Fall detections in humanoid walk patterns using Reservoir Computing based control architecture

Rahul Kanoi[1,2], Cédric Hartland[1]

[1] LRIE - EFREI, Ecole d'Ingénieur
30-32, avenue de la République 94800 Villejuif, FRANCE

[2] Vellore Institute of Technology
Vellore 632014, TamilNadu, India
rahulkanoi2006@vit.ac.in and cedric.hartland@efrei.fr

Abstract : The aim of this paper is to investigate the use of Reservoir Computing for *meta-sensor* conception. In the recent years, complex and low cost robots were designed, embedding larger amounts of various sensors, thus hindering the design of control architectures. In a similar fashion to *sensor fusion* or *smart sensors*, our approach involves generating temporal *meta-sensor* based on actual sensors. Those *meta-sensors* can be trained through evolution algorithms, holding concise meanings thus easing the control architecture designer's work.

Keywords : Reservoir Computing, Echo States Networks, Middleware and Component models for robotics, humanoid robotics

1 Introduction

A controller or its sub-parts are usually designed to consider specific low level sensor values or high level perceptions, even though adequate perceptions may be hard to obtain. One may wonder how will these control architecture's design scale with increasing number of different kinds of sensors. With *simple* robots, the question of whether a sensor is of interest for a given action can appear as obvious but it will probably not remain so in the future as robots complexity increases. In order to remain tractable, the design of control architectures will require higher level informations, with less and less focus on actual sensors.

This paper presents an approach based on Reservoir Computing (RC), involving Echo State Networks (ESN) (Jaeger *et al.*, 2007) to generate *meta-sensor* data, that might be useful to design a control architecture out of many sensors. Compared to standard neural networks, ESNs adds some memory capability, transforming sensors values an some time to a new sensor taking past data into account. Those *meta-sensors* allow one to keep the control architecture design tractable while relying on all possible sensors, whatever their numbers. While such sensors may be high level (i.e. robot fall detector), they might not be considered perceptions at all, depending on their nature.

2 Literature review

ESN involves a set of hidden neurons, referred to as reservoir, which are randomly connected together. The connection matrix is controlled from a density parameter ρ. The hidden neurons express several different dynamics due to random cycles from random connections. The idea behind RC is that the desired output can be sought as a linear combination of the hidden neurons. Formally, input and output neurons are fully connected to the hidden neurons. Some other connections (from input to output neurons, from output to reservoir neurons) can be used, but they are not considered in the present research.

Memory saturation and lifespan can be controlled if the Echo State Property (ESP) hold, i.e. the connection matrix maximal singular value σ, or damping, is below 1 (Jaeger, 2001).

The ESN training only considers the readout matrix (the weights on the connections from the reservoir to the output neurons). In the case where the reservoir offers a sufficiently rich catalogue of dynamics depending on the input sequence, the training task is to approximate the desired dynamics as a linear combination of the reservoir neurons. In the case where the output values are known, ESNs can be trained using linear regression. The main difficulty lies in the stochasticity of the approach: the reservoir dynamics cannot be predicted from the two hyper-parameters (density and damping). In this paper, due to the large amount of data considered, stochastic based evolutionary approach known as Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) is considered for the training on the readout matrix (Hansen & Ostermeier, 2001).

ESNs have been used for robotic control through literature but actual robotic applications remain rare and mainly focused on wheeled mobile robots. Ploger *et al.* (2003) have been considering ESN for both differential mobile robot and motor control. ESNs have been considered in the context of learning by demonstration from human supervision in Hartland & Bredeche (2007). An ESN controller is trained from non-reactive behaviour samples demonstrated by the human supervisor. In Antonelo *et al.* (2007), ESNs are trained so as to recognise specific patterns, enabling the mobile robot to localise to some extent. Environment pattern recognition is considered as well in Hartland *et al.* (2009) in the context of evolutionary robotics. To our knowledge, ESNs were not applied so far for humanoid robot control or sensor preprocessing.

3 Approach

The model works as follows : an ESN is embedded within the control software. This ESN can be seen as a black box model taking as many sensors as possible as input. A readout network is trained over data which represent some concept, involving one to several outputs. After training, the readout is meant to represent this concept for new given sensor inputs, providing the controller with new sensor information. One key interest is that the reservoir can be used for more than one concept. As it is randomly generated at first and only the readout connections are trained, one may add new readout connections together with new outputs. Those new outputs will not impact on the previously trained data. One bioinspired idea behind would be to use such ESNs as local nerve, providing preprocessed data based on local sensors, toward the controller.

The human controller designer need a high level concept, not knowing which sensors may be useful in order to extract such concept. Sensor data are recorded from the robot in the situations where the concept is adequate. The human designer will label the recorded data so as to represent the concept. The ESN model is then trained to discriminate those different situations and will then provide some objective criterion to assess or predict the given concept which can be included directly in the control without having to focus on the required sensors. Adding new sensors may require for a new training, but may lead to better precision for the *meta-sensor*.

Standard approach to training ESN involves linear regression algorithms, however, in this paper, as the amount of training data is quite large, for practical reasons, CMA-ES optimisation algorithm is used to train the ESN on the recorded data.

4 Experiments

As an illustration for the implementation of the model, we consider the problem of anticipating and detecting falls during a walk during which humanoid robots tend to fall[1]. Detecting falls can prove a challenging problem depending on the available sensors and expertise. The goal of the experiment is to provide the controller with a new sensor which task is to provide fall prediction from the many available sensors.

The experiments involve the Nao humanoid robot (Gouaillier *et al.*, 2009), embedding many sensors. We consider a non dynamic walk straight function available on the robot API and 15 sensor[2]. Data are recorded while the robot is walking and labelled accordingly to the final state of the robot (Fig. 1 and 2). If the walk was stable enough, the robot usually does not fall, if not, it tends to fall. With the *meta-sensor* trained over those data, the ultimate goal is to provide the controller with precise stability trends through many sensors so as to take quick decision to avoid falls.

The experiments involve training a model from training data and assess it over test data. The proposed fitness function F is the average squared sum error between the model output and the desired output from the training data $S = \{(x_0, u_0), \ldots, (x_n, u_n)\}$: $F = \frac{1}{n}\sum_{i=0}^{n} \|esn(x_i) - u_i\|^2$ where $esn(x_i)$ is the output produced by the ESN. The desired output is the final state observed over a given sequence. The model is trained using CMA-ES optimisation method.

[1]One can take the example of robotic football competition involving humanoid robots during which many robots keep on falling.

[2]1) one three-axis accelerometer, 2) two one-axis gyrometer, 3) torso angles and 4) eight foot pressure sensors (4 sensors distributed on the front-back-left-right for each foot, making 8 pressure sensors).

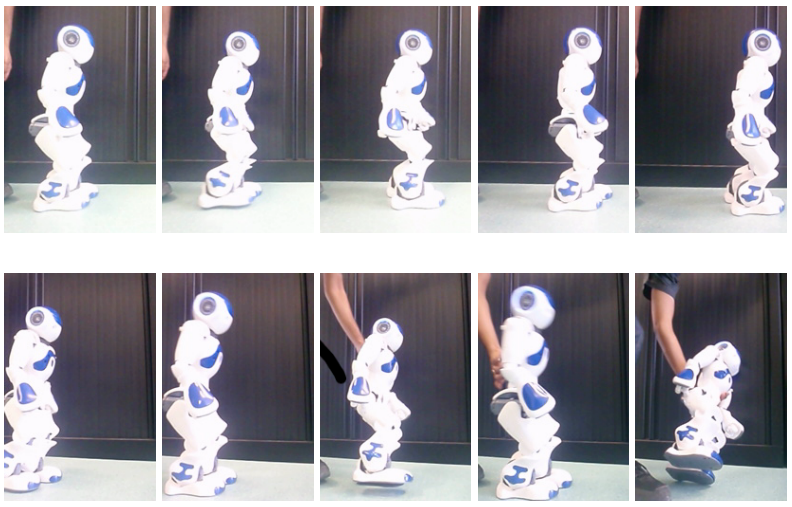


Figure 1: Top sequence displays a typical fine walk pattern on the robot. The bottom sequence shows a situation in which the robot ends falling for the same walk command.

4.1 Results

Results are provided in section 6. In the figure 3, for the target concept (green line), a 0 value indicates a no movement stable state of the robot, 0.5 is a stable walk and -0.5 indicates instability leading to fall. While learning robot has been trained to predict the no movement state as a stable state, it hence produce an output of 0.5. The plot however considers a 0 value so as to help distinguish different walk samples.

One among the best models obtained is displayed in figure 3. The model is assessed over 16 walk samples, up to 4000 points plotted[3].

For stable samples ESN provides negative output at very few points; ESN has detected instability in the walk patterns which are otherwise marked stable. This indicates the fact that though there did exist minor instabilities in that particular walk but the robot eventually did not fall and completed its walk of 0.5 meters. For the fall samples, ESN does not immediately classify a data as "Fall". This indicates the fact that the robot did completed few steps successfully before the fall.

The output value does not jump directly from $+0.5$ to -0.5; it can be seen more clearly from the figure 4. This is the most important part of observation as we can notice few intermediate values indicating the unstable phase. This gives us the chance to predict a fall in advance and we have almost half a second to initiate an action to avoid or manage the fall to ensure minimum damage to the robot hardware. Observing the actual robot behaviour, it is evident from the plot that high instability occurs in the walks which lead to fall, and sometimes instability is also observed in the walks where

[3]10 points per seconds making around 400 s of walk recorded in segments of 0.5 meter each, or less if the robot fell.

the robot did not fall. Particularly for walk 2 (seconds 30 to 55) and walk 5 (seconds 115 to 140), high instability is recorded but eventually no fall occurred.

5 Conclusion

The presented paper investigated the practical applicability of *meta-sensors* through ESN implementation. The goal was to define higher level concept out of many row sensor data from within a simple training framework.

Models trained showed able to provide useful hints toward the robot status in the context of fall detections together with a low error rate, leading to classification success around 0.77 seconds. The model did scale on-line and proved accurate in detecting unstable patterns possibly leading to robot falls while walking. We were able to generate a new *meta-sensor* based on the many available on the robot so as to provide useful new information.

Future works will focus on data labelling, including intermediate labels rather than just fall/not fall. Considering the training difficulties, several trails are to be investigated, including reservoir/readout pruning and reservoir fine tuning. A complete walking task (including turning or side-walking) could also be investigated so as to ensure generality of the approach.

References

ANTONELO E., DUTOIT X., SCHRAUWEN B., STROOBANDT D., VAN BRUSSEL H. & NUTTIN M. (2007). A step towards autonomy in robotics via reservoir computing. In M. T. JAN PETERS, Ed., *NIPS 2007 Workshop: Robotics Challenges for Machine Learning*, Vancouver.

GOUAILLIER D., HUGEL V., BLAZEVIC P., KILNER C., MONCEAUX J., LAFOURCADE P., MARNIER B., SERRE J. & MAISONNIER B. (2009). Mechatronic design of nao humanoid. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, p. 2124–2129, Piscataway, NJ, USA: IEEE Press.

HANSEN N. & OSTERMEIER A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, **9**(2), 159–195.

HARTLAND C. & BREDECHE N. (2007). Using echo state networks for robot navigation behavior acquisition. In *IEEE International Conference on Robotics and Biomimetics (ROBIO07)*, p. 201–206, Sanya, China: IEEE Computer Society Press.

HARTLAND C., BREDECHE N. & SEBAG M. (2009). Memory-enhanced evolutionary robotics: The echo state network approach. *E-Commerce Technology, IEEE International Conference on*, **0**, 2788–2795.

JAEGER H. (2001). The echo state approach to analysing and training recurrent neural networks. In *Technical report GMD report 148. German National Research Center for Information Technology.*

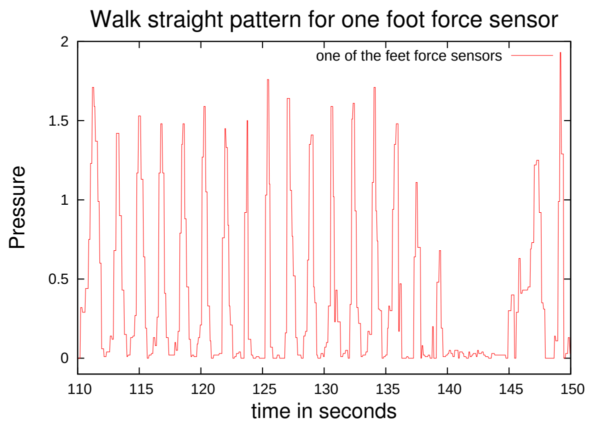
JAEGER H., MAASS W. & PRÍNCIPE J. C. (2007). Special issue on echo state networks and liquid state machines. *Neural Networks*, **20**(3), 287–289.

PLOGER P., ARGHIR A., GUNTHER T. & HOSSEINY R. (2003). Echo state networks for mobile robot modeling and control. In *In Proc. RoboCup*, p. 157–168.

6 Annexe

Training error		Test error	
Mininum	average + std	Mininum	average + std
0.21	$0.23 \pm 0.1 \times 10^{-1}$	0.18	$0.22 \pm 0.2 \times 10^{-1}$

Table 1: The table provide the best model (with the lower error rate) and the average error rate over the evolution processes for each best model found. Evaluation is performed on training data (left) and test data (right).



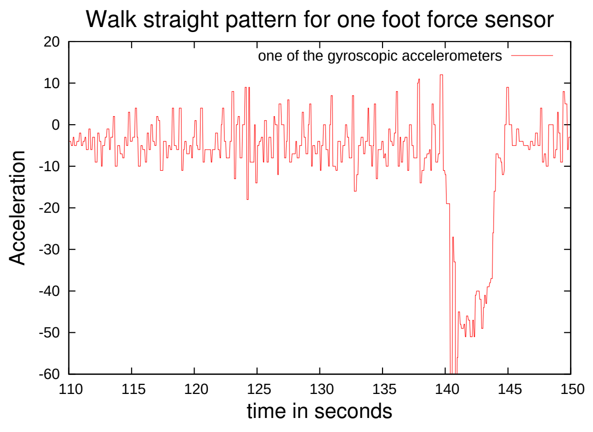


Figure 2: On top figure, we observe the value of one foot force sensor during part the walk samples during which the robot falls (sec 140). On the bottom figure, the same sample seen with one of the accelerometers on the robot.

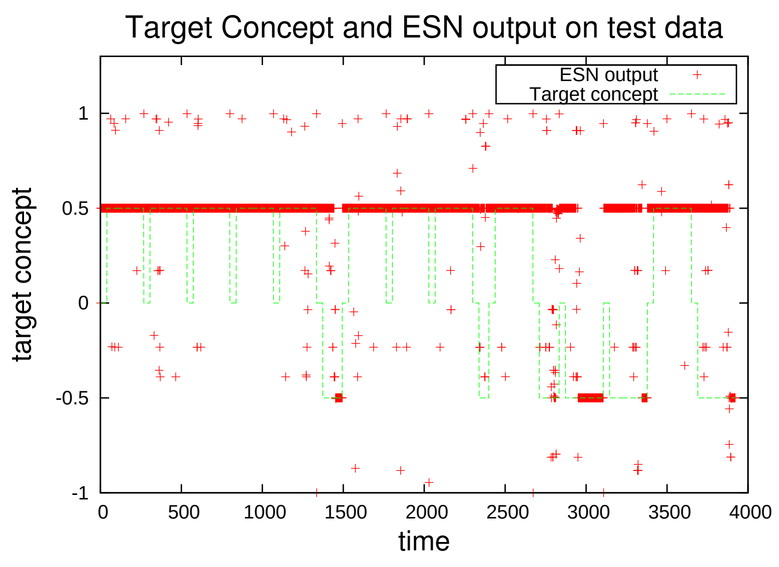


Figure 3: The desired output together with the predicted output of the trained ESN over test data.

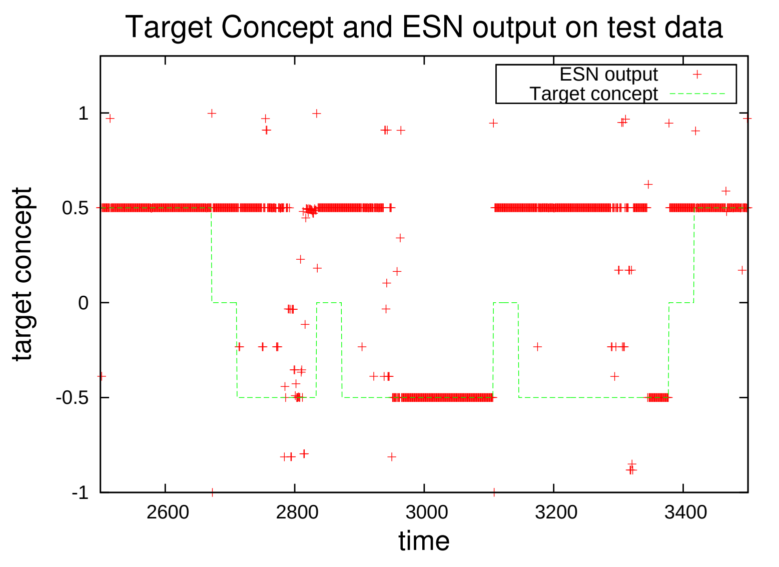


Figure 4: Zoomed in version of figure 3 on data from seconds 250 to 350, showing representative cases where the robot end-up falling.