

Adaptive Shared Autonomy Using Reservoir Computing

Xavier Dutoit

Dirk Vanhooydonck

Hendrik Van Brussel

Marnix Nuttin

Mobile Learning Robots Group
Dept. of Mechanical
Engineering
Katholieke Universiteit Leuven
Celestijnenlaan 300b
3000 Leuven, Belgium

ABSTRACT

Electric wheelchairs can provide necessary help to elderly or disabled people. If the people are unable to give precise and fine steering commands, an intelligent electric wheelchair can assist the driving by implementing a collision avoidance behaviour. However, a straightforward implementation of such an algorithm is not desired as the intent of the users is not always to avoid the obstacle (if for instance they want to dock at a table or pick a book from a shelf). A possible improvement is then to provide the users with assistance adapted to their intent. The problem is then to estimate when the users actually need assistance. To solve this task, we apply here a neural network technique, Reservoir Computing, which is a new and simple yet powerful way to use Recurrent Neural Networks.

Categories and Subject Descriptors

I.2.9 [Artificial Intelligence]: Robotics

General Terms

Autonomous vehicles

Keywords

Adaptive Shared Autonomy, Reservoir Computing, Echo State Networks

1. INTRODUCTION

In a world where mobility becomes increasingly important, a significant number of elderly and disabled people still encounter huge problems to move themselves without the help of somebody else. Their limitation in mobility causes a decrease of their autonomy and possibly a decrease of their social life quality.

It is possible to overcome this problem with an electric wheelchair, which is a first step towards more independence. Nonetheless, a large group of disabled people still experience important difficulties to control their electric wheelchair. A possible solution is to equip the wheelchair with an intelligent assistance system composed by sensors and a computer, so the wheelchair control can be shared between the user and the controller. The problem is then to estimate when the human actually needs assistance. To do this, one can either use an explicit [1] or implicit [9] model of the user. In the

second approach, the model can be learned from examples, with Neural Networks for instance.

We use here Recurrent Neural Networks (RNNs), in which the recurrent loops provide the network with some memory and thus with the ability to deal with time-related data. More precisely, we use the relatively new technique of Reservoir Computing (RC) [2, 4, 7]. RC is an elegant and efficient way to use RNNs where one can obtain a computational power comparable to state-of-the-art RNNs with a much simpler training algorithm, which is linear and can be guaranteed to find the global optimum.

The main idea of RC is to project the input data into a high-dimensional space, the reservoir, and then to train a simple memoryless readout to map the state of this reservoir to the desired outputs. The reservoir itself, which usually consists of a large RNN, is not trained, thus drastically decreasing the training complexity.

RC allows to consider any task in a black-box approach. We just need to provide a dataset consisting of input data and the corresponding desired output data, and we can create a reservoir and then train a linear readout to do the mapping. It is thus well suited to estimate the user model implicitly.

One design problem with RC is that the reservoir is created randomly. It is then hard to say a priori which characteristics the reservoir should exhibit in order to have a good performance, and one typically has to try several random reservoirs and select the best one. However, it is still possible just by randomly trying only a few reservoirs to achieve a very reasonable performance on a wide variety of tasks.

2. ECHO STATE NETWORKS

An Echo State Network (ESN)[2] is a special type of Reservoir Computing architecture where the reservoir is a network of sigmoid neurons. It is described by an input matrix \mathbf{W}_i^r , a reservoir matrix \mathbf{W}_r^r and an output matrix \mathbf{W}_o^r (see Figure 1). The activity state of the neurons in the reservoir is given at time step t by $\mathbf{s}[t]$, the *state vector*, which is updated according to

$$\mathbf{s}[t+1] = f\left(\delta \cdot (\mathbf{W}_i^r \cdot \mathbf{i}[t] + \mathbf{W}_r^r \cdot \mathbf{s}[t]) + (1 - \delta) \cdot \mathbf{s}[t]\right), \quad (1)$$

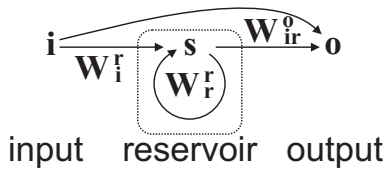


Figure 1: Structure of an Echo State Network (see text for details).

where $\mathbf{i}[t]$ denotes the input at time step t , f is a nonlinear sigmoid function (we use here a hyperbolic tangent), δ is a parameter, the *leak rate*, tuning the dynamics of the reservoir (cf. later) and the state is initialised as $\mathbf{s}[0] = \mathbf{0}$.

The output $\mathbf{o}[t]$ is given by

$$\mathbf{o}[t] = \mathbf{W}_{\text{ir}}^{\text{o}} \cdot \begin{bmatrix} \mathbf{i}[t] \\ \mathbf{s}[t] \\ 1 \end{bmatrix}. \quad (2)$$

The main advantage of the Echo State Network approach (and of the Reservoir Computing approach in general) over classical Recurrent Neural Networks is that the recurrent part itself is not trained. Indeed, the matrices \mathbf{W}_i^r and \mathbf{W}_r^r are randomly created a priori, and only the output matrix $\mathbf{W}_{\text{ir}}^{\text{o}}$ is trained.

2.1 Network Initialisation

The input and reservoir matrices are created randomly on beforehand. If we let n_i denote the number of inputs and n_r the number of neurons inside the reservoir, the input matrix \mathbf{W}_i^r is a $n_r \times n_i$ matrix where each element is $+0.1$ or -0.1 with equal probability.

The reservoir matrix \mathbf{W}_r^r is a $n_r \times n_r$ matrix where each element is drawn from a zero-mean Gaussian distribution with variance 1. The matrix is then re-scaled to make its spectral radius equal to 0.9 in order to satisfy the *echo state property* [2]. This property states, intuitively, that the initial conditions have an asymptotically decreasing influence on the current state of the network.

2.2 Time Scale

The δ parameter ($0 < \delta \leq 1$) sets the short-term memory time scale of each neuron. A δ equal to 1 means that the neurons have no short-term memory, whereas a δ close to 0 expands the memory span. Changing the short-term memory of each neuron changes the dynamics of the reservoir. It has been shown [3, 6] that a crucial point to obtain good results is to make the dominant time scale of the input data match the dominant time scale of the reservoir. One way to do so is by tuning the δ parameter (another way would be by resampling the input data at a different rate). However it is typically very hard or even impossible to determine a priori what the dominant time scale of the input is, so the usual approach is to find the optimal δ parameter or resampling factor by trial and error.

It is worth noting that having a δ smaller than 1 still guar-

antees the echo state property. Indeed, equation (1) can be rewritten as

$$\mathbf{s}[t+1] = f(\tilde{\mathbf{W}}_i^r \cdot \mathbf{i}[t] + \tilde{\mathbf{W}}_r^r \cdot \mathbf{s}[t]),$$

with $\tilde{\mathbf{W}}_i^r = \delta \cdot \mathbf{W}_i^r$ and $\tilde{\mathbf{W}}_r^r = \delta \cdot \mathbf{W}_r^r + (1 - \delta) \cdot \mathcal{I}$, \mathcal{I} being the identity matrix. In this form, a sufficient condition for the echo state property is that the spectral radius of $\tilde{\mathbf{W}}_r^r$ must be smaller than 1. This is guaranteed if the spectral radius of \mathbf{W}_r^r is smaller than one, which is the case by construction.

2.3 Training

The output matrix $\mathbf{W}_{\text{ir}}^{\text{o}}$ is the only part of the ESN which is trained. If we let n_o denote the number of outputs, then $\mathbf{W}_{\text{ir}}^{\text{o}}$ is a $n_o \times (n_i + n_r + 1)$ matrix which should satisfy:

$$\mathbf{W}_{\text{ir}}^{\text{o}} \cdot \mathbf{S} = \begin{bmatrix} \hat{\mathbf{o}}[1] & \hat{\mathbf{o}}[2] & \dots & \hat{\mathbf{o}}[n_t] \end{bmatrix}, \quad (3)$$

where $\hat{\mathbf{o}}[t]$ is the desired output at time t , n_t is the number of time step in the training set and

$$\mathbf{S} \triangleq \begin{bmatrix} \mathbf{i}[1] & \mathbf{i}[2] & \dots & \mathbf{i}[n_t] \\ \mathbf{s}[1] & \mathbf{s}[2] & \dots & \mathbf{s}[n_t] \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

is the *state matrix* of all the activation states of the neurons inside the reservoir and of the inputs.

The training procedure consists of first feeding the ESN with the input, recording the corresponding states in the state matrix, and then solving the Equation (3).

This equation is solved originally in the mean square sense. However, in order to improve the generalisation of the solution, we apply a ridge regression technique and solve instead

$$\mathbf{W}_{\text{ir}}^{\text{o}} = \arg \min_{\mathbf{W}_{\text{ir}}^{\text{o}}} (\|\mathbf{W}_{\text{ir}}^{\text{o}} \cdot \mathbf{S} - \hat{\mathbf{O}}\|^2 + \rho \cdot \|\mathbf{W}_{\text{ir}}^{\text{o}}\|^2), \quad (4)$$

where $\hat{\mathbf{O}}$ is the matrix containing all the desired outputs. The optimal ρ parameter is found by grid-search, by leaving out a validation set during training.

3. EXPERIMENT

We consider the Sharioto intelligent wheelchair depicted in Figure 2. It is an electric wheelchair driven by a joystick input. It is equipped with a laser range sensor spanning the area in front of the wheelchair. For a more detailed description of this platform, please refer to [5] and [8].

Simulation data have been created using a simulator based on the Simrob simulator¹ developed by the Bremen Institute for Safe Systems at the university of Bremen.

The wheelchair can drive at a speed ranging from -150 to 150 cm/s and can turn at a rate from -2 to 2 rad/s. Both translational and rotational velocities are set by the joystick, and a joystick input is read every 200 ms. The joystick input is modified to simulate a disability.

We use two simulated disabilities. The first one models a user who is only able to give a coarse indication of the desired direction through the joystick. The second one models a

¹available at <http://www.informatik.uni-bremen.de/simrobot/>



Figure 2: The smart wheelchair sharioto

user with a limited extension of the wrist who has then not enough strength to push the joystick towards the right (we consider users steering with their right hand).

We consider two setups: first, the simulation environment shown in Figure 3; second, a real-world environment. We used the first disability in simulation and the second one in the real world. In both cases, the data recorded are the distance sensors readings spanning the front of the wheelchair and the joystick inputs. There are 37 distance readings (in steps of 5 degrees) for the simulation setup and 20 (in steps of 9 degrees) for the real world. The joystick inputs are, in both setups, 2 values (a translational and a rotational velocities) ranging from -1 to +1.

At each time step, the instantaneous distance to collision is computed. It is defined as the distance to the nearest obstacle if the wheelchair keeps on moving with the same instantaneous translational and rotational velocities.

Two sets of experiments have been recorded in each setup. In the first set, the intent of the user is to avoid the obstacle; in the second set, it is to approach the obstacle (in order to dock at a table for instance). Due to the simulated disability, the intent of the user is not perfectly applied to the wheelchair, and the runs from both sets end up with a collision.

The first task that we consider is to predict when a collision avoidance behaviour should be activated. By construction, we consider that it should be activated for all points in runs where the intent is to avoid the obstacle and when the distance to collision is less than one metre. The second task is to predict the instantaneous distance to collision.

In an attempt to be more plausible biologically, the distance input is reversed (a high value means a very near obstacle), and the target distance to collision is then also reversed, so it is more a measure of "collision imminence" (a value close to 0 means a large distance to collision, and a value of 1

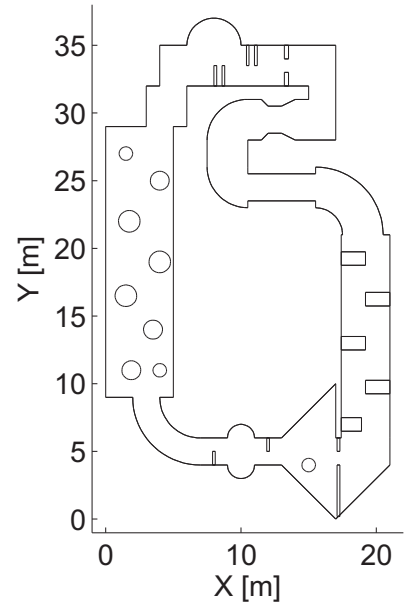


Figure 3: The map used in simulation

means a collision).

4. RESULTS

We apply now the Echo State Networks technique to predict when the user needs assistance and to estimate the instantaneous distance to collision.

In the first setup, 99 runs have been recorded, and the intent was to avoid collision in 61 of them, and to dock or approach the obstacle in the other 28. For the second (real-world) setup, 28 runs have been recorded, and in 15 of them the collision should be avoided. As the collision avoidance behaviour was disabled during the recording of the training data, all runs ended up with a collision.

As the reservoir is created randomly, the results presented here are averaged over 25 different reservoirs of 200 neurons each. The leak rate δ is set to 1 (i.e. complete leak) for the simulation and 0.3 for the real world, and the regularisation parameter ρ to $2.6 \cdot 10^{-3}$ and 14.69 (the optimal values for these parameters have been found experimentally). The high value of the regularisation parameter for the real-world setup might come from the fact that there are only 28 runs and thus it is hard to generalise on such a small dataset.

In the simulation setup, on average, an ESN is able to predict whether or not the user needs assistance 95.7% of the time, and the MSE between the actual and predicted collision imminence is 0.027. A sample plot is shown in Figure 4(a), where we consider 6 runs, 3 of which required assistance. The upper row shows whether or not assistance is required. The solid line is the target value, the dotted one the ESN output. Most of the time, the ESN output is correct. In the 3 runs which required assistance, however, the ESN is a bit too late to estimate when the assistance is re-

quired. The lower row shows the collision imminence, where a value of 1 means a collision and value tending towards 0 means a distance to collision tending towards infinity. In the last 3 runs, the ESN detects that a collision is about to occur, but it still predicts that no assistance is required, and thus that the intent of the user is to approach the obstacle.

In the real-world setup, the assistance prediction is correct on average 87.7% of the time, and the average MSE on the distance to collision is 0.025. A sample plot is shown in Figure 4(b). Six runs are shown, and the first 3 required assistance. Here too, the ESN is able to detect that in some cases no assistance is required even though a collision becomes imminent.

5. CONCLUSION AND FUTURE WORK

We applied Echo State Networks to estimate when a disabled user driving an electric wheelchair needs assistance.

The main goal of this work was to predict whether the user wants to approach or avoid an obstacle. This task is complex, as not only must the ESN detect if the wheelchair is approaching an obstacle, but it must also predict the intent of the user based on an imperfect joystick input.

We can see that the ESN is able to estimate when a collision is about to occur. Moreover, even when it is the case, the ESN does not necessarily predict that assistance to avoid the obstacle is necessary. This demonstrates that the ESN is able to estimate the true intent of the user.

This work was done off-line with pre-recorded data, so the next step is obviously to close the loop and implement an on-line ESN to predict the intent in real-time. However, the results presented here already shows that ESN is a very promising approach for service robotics.

6. ACKNOWLEDGMENTS

This work was partially funded by the FWO Flanders project G.0317.05 and partially by the European IST Programme FET Project FP6-003758.

The authors are grateful to the ELIS group for providing the RC toolbox (<http://snn.elis.ugent.be/rct>).

7. REFERENCES

- [1] E. Demeester, A. Hüntemann, D. Vanhooydonck, G. Vanacker, A. Degeest, H. Van Brussel, and M. Nuttin. Bayesian Estimation of Wheelchair Driver Intents: Modeling Intents as Geometric Paths Tracked by the Driver. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5775–5780, Beijing, China, 2006.
- [2] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, 2001.
- [3] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert. Optimization and applications of Echo State Networks with leaky integrator neurons. *Neural Networks*, 2007.
- [4] W. Maass, T. Natschläger, and H. Markram. Real-Time Computing Without Stable States: A New Framework

- for Neural Computation Based on Perturbations. *Neural Computation*, 14:2531–2560, 2002.
- [5] M. Nuttin, E. Demeester, D. Vanhooydonck, and H. Van Brussel. Obstacle Avoidance and Shared Autonomy for Electric Wheel Chairs: An Evaluation of Preliminary Experiments. *Robotik 2002*.
- [6] B. Schrauwen, J. Defour, D. Verstraeten, and J. Van Campenhout. The introduction of time scales in Reservoir Computing, applied to isolated digits recognition. In *ICANN*, 2007.
- [7] J. J. Steil. Backpropagation-Decorrelation: online recurrent learning with O(N) complexity. *Proc.IJCNN*, 1:843–848, 2004.
- [8] D. Vanhooydonck, E. Demeester, M. Nuttin, and H. Van Brussel. Shared Control for Intelligent Wheelchairs: an Implicit Estimation of the User Intention. In *Proc. of ASER 2003*, pages 176–182, 2003.
- [9] D. Vanhooydonck, E. Demeester, G. Vanacker, A. Huentemann, J. Philips, H. Van Brussel, and M. Nuttin. Adaptable Navigational Assistance for Intelligent Wheelchairs by means of an Implicit Personalized User Model. *submitted to Robotics and Automation Society journal*.

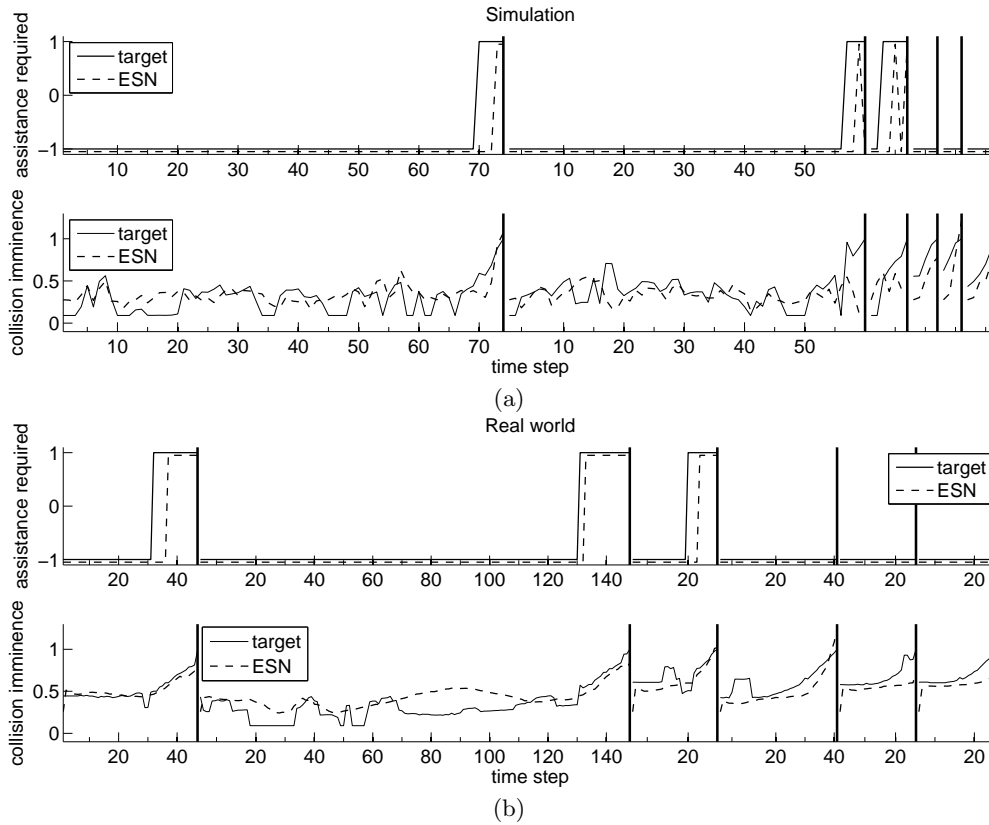


Figure 4: Simulation (a) and real-world (b) setup results. Upper row: assistance requirement; the solid line is the target, the dotted line the ESN output and the thick vertical lines mark the separation between the different runs; the ESN output is slightly shifted down for visualisation purpose; a value of 1 means that assistance is required, and -1 that no assistance is required. Lower row: collision imminence; the solid line is the target, the dashed line the ESN prediction; a value of 1 means a collision, and a value tending towards zero means a high distance to collision