Using Echo State Networks for Robot Navigation Behavior Acquisition

Cedric Hartland and Nicolas Bredeche
TAO/LRI – Univ. Paris-Sud ; CNRS ; INRIA futurs
Orsay, France
{*cedric.hartland, nicolas.bredeche*}*@lri.fr*

Abstract— **Robot Behavior Learning by Demonstration deals with the ability for a robot to learn a behavior from one or several demonstrations provided by a human teacher, possibly through tele-operation or imitation. This implies controllers that can address both (1) the feature selection problem related to a great amount of mostly irrelevant sensory data and (2) dealing with temporal sequences of demonstrations. Echo State Networks[10] have been proposed recently for time series prediction and have been shown to perform remarkably well on this kind of data. In this paper, we introduce ESN to robot behavior acquisition in the scope of a mobile robot performing navigation tasks. ESN actually show comparable and even better performance with that of other algorithms from the literature in similar experimental conditions. Moreover, some properties regarding dynamics of ESN in the context of learning by demonstration are investigated.**

Index Terms— **Robot Programming by Demonstration, Behavior Acquisition, Mobile Robotics, Echo State Networks.**

I. Introduction

Robot programming by demonstration, or robot behavior acquisition through demonstration, provides a trade-off between fully autonomous learning robot and ad hoc controllers. In this setup, it is possible to both demonstrate a task which would be difficult to formalize and to incrementaly shape behaviors in a trial and error fashion. This field has long been studied in the scopes of mobile robot navigation task and robotic arm manipulation or coordinated mouvement (mostly through imitation).

In both setup, the main issue to be adressed is that of dealing with many, possibly noisy and vastly irrelevant data and with temporal sequence. To address these problems, previous works showed that probabilistic models, coupled with efficient dimension reduction algorithms, or either continuous or discrete recurrent neural networks, achieved great results.

Recently, Jaeger proposed a new approach to recurrent neural network : the Echo State Network (ESN [10]). This approach has been shown to be very efficient considering time series prediction by reformulating the task of learning temporal sequence into selection of combined available internal dynamics. As a result, ESN provides rich dynamics along with a simple learning algorithm.

In the scope of this paper, we propose to address the problem of robot learning by demonstration using an ESN as controller. We consider the task where a real world mobile robot is teached by a human supervisor to perform simple navigation tasks in an environment and compare the efficiency of our ESN controller to a learning by demonstration algorithm from the literature.

While learning by demonstration is not limited to time series prediction, it turns out that ESN properties makes it a relevant choice in this setup. Indeed, further experiments show that ESN are able to both perform implicit feature selection over the sensory inputs and to be robust towards disruptions in the environment.

The paper is organized as follow : section II gives an overview of robot learning by demonstration and section III presents the general setup of ESN as well as a preliminary experiment. Then, the ESN architecture is described and evaluated in sections IV and V. Section VI provides further insights as to why and how the ESN approach actually works in the context of robot programming by demonstration. Finally, conclusions and perspectives are described in the last section.

II. Robot programming by demonstration

Robot programming by demonstration is a key research interest in robotics. In robot programming by demonstration, a human teacher helps a robot learning new skills [5] and eventually refining by correcting the acquired behavior thanks to the teacher feedback[7]. Works in this field tackle the development of algorithms for motor control and learning, gesture recognition and visuo motor integration. Two main areas of research can be distinguished :

1) ”programming by demonstration” : a human supervisor takes control over the robot (e.g. with a joystick), which is usually considered in the context of mobile robot [13], [9], [12];
2) ”programming by imitation” : a human teacher shows the correct behavior (e.g. using a camera video, laser sensors, etc.), which concerns mainly object manipulation or sequences of mouvement using a robotic arm [2], [6], [17].

In the following we are interested in navigation behaviour acquisition for mobile robot. On the one hand, autonomous learning robot (e.g. evolutionary robotics[14], reinforcement learning[19], etc.) often ends up with very simple behavior when it comes to real robots because controllers are too difficult to optimize with regards to the size of the search space. On the other hand, handwritten controller might capture domain knowledge but often fail to provide efficient behaviours in environment difficult to model with accuracy (i.e. where no optimal analytical solution can be drawn). Robot programming by demonstration offer a compromise between these two approaches, where the sensori-motor space in which the controller is to be optimized is constrained by demonstration from the (human) supervisor.

There are several main setups (amount of data, number of demonstrations, ...) and issues (dimension reduction, dealing with time series, learning from few examples) regarding robot programming by demonstration. One of the main issue is the design of controller that is able to deal with time series prediction. Indeed, robot navigation can be considered as a highly sequential task where the markov property may not hold (i.e. $f(s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) \rightarrow s_{t+1}$ rather than $f(s_t, a_t) \rightarrow s_{t+1}$). Current and past approaches include recurrent neural networks [3], various flavours of markov models [4] and other approaches based on standard regression setup [9].

In this scope, Echo State Network provides a good alternative but, so far, has never been used in the context of Robot Programming by Demonstration.

III. Echo State Network

Echo state networks (ESN) have been proposed by Jaeger in 2001[10] with the objective to endow a neural network with rich dynamics behavioral patterns while keeping learning complexity at a low level. An ESN is a discrete time, continuous state, recurrent neural network using a sigmoidal activation function for all neurons.Figure 1 shows the typical ESN that will be used in this paper: the input layer is totally connected to the hidden layer, both the hidden and input layers are totally connected to the output layer. Moreover, the output layer is connected backward to the hidden layer. All weights are randomly set once for all except weights on arcs *connected towards* the outputs - those will be learnt. The hidden layer, or *reservoir*, is also randomly generated: N neurons are randomly connected up to a user-defined density of connections ρ. The weights of those connections are randomly uniformly set in $[-1, 1]$, and are scaled so that the spectral radius of the connection matrix is less than a given value $\alpha < 1$, ensuring that the newtork exhibits the "echo state property", i.e. stays out of the chaotic behavior zone whatever the input sequence (see e.g. [11]). An ESN is thus determined by the 3 parameters N, ρ and α. In some sense, an ESN can be seen as a universal dynamical system approximator, which linearly combines the elementary dynamics contained in the reservoir [15]. ESN have been shown to perform surprisingly well in the context of supervised learning, in particular for problems of prediction of times series, though it has also been successfully used in the context of (supervised) robot control learning [16] (also see [1] for an overview of ESN applications).

Figure 2 illustrates the classic learning protocol for ESN using simple linear regression as learning algorithm. The task is to learn $sinus(n/5) \rightarrow 1/2 * sinus^7(2PI/10PI)$ (i.e. one input node, one output node). Parameters are described in [10]: **(1)** the first 100 steps are not recorded but are still useful to stabilize the reservoir internal dynamics; **(2)** the next 200 steps are recorded, at each new step, the correct output value is written to the output nodes (this is called "teacher forcing"); **(3)** At step 300, a simple linear regression is performed to compute weights from arcs going to output nodes; **(4)** the network is then evaluated during the 200 next steps - as shown here, matching is nearly perfect (error is below 10^{-7}, averaged on 11 runs).

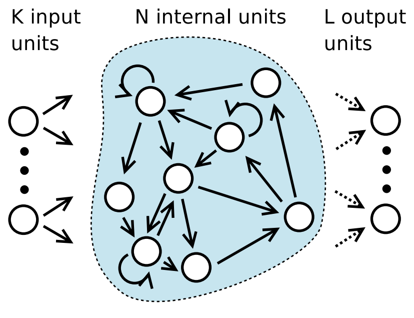

Fig. 1. Schematic view of an Echo State Network. Plain arrows stand for weights that are randomly chosen and remain fixed, while dashed arrows represent the weights to be optimized.

This figure, as well as all other results shown in the following of this paper, have been obtained using our home-made ESN implementation into the freely available open-source PicoNode Neural Network library[1][8].

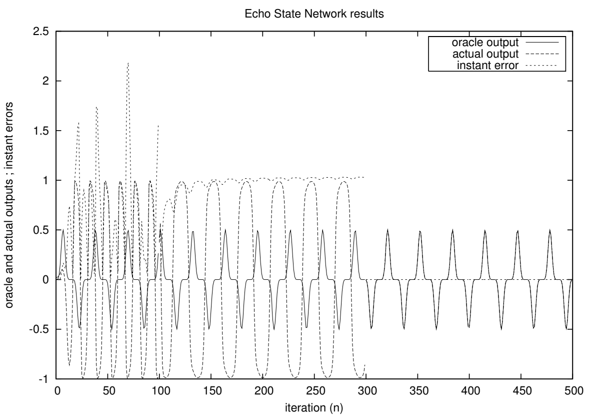

Fig. 2. Classic learning and testing protocol for ESN using PicoNode. Learning is performed at step 300. Parameters are similar to [10].

IV. Robot learning by demonstration using ESN

The Learning by Demonstration task is defined as the following. Firstly, the teacher takes control of the robotic agent (e.g. through a joystick) and perform a set of demonstrations with a given objective in mind. This results in a *log* of sensors and actuators states for each time steps. In practical, this log may eventually contain noisy and incomplete data

[1]simbad.sourceforge.net

for a difficult task. Relying on learning may still be an option as long as there is redundant information in the data log (e.g. noisy sensors in a dynamic environment may be compensated by showing the task several times). Of course, finding an optimized controller depends on the quality of the demonstrations, the difficulty of the task, the ability of the controller representation formalism, the amount of the available data and the ability for the learning algorithm to capture relevant information.

As seen in the previous section, some works have focused on using discrete or continuous time recurrent neural network in the context of Robot learning by demonstration. In this paper, a simple Echo State Network acts as the robot controller (formally, $f_{esn}(sensoryinputs) \rightarrow actuatorvalues$. The parameters used in these experiments are the following : 144 inputs (eg. 8x6 image, h,s,b values), 100 neurons in the reservoir and 2 output neurons. The connection density is 5%, the damping value is 0.8 and the noise value for regression algorithm is 0.1. The weights are drawn in $[0.5 : 0.5]$ range, and the output neurons uses Hyperbolic tangent. As shown, ESN Parameters are classic parameters except for the lack of backward connections from the output nodes to the reservoir. Indeed, a demonstration should capture only a biased, incomplete experience of the environment, implying that output actuator values should depend on the present and past sensory input values, but *not* the other way round (e.g. robot may encounter a wall anytime in a new environment).

Considering learning, the classic simple regression algorithm described in [11] is performed. In this setup, one single demonstration sequence is presented several times to the ESN and then regression is performed considering reservoir states and output values at each step. The demonstration sequence may (and should) include several sub-parts where the task is shown several times so as to cope with environment variability. During exploitation, this sequence will be considered as the temporal database from which the controller fetches correspondance with the actual context. The important point is that the sequence in itself is relevant from the temporal viewpoint since this is an important information regarding the environment. Of course, the richer the demonstration sequence, the more robust the controller should be.

V. EXPERIMENTAL RESULTS

A. *Methodology*

All experiments are performed on a Khepera II with a 2D color camera using a pad controller. Both the 8 IR sensor values, video rgb values and left/right motor actuator values are recorded during demonstration. Video image is downsized to a 8x6 resolution grid[2], which is more robust and less sensitive to noise than the full original image, and pixel hue, saturation and brightness is used instead of RGB.

The task considered in the following consists in (1) finding and then (2) getting closer to a possibly moving red ball (i.e. also implies target following). The supervisor demonstrates the task by going forward slowly and turning on the left until the red ball can be seen by the robot and then going straight forward in the direction of the target, always trying to keep the ball in the middle of the video image. Figure 3 shows the experimental setup.

[2]results not shown here were performed with other resolutions (16x12, 4x3), but no significant differences appeared compared to using 8x6 resolution.

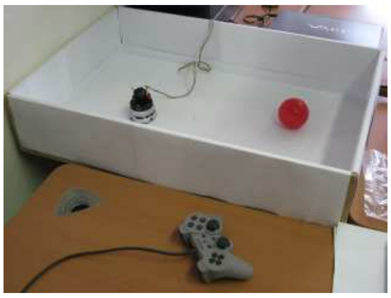
Fig. 3. Experimental setup

Validation is performed in two steps: **(1)** Off-line validation: two demonstrations are obtained with a real Khepera robot. The first is used for learning and the second for validation (compute the squared error), as it is usually the case in the standard Machine Learning scheme. This validation method makes it possible to estimate the accuracy of the controller as if it was a classifier, not an agent performing a task ; **(2)** On-line validation: the robot is evaluated (empirically, by the human supervisor) in the real world with regard to the objective task according to a validation protocol (described later). The main point of this validation step is to evaluate the agent sensory-motor coupling w.r.t. task.

In order to evaluate the ESN controller approach, all experiments are also performed with two reference algorithms: a naive non-recurrent perceptron with hidden layer and MPL[9], a learning by demonstration algorithm from the literature that is known to be efficient in the context of a mobile robotics with video camera and poor sensor quality. MPL couples a simple yet very efficient image stochastic feature selection algorithm through random sampling of just a few features in the images with a simple correlation-based learning algorithm. This algorithm was evaluated and validated on real world Pioneer 2 DX mobile robot platform with video camera for a set of navigation tasks (including target following and slaloming), which is close to the experimental setup described in this paper.

B. *Off-line experiments*

Perceptron is trained using the classic back-propagation algorithm. MPL sampling considers using 4 random pixels on each frame - other values have been tried, but using only 4 appeared as roughly equivalent to using the whole image in term of performance, in much less computation time. MPL provides an ensemble of weighted rules which can be tested on the validation set. As for the ESN approach, each log is submitted 4 times, that is 1900 to 3520 steps, from which is performed a simple linear regression. As stated before, ESN does not feature backward connections from output nodes to reservoir[3]. A simpler version of ESN without recurrence is also evaluated, i.e. a multi-layer neural network with random weights between inputs and reservoir and no recurrent connection. Only the weights linking reservoir to outputs are learned.

Two setup are considered: firstly, sensory input is limited to a single value that is the horizontal localization of the target (a red ball). secondly, a 2D video 8x6 color image with hue, saturation and value is considered (i.e. 144 input nodes). Two outputs (left and right motors) are always considered. Regression errors are shown in table 4. Two experimental scenario are considered using two demonstration logs : log no.1 is used for learning and log no.2 for testing, then the other way round is considered. All experiments are performed as described previously and each experiments is repeated 11 times.

The ESN approach gives the best results and tends to behave very well in higher dimensions. MPL performs quite well, which is not surprising since it was designed to be particularly efficient in dimension reduction where there is a great amount of data with high redundancy[4]. This is not the case with Perceptron however: it encounters difficulties as number of dimensions grow, lacking dedicated feature selection as does MPL. As for ESN, a possible explanation for the ability to deal with growing input nodes may rely in the fact that learning weights is only concerned with connections between reservoir and outputs. The randomly set input-to-reservoir weights can be compared to feature selection by reweighting[5]. The close results obtained with and without recurrent connections do seem to vote in favor of the feature selection hypothesis rather than the dynamic property of standard ESN for this problem.

C. On-line experiments

The regression error is not sufficient to evaluate controllers as it does not capture the features relevant for sensory-motor coordinated actions, especially under-represented yet very important situations. In order to evaluate precisely the acquired behaviors, we need to proceed in real-world evaluation. The evaluation protocol is thus defined as two complementary experiments : **(exp.1)** Evaluates wrt. finding and reaching the target: The robot is placed in the environment with a red ball target approximately 90 degrees on the left of the robot at approximately 15 to $20cm$. The task is considered accomplished if the robot comes near the target by 0 to $2cm$. Once reached or not (i.e. robot stops moving), the ball is moved to another position with the same settings according to the current position of the robot. The overall success rate is then computed over 11 controllers times 11 tries each (i.e. all results are averaged on a total of 121 runs) ; **(exp.2)** Evaluates wrt. time taken to reach the target from three different initial positions with different orientations: the robot is placed at $23cm$ from the target, with different orientations (ie. having the target on its left, right or rear). Each experiments were repeated 3 times for each of the 11 controllers.

The first consideration is that the basic neural network approach could not be considered because controller learnt off-line failed to produce any relevant behavior in the real world. Further experiments (not shown here) revealed that perceptron-based controller are hardly able to locate the target and react (slowly) only if placed very close to the target.

Regarding the first experiment, the Echo State Network success probability is 73% with recurent connections and 70% without, while MPL success rate is 67% (success is finding and reaching the target). While the MPL architectures show very regular behavior, ESN controllers showed different behaviors. In particular, a few number of controllers rotate the other way round to look for the target, and even one (failed) controller never tried to turn at all. However, all controllers go forward to reach the target once it is found.

Results for the second experiment are shown in table 5. MPL gives regular results while in some cases it failed to ever reach the target. On the contrary, ESN controllers could perform the task in all the conditions but showed high variance regarding time elapsed before target was reached.

An interesting remark can be noted about the difference between positions 2 (left) and 3 (right): both display the same starting position but with either a 90 or -90 degrees initial rotation away from the target, which is objectively the same wrt. distance to the target. However, position 3 results in a much higher time to reach the target for all experiments. This is due to the fact the human supervisor performed all demonstrations by always rotating on the left when looking for the target, thus encoding a (not-surprising) bias in the exploration behavior.

Algorithm	position 1	position 2	position 3
MPL	57s to 80s	9s to 10s	never reach
ESN	11s to 32s	7s to 8s	12s to 19s

Fig. 5. positions 1 to 3 are target on : rear, left, right

D. Discussion about the experiments

While perceptron is the worst, but not that bad, in the off-line experiment, it failed to provide any results in the on-line experiments. In fact, the not-so-bad result with perceptron for the regression task can be explained by the very distribution of examples. As shown in figure 6, the distribution of the inputs are highly biased depending on the demonstration, which implies over-sampling for particular values. This could be enough to provide good regression results as long as the over-sampled examples are easy to classify, which is actually the case. For example, by carefully looking at the data log, it can be observed that for the vast majority of the example where the target is in the center of the image, the given action is to

[3]Indeed, if these connections are actually activated, learning may fail with a probability of approx 0.1.

[4]This is also why MPL was not tested with the simple X target coordinate setup since its main feature is that of dimension reduction

[5]This is more feature selection than dimensionality reduction since all arcs are randomly weighted but none are pruned.

Algorithms	demo 1, learning	demo 2 testing	demo 2 learning	demo 1 testing
Using x coordinate as input				
Perceptron	$1.9.10^{-2} \pm 4.0.10^{-4}$	$1.8.10^{-2} \pm 9.0.10^{-4}$	$1.9.10^{-2} \pm 8.7.10^{-4}$	$1.6.10^{-2} \pm 4.8.10^{-4}$
ESN	$0.9.10^{-2} \pm 9.7.10^{-4}$	$1.1.10^{-2} \pm 6.2.10^{-4}$	$0.7.10^{-2} \pm 4.2.10^{-4}$	$2.1.10^{-2} \pm 3.5.10^{-3}$
Using 8x6 image as input				
Perceptron	$2.0.10^{-2} \pm 1.6.10^{-3}$	$2.2.10^{-2} \pm 2.4.10^{-3}$	$2.2.10^{-2} \pm 2.6.10^{-3}$	$1.8.10^{-2} \pm 1.6.10^{-3}$
MPL	$4.0.10^{-3} \pm 4.2.10^{-5}$	$0.9.10^{-2} \pm 9.5.10^{-5}$	$6.0.10^{-3} \pm 3.7.10^{-5}$	$0.6.10^{-2} \pm 9.7.10^{-5}$
ESN	$4.0.10^{-4} \pm 3.5.10^{-5}$	$0.6.10^{-2} \pm 1.7.10^{-3}$	$6.1.10^{-4} \pm 4.7.10^{-5}$	$0.4.10^{-2} \pm 1.2.10^{-3}$
ESN without recurences	$3.9.10^{-4} \pm 1.6.10^{-5}$	$0.5.10^{-2} \pm 1.5.10^{-3}$	$5.2.10^{-4} \pm 3.0.10^{-5}$	$0.4.10^{-2} \pm 1.3.10^{-3}$

Fig. 4. Learning and testing errors averaged over $N = 11$ runs

go straight-forward, which actually is the classifier's choice. As a consequence, crucial yet not as represented examples are not taken into account such as examples related to finding the target.

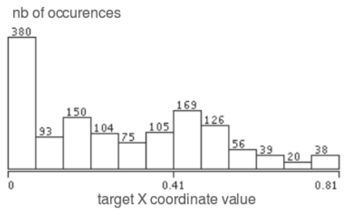

Fig. 6. Sensory space and example distribution for the one input setup (X target coordinate). this figure plots the number of occurences of the target depending on the horizontal sensory space of the robot for all examples (e.g. target in the center a step *i* adds one occurence to the bar in the middle).

MPL, which also lacks memory, is a still better choice than the perceptron in the off-line setup and much better in the on-line setup. The main difference is that MPL does consider each log example separately and is less subject to over-generalization than perceptron. This is due to the learning algorithm, specificaly designed for learning by demonstration purpose, which somewhat automaticaly re-weight under-represented examples (while perceptron naturally doesn't - except if combined with a resampling method such as, e.g. boosting[18]).

The ESN approach provides the best results, for both off-line and on-line experiments. The difference is even more relevant in the real-world validation experiment. As MPL, ESN is able to capture small one-time change in the log but thanks to another feature. Comparing standard ESN and similar network without recurrent connections, it shows that for the problem at hand, the feature selection feature due to randomly set weights between the inputs and reservoir is more important than the ability to be able to deal with temporal sequence. Indeed, such a feature can be related to random projection in dimensionality reduction except that connection are randomly weighted rather than pruned. Nevertheless, ESN is particularly well fitted for time sequence prediction, i.e. internal dynamics are trained to predict forthcoming one-time change in the environment. This means that even a one-time sensor-motor specific coupling is considered as an important information since it may be formulated as all previous steps were considered only to predict this one time change.

VI. Discussion: ESN and robot programming by demonstration

In previous section we have shown that ESN provides interesting results for demonstration. In the following, several insights and analysis are given to advocate the fact that ESN may be well-fitted for such a task.

While it is known that ESN are reliable time sequence predictor, the setup of Robot programming by Demonstration differs slightly from time sequence prediction. Indeed, the provided time sequence does not represent a canonical behavior but rather a set of useful sub-behaviors (e.g. looking for target, following target, avoiding obstacle - all packed in one single data log), possibly shown several times with some variations and, possibily, involontary glitches added by the user. As a consequence, an ESN-based robot controller can be seen as behaving as a quick time-series sequence *offset* matching where the target sequence offset is itself perturbated from time to time (i.e. actual perception wrt. offset value in recorded sequence). So a desired property is good/quick re-convergence property to the new offset in the sequence (as long as the demonstration sequence in itself is representative of what will be encountered in the environment).

To illustrate this property, we show a typical example in figure 7 of what happens when we add strong white noise perturbation to the input after learning. Experimental setup is the same (classic) setup shown for the experiment in section III (sinus prediction problem). At iteration 500, learning has been performed and correctly evaluated as shown previously in figure 2. Then, white noise is added to the inputs for 100 iterations. Between iterations 600 and 800, we evaluate recovery from noise, which is quick and perfect. Then, a new (punctual, this time) perturbation is added at iteration 800 that is an offset jump of half the phase length of the signal in the input sequence. Then again, starting iteration 801, recovery is evaluated. Same as before, recovery is performed (in even shorter time). In both case recovery is perfect (96/100 experiments[6]), i.e. error is the same as error during validation

[6]Regarding the 4% of failed experiments, a greater amount of noise during learning may solve this issue.

in figure 2 (steps 300-500).

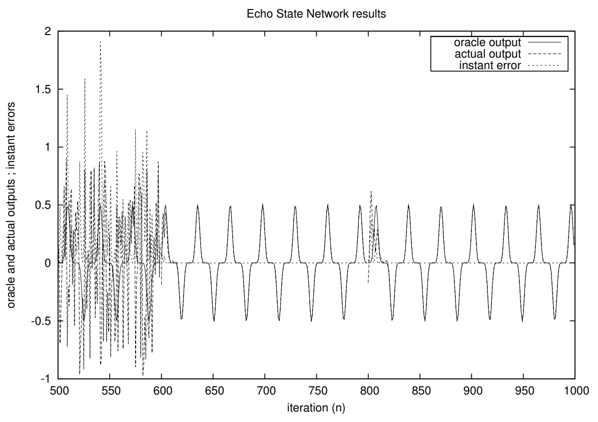

Fig. 7. Perturbating an ESN: (1) recovery from 100 steps white noise perturbation ; (2) recovery from $1/2$ phase offset shift perturbation.

Thus, recoveries from white noise perturbation and offset perturbation both result in illustrating the existence of a strong basin of attraction (as long as the input is relevant, i.e. included in the original demonstration). This basin of attraction depends on the actual input sequence rather than the initial condition, which is a useful feature when it comes to rapid offset shifting as it is usually needed in robot programming by demonstration. As a consequence, initial weight range for connections between input nodes and reservoir should be large enough to provide strong influence from the input sequence to the reservoir dynamics. Moreover, in all experiments with ESN shown in the previous section, backward connections from the output nodes to the reservoir where discarded, which also indirectly impacts on the influence of the input sequence.

VII. Conclusions and Perspectives

Echo State Networks can be trained as efficient time series predictors, with only a simple linear regression algorithm. Moreover, it is rather straight-forward to apply ESN in the scope of mobile robot behavior acquisition by demonstration. Indeed, we showed in this paper that results with an ESN-based controller are competitive with that of a state-of-the-art algorithm.

Even more important, we showed that ESN display very good behavior with regards to both feature selection and relevant temporal information acquisition, which are two key features regarding the task at hand. Firstly, ESN learning algorithm is able to indirectly perform feature selection by ignoring reservoir nodes that are too strongly influenced by non relevant sensor inputs - we showed that this feature alone may even be enough for simple tasks (i.e. no recurrent connection is needed). Secondly, ESN damping property makes it possible to quickly match the actual output sequence with desired output sequence when the input sequence break temporal coherence. This property is even more grounded if backward connection from output nodes to reservoir are suppressed. Yet, this remains to be studied deeper in the context of tasks defined as a sequence of sub-tasks.

We are currently concerned with addressing more complex tasks requiring the ability to deal with temporal sequence such as slaloming in a real world environment with many objects. Another promising extension of this work is to deal with interactive learning when the robot produces new behaviors that are evaluated by the human teacher, making it possible to refine parts of the initial demonstration.

References

[1] *NIPS 2006 Workshop on Echo State Networks and Liquid State Machines*, 2006. `http://www.esn-lsm.tugraz.at/`.

[2] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proc. 14th International Conference on Machine Learning*, pages 12–20. Morgan Kaufmann, 1997.

[3] A. Billard, K. Dautenhahn, and G. Hayes. Experiments on human-robot communication with robota, 1998.

[4] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering Optimal Imitation Strategies, in robotics. 47, 2004.

[5] A. Billard and R. Siegwart. Special issue on robot learning from demonstration. *Robotics And Autonomous Systems*, 47(2-3):65–67, 2004.

[6] S. Calinon and A. Billard. Learning of gestures by imitation in a humanoid robot. In *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*, pages 153–177. Cambridge University Press, K. Dautenhahn and C.L. Nehaniv edition, 2007.

[7] K. Dautenhahn. The art of designing socially intelligent agents: science, fiction and the human in the loop. *Applied Artificial Intelligence Journal*, 1(7), 1998.

[8] L. Hugues and N. Bredeche. Simbad : an autonomous robot simulation package for education and research. In *Proceedings of The Ninth International Conference on the Simulation of Adaptive Behavior (SAB'06)*, pages 831–842., Rome, Italy, 2006.

[9] L. Hugues and A. Drogoul. Synthesis of robot's behavior from few examples. In *IEEE/RSJ International Conference on Intelligent Robots and Sytems, IROS'02*, 2002.

[10] H. Jaeger. The echo state approach to analysing and training recurrent neural networks. Technical report GMD report 148. German National Research Center for Information Technology, 2001.

[11] H. Jaeger. A tutorial on training recurrent neural networks. covering bptt, rtrl, ekf, and the echo state network approach. GMD report 159. German National Research Center for Information Technology, 2002.

[12] D. Katagami and S. Yamada. Real robot learning with human teaching. In *The4-th Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*, pages 263–270, 2001.

[13] M. N. Nicolescu and M. J. M. c. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *In Proceedings of the Second International Joint Conference on Autonomous Agents and MultiAgent Systems*, 2003.

[14] S. Nolfi and D. Floreano. Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines. Cambridge, MA: MIT Press/Bradford Books, 2000.

[15] M. C. Ozturk, D. Xu, and J. C. Principe. Analysis and design of echo state networks,. In *Neural Computation, 19:1*, pages 111–138. MIT Press, 2007.

[16] M. Salmen and P. G. Ploger. Echo state networks used for motor control. In *In Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1953–1958, 2005.

[17] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *HRI '06: Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006.

[18] R. Schapire. A brief introduction to boosting. In *In Proceedings of International Joint ConferenceonArtificial Intelligence*, pages 1401–1405, 1999.

[19] R. Sutton and A. Barto. Reinforcement learning: An introduction. MIT Press, 1998.