks. The reservoir is a dynamical system whose states are mapped in the readout output layer. This mapping is learned through standard linear regression techniques, which makes this technique very powerful, avoiding the convergence problems of previous algorithms like Backpropagation Through Time as well as the need to unfold the network in time as in [18]. The technique is biologically plausible [8], what is in line with the most recent advances in intelligent autonomous systems which seek models with more and more biological foundations.

The road sign problem is a type of delayed response task, where relevant input information gathered in the past (e.g., the light sign) determines adequate output decisions (e.g., turn right or left) after some delay period. The T-maze task is the most common form of the road sign problem and is tackled in this work. The difficulty of this task relies on the time gap which exists between the cue/sign received and the appropriate required response. Only a single control module is used for solving this task so that reactive and sequential behaviors are integrated and learned simultaneously. Furthermore, we show that the control task is accomplished successfully with time gaps of up to 18 timesteps between the cue and the response.

This work is a significant extension of previous experiments developed in [3]. In the current paper, the testing of robot controllers is done in real-time in contrast to the offline testing on pre-recorded sensory inputs in [3]. Other

differences for the current work include: the speed is not constant and is controlled by the RC network, the sensor data are not resampled and the robot model has a limited number of sensors. We generate statistics for the experiments in this work, making it possible to draw reliable conclusions.

Further study on longer T-maze environments which require longer delay periods for the postponed response is left as future work. In this case, the design of an unsupervised method for adapting the timescale of the reservoir to the input flow may be an interesting approach that would work for arbitrary delay periods. For instance, by lowering the timescale of the reservoir when the input is slowly varying (when the robot drives in a straight line along the main corridor) and increasing this timescale back otherwise, the performance can greatly be enhanced for long delay periods because the memory of the reservoir is increased with this new scheme[15]. These ideas of working with the timescale of reservoirs can find applications in other areas such as speech recognition [20], [21]. We also plan to validate the current work on a real robotic setup using the mobile robot e-puck [17].

REFERENCES

[1] E. A. Antonelo, A.-J. Baerlvedt, T. Rognvaldsson, and M. Figueiredo. Modular neural network and classical reinforcement learning for autonomous robot navigation: Inhibiting undesirable behaviors. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN)*, pages 498– 505, Vancouver, 2006.

[2] E. A. Antonelo, B. Schrauwen, X. Dutoit, D. Stroobandt, and M. Nuttin. Event detection and localization in mobile robot navigation using reservoir computing. In J. M. de Sa et al., editor, *ICANN, Part II*, pages 660–669. Springer-Verlag, 2007.

[3] E. A. Antonelo, B. Schrauwen, and D. Stroobandt. Experiments with reservoir computing on the road sign problem. In *VIII Brazilian Congress on Neural Networks (CBRN)*, 2007. (in press).

[4] B. Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems*, volume 2, pages 1475–1482. MIT, 2002.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[6] H. Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In *Proceedings of the 9th International Conference EANN*, 2005.

[7] D. Eck. Generating music sequences with an echo state network. In *NIPS 2006 workshop on Echo State Networks and liquid state machines*, 2006.

[8] S. Haeusler and W. Maass. A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral Cortex*, 17(1):149–162, 2007.

[9] H. Jaeger. Short term memory in echo state networks. Technical Report GMD Report 152, German National Research Center for Information Technology, 2001.

[10] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *Advances in Neural Information Processing Systems*, pages 593–600, 2003.

[11] H. Jaeger and H. Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. *Science*, 308:78–80, April 2 2004.

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

[13] A. Lazar, G. Pipa, and J. Triesch. Fading memory and times series prediction in recurrent networks with different forms of plasticity. *Neural Networks*, 20(3):312–322, 2007.

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

[15] M. Lukosevicius, D. Popovici, H. Jäger, and U. Siewert. Timewarping invariant echo state networks. *IUB technical report*, 2:1–15, 2006.

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

[17] F. Mondada. E-puck education robot, September 2007. http://www.e-puck.org/.

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

[19] M. Salmen and P. G. Plger. Echo state networks used for motor control. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1953–1958, 2005.

[20] B. Schrauwen, J. Defour, D. Verstraeten, and J. Van Campenhout. The introduction of time-scales in reservoir computing, applied to isolated digits recognition. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN)*, 2007.

[21] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. Van Campenhout. Compact hardware for real-time speech recognition using a liquid state machine. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2007.

[22] J. J. Steil. Backpropagation-Decorrelation: Online recurrent learning with O(N) complexity. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN)*, volume 1, pages 843–848, 2004.

[23] M. H. Tong, A. D. Bickett, E. M. Christiansen, and G. W. Cottrell. 2007 special issue: Learning grammatical structure with echo state networks. *Neural Networks*, 20(3):424–432, 2007.

[24] C. Ulbricht. Handling time-warped sequences with neural networks. In e. a. Maes P., editor, *From Animals to Animats 4: Proc. Fourth Int. Conf. on Simulation of Adaptive Behaviour*, pages 180–192. MIT Press, 1996.

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

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