Proceedings of the 24th IASTED International Conference
MODELLING, IDENTIFICATION, AND CONTROL
February 16-18, 2005, Innsbruck, Austria

A FIXED-WEIGHT RNN DYNAMIC CONTROLLER FOR MULTIPLE MOBILE ROBOTS

Mohamed Oubbati, Paul Levi, and Michael Schanz
Institute of Parallel and Distributed Systems, University of Stuttgart
Universitaetsstrasse 38, D-70569 Stuttgart, Germany
{Mohamed.Oubbati, Paul.Levi, Michael.Schanz}@informatik.uni-stuttgart.de

ABSTRACT
In this paper, we demonstrate the ability of a single fixed-weight RNN to act as a dynamic controller for several (here 3) distinct wheeled mobile robots, without exact knowledge about their dynamics parameters. The controller is properly trained to exhibit adaptive behaviour after its weights have been fixed. This capability is a natural consequence of prior 'meta-learning' used recently in the area of RNN.

KEY WORDS
Mobile robot, Recurrent Neural Networks, Meta-learning, and Adaptive control

1. Introduction

Recently, there has been interest in the observed ability of RNNs with fixed weights to model multiple nonlinear systems. It has been shown by Feldkamp et. al.[1][2] that a single fixed weight RNN can perform one-time-step prediction for many distinct time series. In the control domain, it is shown in [3] that a RNN can be trained to act as a stabilizing controller for three unrelated systems and to handle switch between them. This capability is acquired through prior training; instead of learning data from one system, the network was able to learn from several different systems. A theory explanation about this property can be found in [4][5]. The adaptive behaviour of RNNs with fixed weights is named differently. It is termed "meta-learning" in [6], and "accommodative" in [7]. Such "multiple modelling" capabilities of RNNs are potentially useful in mobile robotics control where fast adaptive behaviour of the controller is required.
In our previous work [8], a novel RNN called Echo State Network (ESN) is used to develop a dynamic controller for mobile robots, and implemented successfully on a real omnidirectional robot. In this paper, the meta-learning approach is used to develop an adaptive ESN controller for multiple mobile robots. Adaptation here is the ability of the resulting fixed-weight ESN to recognize the robot parameters variations only through its inputs, and to adjust its behaviour to these changes, without changing any synaptic weight. An advantage in ESN is that no *multi-stream* training is needed, since its training algorithm uses all data for a single time, and does not suffer from the *recency effect*. This property gives the ESN much faster learning times in meta-learning comparing with the networks used in [2][3][9].
Computer simulations will show high performances of the fixed-weight controller to balance between the variety of the reference velocity and the variety of the robots.

This paper is organized as follows. Section 2 introduces the problem to solve. Meta-learning procedure, including training approach of the ESN controller, is described in Section 3. In section 4, simulation results are presented. Finally, discussion and conclusion are drawn in section 5.

2. Problem Statement

The problem to solve is that of developing a single robust dynamic controller for three distinct nonholonomic mobile robots. We adopt here the model presented by Takanori et al. [10]. Each robot has the same model structure, but their specific parameters result in quite different behavior. The mobile robot considered here is shown in Figure 1. P_0 is the origin of the coordinate system and the middle between the wheels. The distance from P_0 to the center of mass P_c is d.

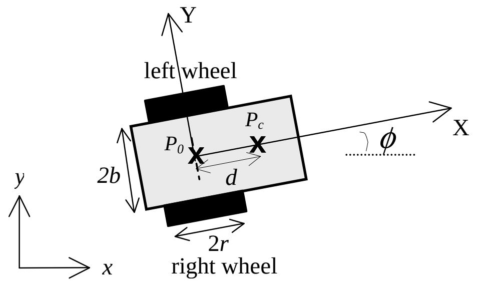

Figure 1. Mobile robot with two actuated wheels

The geometric configuration of the robot is described by $q = [x, y, \phi, \theta_r, \theta_l]^T$ where (x,y) are the coordinates of P_0, ϕ is the heading angle of the mobile robot, and θ_r, θ_l are the angles of the right and left driving wheels, respectively.

Assuming that the wheels roll and do not slip, the kinematic and the dynamic model are given by equations (1) and (2), respectively.

$$\dot{q} = S(q)v \tag{1}$$

$$M(q)\dot{v} + V(q,\dot{q})v = B(q)\tau \tag{2}$$

$v = [\dot{\theta}_r \ \ \dot{\theta}_l]^T$

$\tau = [\tau_r, \tau_l]^T$: Torques applied on right and left wheels.

$$S(q) = \begin{bmatrix} \frac{r}{2}\cos\phi & \frac{r}{2}\cos\phi \\ \frac{r}{2}\sin\phi & \frac{r}{2}\sin\phi \\ \frac{r}{2b} & -\frac{r}{2b} \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} 0 & \frac{r^2}{2b} m_c d\dot{\phi} \\ -\frac{r^2}{2b} m_c d\dot{\phi} & 0 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{r^2}{4b^2}(mb^2 + I) + I_w & \frac{r^2}{4b^2}(mb^2 - I) \\ \frac{r^2}{4b^2}(mb^2 - I) & \frac{r^2}{4b^2}(mb^2 + I) + I_w \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The mass of the body and that of a wheel with a motor are m_c and m_w respectively. I_c, I_w, and I_m are the moment of inertia of the body about the vertical axis through P_c, the wheel with a motor about the wheel axis, and the wheel with a motor about the wheel diameter. I and m are given by:

$I = m_c d^2 + I_c + 2m_w b^2 + 2I_m$,

$m = m_c + 2m_w$.

In this work, the width b, the distance d and the wheel radius r are chosen to be different and specific for each of the three robots. Table 1 lists their values, where we label the robots by I, II, and III. The other parameters values are the same for all robots and chosen as in [10].

	b	d	r
Robot I	0.4	0.1	0.05
Robot II	0.75	0.3	0.15
Robot III	0.3	0.2	0.25

Table 1. Robots specification

Our goal here is to train a *single* ESN to act as a dynamic controller for the robots (I, II, and III), in order to track predefined reference linear and angular velocities. Upon completion of the training, we expect that the ESN controller will be capable of detecting the identity of the robot, only from the robot output in combination with its own state, without changing any synaptic weight. Furthermore, we also desire that the trained network be capable to minimise reasonably errors between the reference and the robots velocities, without knowledge about their dynamic-model parameters.

3. Procedure

RNNs have a great potential for "black box" modelling of non-linear dynamical systems, and they can give complementary/new solutions for identification and control. However, only few results have been published regarding the non-linear system control by RNNs. The major reason is because simple and powerful training algorithms were missing. Echo State Network is a novel RNN in a form of "*Dynamic Reservoir*"(DR), which contains a large number of sparsely interconnected neurons with non-trainable weights (Figure 2). ESN has an easy training algorithm, where only the reservoir-to-output weight connections are to be adjusted using a linear MSE minimization.

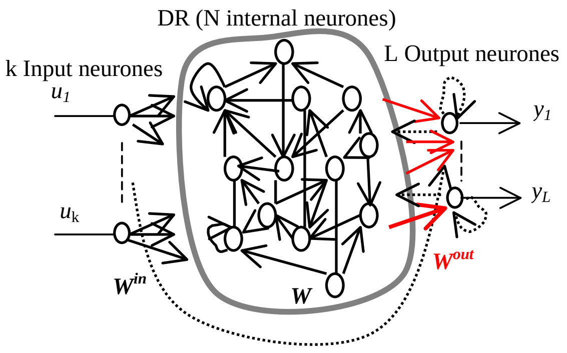

Figure 2. Basic architecture of ESN: *Only* the weights connections (W^{out}) from the internal neurons to the output are to be adjusted. Dotted arrows indicate connections that are possible but not required.

As presented in Figure 2, activations of input neurons at time n are $u(n)=(u_1(n),\ldots,u_k(n))$, of internal units are $x(n)=(x_1(n),\ldots x_N(n))$, and of output neurons are $y(n)=(y_1(n),\ldots y_L(n))$. The activation of internal and output units is updated according to:

$$x(n+1) = f(W^{in}(u(n+1) + Wx(n))) \tag{3}$$

$$y(n+1) = f^{out}(W^{out}(x(n+1), y(n))) \tag{4}$$

where f and f^{out} are vectors of sigmoid functions.

Here is a brief presentation of an offline algorithm for the entire learning procedure.

1. Generate randomly the matrices (W^{in},W), scaling the weight matrix W such that its maximum eingenvalue $|\lambda_{\max}| < 1$ obtains.
2. Drive the network using the training input/output data and equation 3.
3. Collect at each time the state $\mathbf{x}(n)$ as a new row into a state collecting matrix **M,** and collect similarly at each time the sigmoid-inverted teacher output $\mathbf{tanh^{-1}d(n)}$ into a teacher collection matrix **T**.
4. Compute the pseudoinverse of **M**. and put $\mathbf{W^{out}=(M^{-1}T)^{t}}$, t: indicates transpose operation.

The ESN is now trained. For more detailed information, a complete tutorial on ESNs can be found in [11].

Here, the ESN architecture was chosen as follows. 4 inputs (actual and reference linear and angular velocities of the robot), 17 internal neurons, 2 outputs (two torques), no back-connection from the output to the DR, and no synaptic weight connections from the input directly to the output. The input and the internal synaptic connections weights were randomly initialized from a uniform distribution over [-1, +1]. The internal weight matrix W has a sparse connectivity of 20% and scaled such that its maximum eingenvalue $|\lambda_{\max}| \approx 0.3$.

To train the ESN as a dynamic controller for mobile robots, we used the same training approach described in [8]. Figure 3 depicts a block diagram of this approach. At time n, actual and delayed angular and linear velocities $\{(w,v)_n, (w,v)_{n+1}\}$ were given as inputs to the ESN, and the correspondent torques as teacher signals. The ESN learned the teacher signals, which bring the robot from the actual (n) to the delayed (n+1) angular and linear velocities.

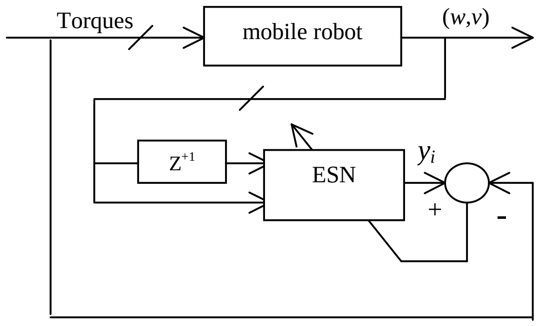

Figure 3. Training of ESN as a dynamic controller for mobile robots

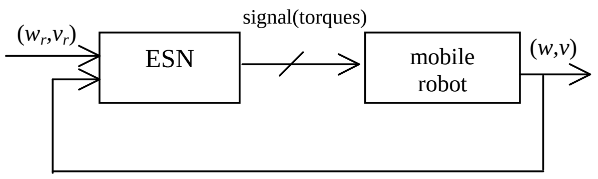

Figure 4. Exploitation of ESN as a dynamic controller for mobile robots

Training was carried out using 3000 random heterogeneous input-output sequences. Each 1000 sequences are collected from one robot. No *multi-stream* training was needed, since the ESN batch learning procedure uses all data at once, and does not suffer from the *recency effect*. After network training has been completed, the delayed velocities were replaced by the reference (desired) ones (w_r,v_r) (Figure 4).

In the simulation, we performed several tests on the behaviour of the trained network, two of which are reported here. In the first, the fixed-weight ESN controller was used to control each robot separately. In the second, we tested its capability to handle switch between the three robots. In the two cases, the reference linear and angular velocities are chosen as in [10].

4. Results

During preparation of the network, it was not easy to find its optimum parameters in order to obtain a “rich” variety of internal dynamics. Using a “relatively” large dimension (more than 30 internal neurons) the network lost stability at many times and exhibited sometimes high-frequency oscillations on smooth test signals. With small dimension (say 4-9 internal neurons), the network could not react quickly to the velocity variations. For meta-learning, an optimum size was found between 16 and 19 internal neurons, partially interconnected (20% of connectivity).

Control results for the first case are present in Figure 5 and those for the second case in Figure 6. In each figure, the panels on the left-hand side show the evolution of a robot state (linear and angular velocities), while the right-hand side panels provide the corresponding control signals (torques) given by the ESN controller.

In figure 5, the right-hand column provides the three control signals delivered by the same fixed-weight ESN controller for the three robots separately. The three signals are superposed in order to facilitate the comparison, and to have an idea about the three robots behaviors. As can be seen on the left-hand panels, the excellent velocity tracking of all robots is evident. On the predefined reference linear and angular velocities, the robots I, II, and III showed mean square errors of MSE={(1.7534e-004, 1.2619e-004), (7.9237e-005, 8.6139e-005), (0.0013, 7.3374e-004)}, respectively. Here, the fixed-weight controller did a reasonable job, and could effectively deliver the appropriate control signals for each robot. We recognize here that the controller trained for many robots may not be as effective on a given robot as a controller trained only for that robot. In [8], we trained an ESN network to control the same robot model, and we had an MSE=(5.7670e-007, 1.2057e-005). This result was obtained on other reference velocities, but we tried the same network also on the reference velocities used here, and we obtained almost the same range of MSE value.

The second case is more complicated. In this case, the controller had to handle balance between tracking the reference velocity and switch between the three robots. In

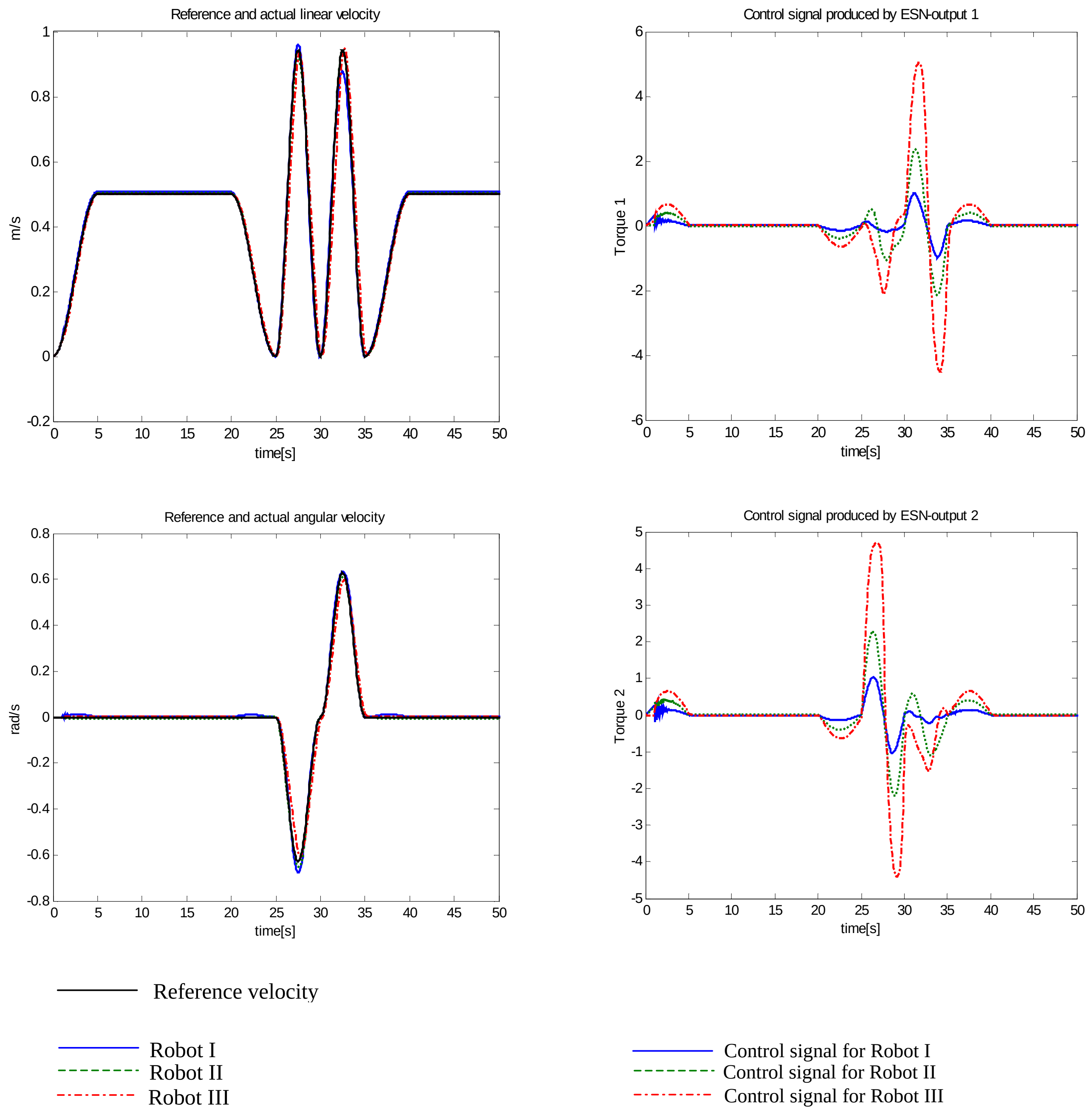

Figure 5. Linear and Angular velocity tracking (left) and controls (right). Each robot is controlled separately with the same fixed-weight ESN controller.

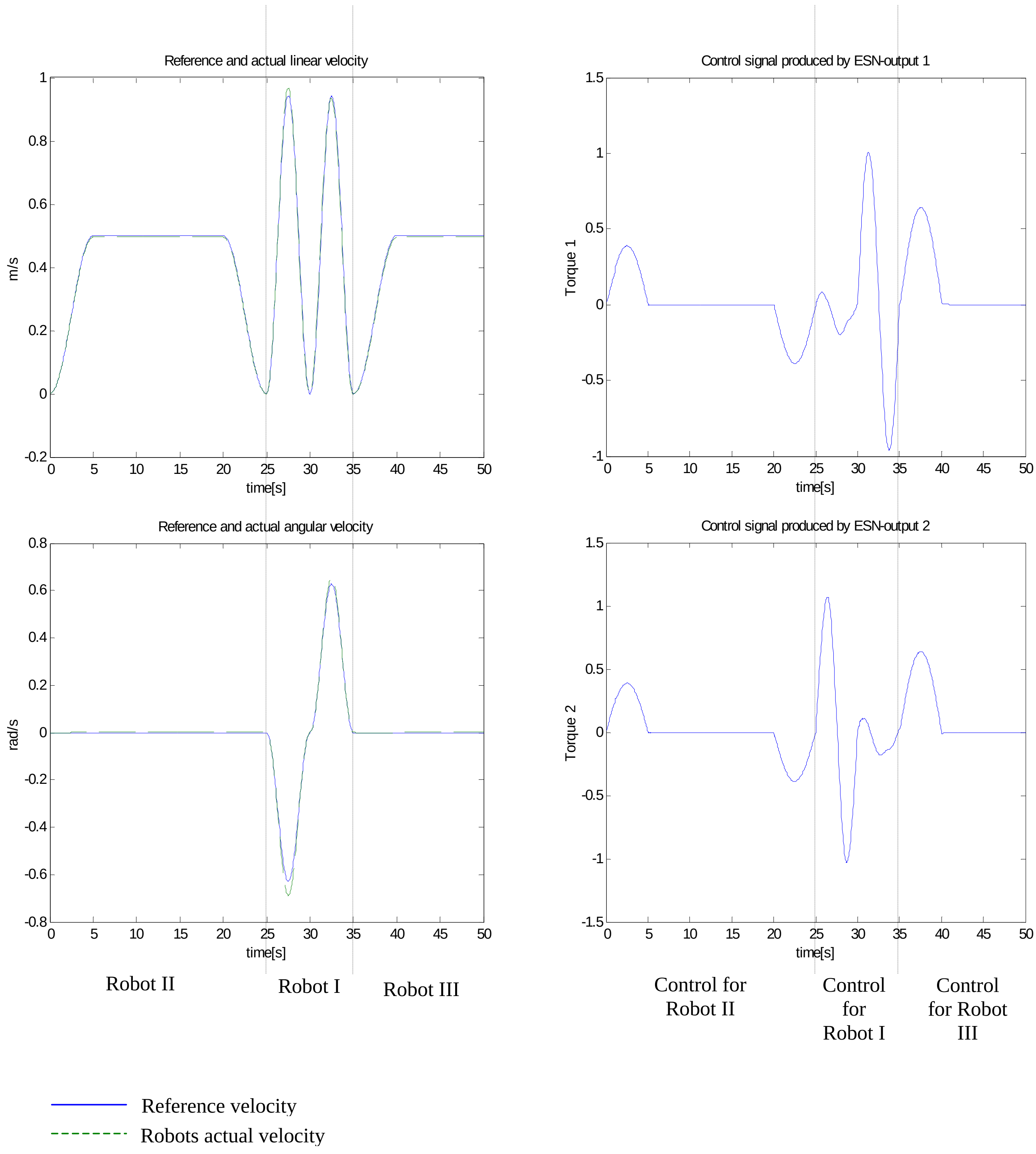

Figure 6. Linear and Angular velocity tracking (left) and controls (right). The fixed-weight ESN controls the three robots, following switches between them. The first switch occurs at time 25s from Robot II to I. The second switch occurs at time 35s from Robot I to III.

figure 6, the first switch occurs at time 25s from Robot II to Robot I, and the second one occurs at time 35s from Robot I to Robot III. A switch from one Robot to another requires the controller outputs to change, since each robot has its proper dynamic characteristics. Surprisingly, the control is barely affected by these switches. The resulting network controller showed smooth and rapid adaptation to these changes (see right-hand panels), and the three robots tracked reasonably the reference linear and angular velocities in their respective time intervals.

5. Discussion and conclusion

Here, a single ESN is being asked to control three distinct nonholonomic mobile robots. In other words, the ESN controller is asked to exhibit a characteristic, normally ascribed to adaptive controllers, whose parameters change in response to an environmental change. “Adaptation” in this work is defined as the ability of the controller to recognize change only through the robot output and its own state, without changing any synaptic weight. This capability is a natural consequence of prior ‘meta-learning’ used during training. When a “new” input (from one robot, which is already learned) is provided, the state of the network switch from one family of orbits to another, which corresponds to the new input. Also, because of the “rich” variety of its internal dynamics, the network could make an excellent generalization on new incoming data and deliver the appropriate control signals for the three robots, in order to track the reference velocity. Here the controller is effective only for those robots, for which it has been trained, not for arbitrarily chosen robot.

After training, the fixed-weight ESN controller showed a reasonable balance between the variety of the reference velocity and the variety of the robots. Furthermore, no knowledge about the robots dynamics was required, since the controller is designed only by learning their Input/Output data. This property is very important in practical cases, where it is almost impossible to have the real parameters values of a robot.

All in all, the training experiments carried out here demonstrate that a single, small and partially interconnected ESN can be trained to act as a dynamic controller for multiple distinct mobile robots. However, we are aware of a certain degree of arbitrariness in our choice of the controller network parameter and architecture. Therefore, substantial investigation on ESN architecture and more experiments on much larger data sets and real implementations are still needed to ensure that the results we have achieved to date are indeed statistically significant.

In this paper only simulation testing results are shown. Our future work is to implement and test meta-learning approach on the omnidirectional Soccer-robots team available at our Robotics Lab.

Reference:

[1] L.A. Feldkamp, G.V. Puskorius, and P.C. Moore, Adaptation from fixed weight dynamic networks, *IEEE Int. Conf. on Neural Networks*, Washington, 1996,155-160.

[2] Feldkamp L.A., G.V. Puskorius, and P.C. Moore, Adaptive behaviour from fixed weight dynamic networks, *Information Sciences* 98, 1997, 217-235.

[3] Feldkamp, L. A., & Puskorius, G. V, Fixed weight controller for multiple systems, *IEEE Int. Conf. on NN*, Vol 2, Texas, USA 9-12 June 1997, 773-778.

[4] Andrew.D. Back, Multiple and time-varying dynamic modelling capabilities of recurrent neural networks, *Neural Networks for Signal Processing 7, IEEE Press*, 1997.

[5] Andrew D. Back, Tianping Chen, Universal Approximation of Multiple Nonlinear Operators by Neural Networks, *Neural Computation* 14(11), 2002, 2561-2566.

[6] Prokhorov, D., Feldkamp, L., and I. Tyukin, Adaptive Behavior with Fixed Weights in Recurrent Neural Networks: An Overview, *Int. Joint Conference on Neural Networks*, Honolulu, Hawaii, May 2002.

[7] J. Lo, Adaptive vs. Accommodative Neural networks for Adaptive System Identification, *Int. Joint Conf. on Neural Networks*, 2001, 2001-2006.

[8] M. Oubbati, P. Levi, M. Schanz, Recurrent Neural Network for Wheeled Mobile Robot Control, *WSEAS Transaction on Systems*, vol. 3, August 2004, 2460-2467.

[9] D. Prokhorov, G. Puskorius, and L. Feldkamp, Dynamical Neural Networks for Control, *In J. Kolen and S. Kremer (Eds.) A Field Guide to Dynamical Recurrent Networks*, *IEEE Press*, 2001.

[10] T. Fukao, H. Nakagawa, and N. Adachi, Adaptive Tracking Control of a Nonholonomic Mobile Robot, *IEEE Transactions on robotics and automation*, vol. 16, n°. 5, October 2000, 609-615.

[11] H. Jaeger, Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach, *GMD*, Report 159, 2002.

[12] A.D. Back and T.P. Chen, Approximation of hybrid systems by neural networks, *Proc of Int. Conf. on Neural Information Processing, Springer-Verlag*, 1997, 326-329.

[13] S. Younger, P. R. Conwell, and N. E. Cotter, Fixed-Weight On-Line Learning, *IEEE Transactions on Neural Networks*, vol. 10, no. 2, March 1999.

[14] M-S Kim, J-H Shin, S-G Hong, J-J Lee, Designing a robust adaptive dynamic controller for nonholonomic mobile robots under modelling uncertainty and disturbances, *Mechatronics 13*, 2003, 507–519.

[15] M.L Corradini, G. Orlando, Control of mobile robots with uncertainties in the dynamical model: a discrete time sliding mode approach with experimental results, *Control Engineering Practice* 10, 2002, 23-34.