An Experimental Evaluation of Reservoir Computation for Ambient Assisted Living

Davide Bacciu[1], Stefano Chessa[1], Claudio Gallicchio[1], Alessio Micheli[1], and Paolo Barsocchi[2]

[1] Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo, 3 - 56127 Pisa, Italy
{bacciu,ste,gallicch,micheli}@di.unipi.it
[2] ISTI-CNR, Pisa Research Area, Via Morruzzi - 56124 Pisa, Italy
paolo.barsocchi@isti.cnr.it

Abstract. In this paper we investigate the introduction of Reservoir Computing (RC) neural network models in the context of AAL (Ambient Assisted Living) and self-learning robot ecologies, with a focus on the computational constraints related to the implementation over a network of sensors. Specifically, we experimentally study the relationship between architectural parameters influencing the computational cost of the models and the performance on a task of user movements prediction from sensors signal streams. The RC shows favorable scaling properties results for the analyzed AAL task.

Keywords: Reservoir Computing, Echo State Networks, Wireless Sensor Networks, Ambient Assisted Living.

1 Introduction

The aim of Ambient Assisted Living (AAL) [8] applications is to integrate different technologies to improve the quality of life of elders and disable people, by assisting them in the environments where they live and work. Recently, the EU FP7 RUBICON [1] (Robotic UBiquitous COgnitive Network) project [1,3], has proposed the use of self-adaptive robotic ecologies to approach tasks in the field of AAL. A robotic ecology consists in a network of heterogeneous devices including mobile robots, wireless sensor networks (WSNs) [5] and actuators. The objective of RUBICON is to integrate self-sustaining learning solutions to provide cheap, adaptive and efficient coordination of the robotic ecology. RUBICON is based on the interplay among different layers implementing state-of-the-art techniques in machine learning, WSNs, cognitive robotics and agent control systems. Learning in a network of distributed devices with very limited computational capabilities (such are WSNs), raises novel challenges related to the effectiveness and the efficiency of the learning models used to handle the temporal data sensed by

[1] http://www.fp7rubicon.eu/

B. Apolloni et al. (Eds.): *Neural Nets and Surroundings*, SIST 19, pp. 41–50.
DOI: 10.1007/978-3-642-35467-0_5

the ecology. Reservoir Computing (RC) [14], and in particular the Echo State Network (ESN) [12,11] model, represent an emerging paradigm for modeling Recurrent Neural Networks (RNNs), and offer in principle an interesting trade-off between computational efficiency and the ability of learning in domains of temporal sequences. RC is therefore identified as a potentially suitable approach for the implementation of learning models on-board the nodes of robotic ecologies, although the parameters of the RC networks should be tailored to this specific applicative context. Moreover, in [10,4] it was already experimentally shown that ESNs are particularly suitable for treating information gathered by WSNs for tasks related to the prediction of indoor user movements. The problem of reducing the computational requirements of RC implementations (e.g. [15,9]), in particular for their embedding into WSN nodes [6] is currently a topic of active research.

In this paper we present an experimental study of the relation between the performance and the implementation cost of an RC system in a real-world AAL task consisting in predicting user movements in indoor environments. Such study extends the work in [10,4] to consider the effect of different components of the RC architecture such as the number of reservoir units and the weight encoding. The proposed investigation is useful to understand the potentiality of the RC approach for practical implementations on the nodes of robotic ecologies.

2 Learning in RUBICON Robotic Ecology

In this Section, we characterize the RC based Learning Layer in the RUBICON ecology system. By embedding learning functionalities on-board the nodes of the ecology, the role of the Learning Layer in RUBICON is to supply the general purpose learning infrastructure necessary for achieving self-sustaining learning capabilities.

The purpose of the Learning Layer (LL) is to respond to the need of adaptivity and analysis of temporal context (including the sensor data dynamics) of the RUBICON environments, by providing learning mechanisms that are used by higher (control and cognitive) layers of RUBICON architecture. Specifically, its role is to:

- recognize and detect relevant sensed information by providing predictions which depend on the temporal history of the input signals;
- analyze and process sensed information to extract refined goal-significant information (e.g. information fusion, event recognition).

To this purpose, a networked learning infrastructure, named Learning Network (LN), is built on top of the robotic ecology in order to provide the core learning services to the higher levels of the RUBICON architecture. The LN is a *flexible environmental memory* that serves as a task driven model of the environment that can readily be shared by new nodes connecting to RUBICON. This allows a straightforward sharing of the learned experience.

In the LN design, each node hosts a RC network composed by a number of artificial neurons. Neurons are connected by remote synapses (implemented by

means of communication channels) with neurons residing on other nodes, thus creating a distributed, artificial recurrent neural network. The same synapses are also used to interconnect the output of the LN with the control and cognitive layers of RUBICON. With this approach, each neuron in the local learning model may be considered as a node of a distributed reservoir; the instantaneous state of such reservoir thus incorporates the combined knowledge attained by each sensor node, while reflecting the dynamics of the overall RUBICON. The design and development of the LN is strongly influenced by the issues of scalability and efficiency, especially with regards to the limitations of the nodes (most of which are wireless sensors) in terms of computation, communication and energy constraints. In particular, the design of the LN makes use of RNNs, and specifically of RC models, due to their modular, networked structure which naturally adapts to the distributed nature of the RUBICON ecology, while their recurrent nature allows to effectively capture the dynamics of the system processes. At the same time, the neural paradigm offers a natural approach to robustly cope with noisy sensed data.

Preliminary studies of RC used in combination with sensors for localization applications aimed at forecasting users movements can be found in [10] and [4]. Differently from previous works, in the following we extend the study to consider different components of the RC architecture.

3 Echo State Network Architecture

An ESN [12,11] is made up of an input layer, a reservoir and a readout with N_U, N_R and N_Y units, respectively (see Fig. 1). The reservoir is a large, sparsely connected, non linear, *untrained* recurrent hidden layer, used to compute an input-driven *fixed* contractive encoding of the input sequences into a state space. The readout is a linear, feed-forward output tool, responsible for the output computation and representing the only trained component of the ESN architecture. In standard ESNs, sigmoid reservoir units are used, e.g. implementing *tanh* activation function. In this paper, we take into consideration leaky integrator ESNs (LI-ESNs) [13], in which leaky integrator reservoir units are used, applying an exponential moving average to the reservoir state values. The use of leaky

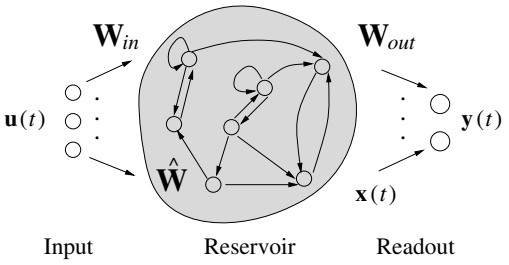

Fig. 1. The ESN architecture

integration of reservoir states in LI-ESNs, implies a better handling of input sequences changing slowly with respect to the sampling frequency [13,14,2], and results in an RC model particularly suitable for treating real-world RSS input signals, as experimentally shown in [10].

Given an input sequence $\mathbf{s} = [\mathbf{u}(1), \ldots, \mathbf{u}(n)]$ over the real input space $\mathbb{R}^{N_U}$, for each time step t the reservoir of a LI-ESN computes the state transition function: $\mathbf{x}(t) = (1-a)\mathbf{x}(t-1) + af(\mathbf{W}_{in}\mathbf{u}(t) + \hat{\mathbf{W}}\mathbf{x}(t-1))$, where $\mathbf{x}(t) \in \mathbb{R}^{N_R}$ is the state of the network at time t, $\mathbf{W}_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix (possibly including also a bias term), $\hat{\mathbf{W}} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir weight matrix, f is a component-wise applied activation function (we use $f \equiv tanh$) and $a \in [0, 1]$ is a *leaking rate* parameter, controlling the speed of reservoir dynamics (smaller values of a imply slower reservoir dynamics). Note that for $a = 1$, LI-ESN state transition function reduces to the case of standard ESN. The output is computed by the readout tool, by linearly combining the activation of the reservoir units. In the following, we restrict our consideration to the case of binary classification tasks on sequences, which is of a particular interest for this paper. In this case, the readout is applied only when the input sequence has been completely seen and the reservoir encoding process has terminated. For each input sequence $\mathbf{s}$ of length n, the readout computes $y(\mathbf{s}) = sgn(\mathbf{W}_{out}\mathbf{x}(n))$, where $\mathbf{W}_{out} \in \mathbb{R}^{N_Y \times N_R}$ is the reservoir-to-readout weight matrix (possibly including also a bias term), $y(\mathbf{s}) = \{-1, +1\}$ is the output classification computed for the input sequence $\mathbf{s}$, and sgn is a sign threshold function.

The reservoir of a LI-ESN is initialized in order to satisfy the *Echo State Property* (ESP) [11,12], i.e. the network state should asymptotically depend only on the driving input sequence [9]. A sufficient and a necessary condition for the ESP are provided in literature [11]. A necessary condition for the ESP is typically considered in RC applications [13,14,11], i.e. $\rho(\tilde{\mathbf{W}}) < 1$, where $\tilde{\mathbf{W}} = (1-a)\mathbf{I} + a\hat{\mathbf{W}}$, and $\rho(\tilde{\mathbf{W}})$ is the *spectral radius* of $\tilde{\mathbf{W}}$. Accordingly, $\hat{\mathbf{W}}$ is randomly initialized and then rescaled to meet the condition on ρ, where values of ρ close to 1 are generally used [14,12,16]. Weight values in $\mathbf{W}_{in}$ are chosen from a random (uniform) distribution over $[-scale_{in}, scale_{in}]$, where $scale_{in}$ is an *input scaling* parameter.

The readout of a LI-ESN is typically trained off-line by using Moore-Penrose pseudo-inversion or ridge regression (see e.g. [10,14]).

Beside the choice of the spectral radius, one of the most relevant parameters in RC applications is the number of reservoir units. Indeed, larger reservoirs often lead to better network performances (e.g. [16,9]). Notice that the cost of ESN application (both in time and space) increases with the reservoir dimension[2]. Another interesting aspect is related to the weight dimension, i.e. the number of bits used to represent each weight in $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$. Typically, such weights are randomly initialized to assume values in an infinite real interval, and in practice floating-points with double precision are used, requiring 64 bits per

[2] The cost is quadratically bounded in general and, due to the sparsity of reservoir connectivity, it can be linear under the assumption of a fixed maximum number of connections for each reservoir unit.

weight memorization. In the following, this setting is referred to as LI-ESN *full weight encoding*. In this paper we also consider the alternative strategy of assuming a small finite set of possible non-zero weights in the untrained RC matrices $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$. The number of different non-zero values is denoted by N_w. In this case, referred in the following as LI-ESN with *reduced weight encoding*, each weight value can still be a double precision floating-point, but a small number of bits (i.e. $\lceil log_2 N_w \rceil$) are sufficient to encode each weight in the network matrices.

The issue of minimizing the cost for the network memorization is of a fundamental relevance in view of the embedding of the RC networks directly on the motes. Indeed, it is worth to note that the motes of a WSN usually host a very limited *total* amount of RAM memory (usually in the order of 8-10 Kbytes). We therefore experimentally assess the impact on the predictive performance of RC networks due to the variation of the number of reservoir units and the type of weight encoding used (i.e. full or reduced), specifically in a task of user movement prediction type.

4 Experiments

Real WSN data for our experiments was collected during a measurement campaign at the first floor of the the ISTI institute of CNR in the Pisa Research Area, in Italy. The environment comprises 4 rooms organized into pairs of coupled rooms (denoted as Room 1 and Room 2 in each couple), with front doors separated by a hallway. Rooms contained typical office furniture including chairs, cabinets, monitors and desks (see Fig. 2), representing harsh conditions for indoor wireless communications. In each couple of rooms, we set up a WSN composed of 5 IRIS motes [7]: 4 fixed sensors, or *anchors*, and 1 *mobile* sensor worn by the user. Sensors embedded a Chipcon AT86RF230 radio subsystem implementing the IEEE 802.15.4 standard. The user moved in the environment according to the 6 possible paths illustrated in Fig. 2. Following a curved trajectory the user remained in the same room, while straight trajectories led to a room change. Each measurement gathered the received signal strength (RSS) information between the anchors and the mobile at a constant frequency of 8 Hz, from the starting point of each movement until the user reached a *marker* point (denoted as M in Fig. 2) located at 60 cm from the door. The RSS data was used to set up a dataset for a binary classification task in which RSS sequences from the different anchors are used as input data and the target consists in $+1$ if the corresponding user movement led to a room change, and -1 otherwise. In practice, the classification tasks consists in predicting if the user is about to change room or not based on the history of the RSS information until the marker. Note that the marker M is the same for all the movements, hence it not possible to distinguish the different paths based only on the RSS values collected at M. The dataset contains 210 input sequences, where 104 of them where collected in the first couple of rooms, and the remaining 106 in the second couple. Using a stratified sampling over the different movement paths, the dataset was divided into a training and a test set comprising 168 and 42 sequences, respectively.

We run experiments for all the 15 possible configurations of the number of anchors considered, i.e. for $N_{anchors} = 1, 2, 3, 4$. We used reservoirs with $N_R \in \{10, 20, 50, 100, 300, 500\}$ units, 10% of connectivity, input scaling $scale_{in} = 1$ and spectral radius $\rho = 0.99$. In the reduced weight encoding setting, each non-zero weight value in $\mathbf{W}_{in}$ and $\hat{\mathbf{W}}$ was randomly chosen in a small weight alphabet with $N_w = 8$ possible values, uniformly sampled in the interval $[-0.4, 0.4]$, thus leading to just a 3 bit encoding per weight memorization. For every choice of the reservoir hyper-parametrization, we considered 10 independent (random guessed) reservoirs (and the results were averaged over the 10 guesses). The readout was trained using pseudo-inversion and ridge-regression, with regularization parameter $\lambda_r \in \{10^{-1}, 10^{-3}, 10^{-5}\}$. The readout regularization was chosen by model selection with stratified holdout validation, using $\approx 30\%$ of the training set as validation set.

Fig. 3 shows the averaged test accuracy of LI-ESN with full weight encoding, for increasing reservoir dimension N_R and varying the number of anchors used $N_{anchors}$. For any value of $N_{anchors}$ considered, results in Fig. 3 are averaged (and standard deviations are computed) over the number of possible configurations of the anchors. It can be noticed that the RC networks can reach excellent test performances, and the test accuracy scales very well with both the reservoir dimension and the number of anchors. As already pointed out in [10,4], the test accuracy of the LI-ESN networks is improved when more anchors are progressively considered in the WSN setting. The best performance (test accuracy up to 97.1%) is obtained for $N_{anchors} = 4$, although very good performances are achieved for simpler WSN settings with a fewer number of anchors (up to 88.6%, 91.5% and 95.4%, for 1, 2 and 3 anchors, respectively). Table 1 details the training and test accuracies (and standard deviations over the reservoir guesses) in the case $N_{anchors} = 4$, for increasing reservoir dimensionality. The best test accuracy, i.e. 97.1%, is obtained for $N_R = 500$, corresponding to test sensitivity and specificity of 99.5% and 95.0%, respectively. Moreover, from Fig. 3 it is clear that for every choice of $N_{anchors}$, the LI-ESN performance is improved as larger reservoirs are considered (adopting regularization in readout training in order

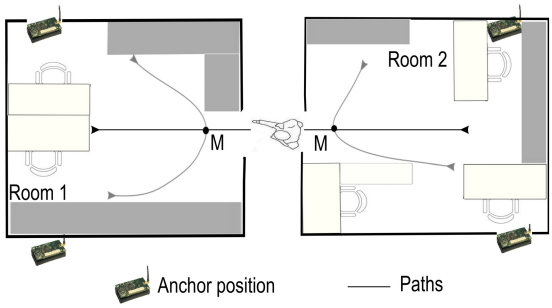

Fig. 2. Prototypical environment where the RSS measurements have been collected, showing the positions of the anchors of the WSN in one couple of rooms and the possible user movement paths

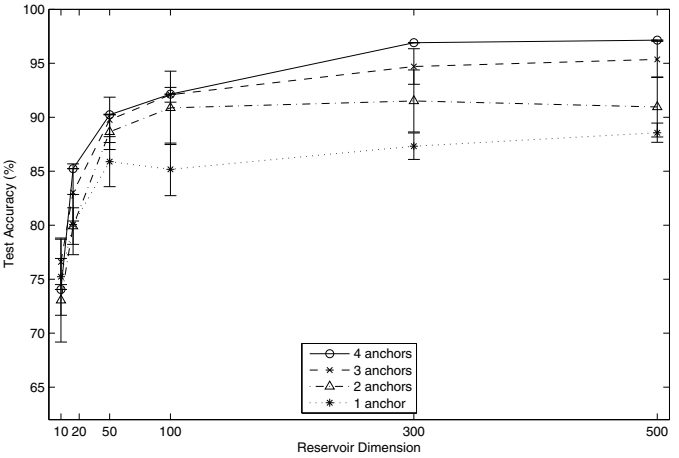

Fig. 3. Mean test accuracy (in %) and standard deviation of LI-ESNs with *full weight encoding*, varying the reservoir dimension and the number of anchors considered

Table 1. Mean accuracy (in %) and standard deviation of LI-ESNs with *full weight encoding* for $N_{anchors} = 4$

	Reservoir Dimension (N_R)					
	10	20	50	100	300	500
Training	76.1(±3.1)	86.7(±2.5)	96.4(±2.1)	99.8(±0.3)	99.0(±0.5)	99.9(±0.2)
Test	74.0(±5.6)	85.2(±4.6)	90.2(±3.9)	92.1(±3.2)	96.9(±1.5)	97.1(±2.3)

to avoid overfitting). At the same time, it can also be observed that the test performance tends to be saturated as the number of reservoir units is increased. Indeed, when a sufficiently large reservoir is considered, a further increase of the reservoir dimension can only slightly improve the test accuracy. This means that small enough reservoirs can be sufficient in practice and thus a small amount of memory can be sufficient for storing the network parameters on the motes of the WSN. For instance (see Table 1), for $N_{anchors} = 4$, the test performance with $N_R = 500$ units is 97.1%, while with smaller reservoirs e.g. with $N_R = 100$ and $N_R = 50$, the test performance is 92.1% and 90.2%, respectively, which are still very satisfactory results for this kind of tasks (from noisy input data).

The averaged test accuracy for LI-ESNs with reduced weight encoding is shown in Fig. 4, varying the reservoir dimensionality and the number of anchors. Analogous considerations can be done as for the case of full weight encoding. RC networks achieve very good test accuracy, which scales well with the number of anchors in the environment and the reservoir dimension. Also in the case of reduced weight encoding, a saturating effect of the networks performance can be observed for increasing the number of reservoir units. Table 2 reports the training

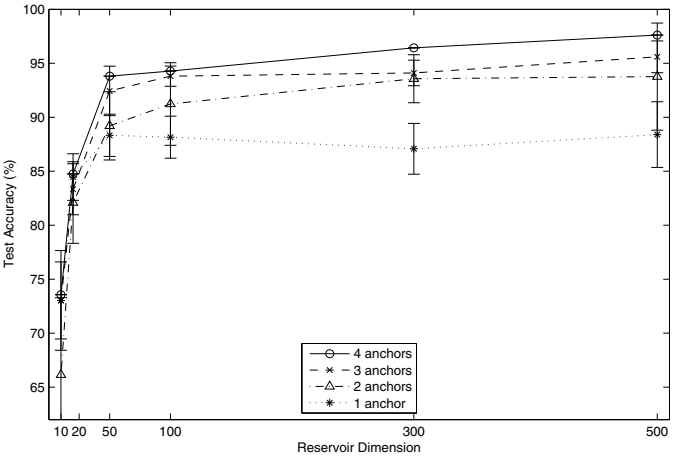

Fig. 4. Mean test accuracy (in %) and standard deviation of LI-ESNs with *reduced weight encoding*, varying the reservoir dimension and the number of anchors considered

and test accuracies achieved by LI-ESNs in correspondence of $N_{anchors} = 4$ and varying the number of reservoir units. The best test performance is obtained for $N_R = 500$, corresponding to a test accuracy of 97.6%, test sensitivity of 99.0% and test specificity of 96.4%. Very interestingly, comparing the performances obtained by LI-ESNs with full and reduced weight encodings (see Figures 3 and 4, and Tables 1 and 2), we can observe that in the two cases, the accuracies are substantially the same (except for small, statistical fluctuations) for correspondent experimental settings. As a result, the use of a reduced weight encoding scheme does not have a relevant practical effect on the LI-ESN model performances, and in particular it does not strictly depend on the reservoir dimension, due to the prevailing role of the number of reservoir units on the accuracy results[3]. This means that in practice, using a small finite set of reservoir weights, thus a few bits per weight encoding, is sufficient for tackling the task at hand, and the small amount of memory typically available on the motes of a WSN can be enough for ESN embedding. To give an idea of the possible reduction in memory requirements, for storing the weights in $\hat{\mathbf{W}}$ of a 100-dimensional reservoir, roughly 8 Kbytes of memory are needed in case of full weight encoding, while roughly 800 bytes are sufficient in case of reduced weight encoding. For 50-dimensional reservoirs, the memory requirement reduces from roughly 2 Kbytes (for full weight encoding) to roughly 250 bytes (for reduced weight encoding).

[3] Note that the number of bits used in our experiments for the reduced weight encoding scheme, i.e. $log_2(N_w) = 3$ bits, is even smaller than what could be necessary for embedded implementations of our RC system on-board the motes of a WSN. Considering a further reduction of the number of bits used for the weight values encoding could lead to investigations of architectural variants of the ESN model, e.g. see [15].

Table 2. Mean accuracy (in %) and standard deviation of LI-ESNs with *reduced weight encoding* for $N_{anchors} = 4$

	Reservoir Dimension (N_R)					
	10	20	50	100	300	500
Training	76.8(±2.4)	85.5(±3.9)	96.6(±1.4)	100.0(±0.0)	100.0(±0.0)	100.0(±0.0)
Test	73.6(±4.3)	84.8(±3.9)	93.8(±3.4)	94.3(±2.7)	96.4(±2.7)	97.6(±1.8)

Concerning the computational power constraints, we observed that on motes with an 8 MHz processor, like the IRIS motes used in our experiments, the LI-ESN computations required for each input-output step approximatively 0.7 and 2 milliseconds, respectively for reservoirs of dimension 50 and 100, which is therefore plausible for a real-time processing on-board the nodes of a WSN.

5 Conclusions

We have presented an experimental investigation of the relation between the performance and the implementation cost of RC networks in an AAL task consisting in anticipating user movements in indoor environments. Results have shown that the proposed RC system achieves a very good predictive performance on the considered task. Such performance scales well with the number of anchors used and with the cost of the network memorization. In particular, a decrease in the number of reservoir units does not lead to a dramatic degeneration of the performance (up to 50 units), and the use of a reduced encoding scheme (requiring only 3 bits of memory for each weight value) does not affect significantly the accuracy of the model. A small reservoir with each weight value encoded in a few bits is sufficient in practice for the analyzed task. Such promising results allow us to envisage practical solutions for the embedding of the RC networks on-board the motes of WSNs.

The experimental analysis presented in this paper represents a ground for further investigations on the development of local and distributed learning capabilities based on RC systems distributed across the nodes of the RUBICON ecology.

Acknowledgement. This work is partially supported by the EU FP7 RUBICON project (contract n. 269914).

References

1. Amato, G., Broxvall, M., Chessa, S., Dragone, M., Gennaro, C., Lopez, R., Maguire, L., Mcginnity, M., Micheli, A., Renteria, A., O'Hare, G., Pecora, F.: Robotic UBIquitous COgnitive Network. In: Proceedings of the 3rd International Symposium on Ambient Intelligence (ISAMI), Salamanca, March 28-30 (To Appear, 2012)

2. Antonelo, E.A., Schrauwen, B., Stroobandt, D.: Event detection and localization for small mobile robots using reservoir computing. Neural Networks 21(6), 862–871 (2008)
3. Bacciu, D., Broxvall, M., Coleman, S., Dragone, M., Gallicchio, C., Gennaro, C., Guzman, R., Lopez, R., Lozano-Peiteado, H., Ray, A., Renteria, A., Saffiotti, A., Vairo, C.: Self-Sustaining Learning for Robotic Ecologies. In: Proceedings of the 1st International Conference on Sensor Networks (SensorNets), Rome, February 24- 26 (To Appear, 2012)
4. Bacciu, D., Gallicchio, C., Micheli, A., Chessa, S., Barsocchi, P.: Predicting user movements in heterogeneous indoor environments by reservoir computing. In: Bhatt, M., Guesgen, H.W., Augusto, J.C. (eds.) Proceedings of the IJCAI Workshop on Space, Time and Ambient Intelligence STAMI 2011, pp. 1–6 (2011)
5. Baronti, P., Pillai, P., Chook, V.W.C., Chessa, S., Gotta, A., Hu, Y.F.: Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. Computer Communications 30(7), 1655–1695 (2007)
6. Chang, M., Terzis, A., Bonnet, P.: Mote-Based Online Anomaly Detection Using Echo State Networks. In: Krishnamachari, B., Suri, S., Heinzelman, W., Mitra, U. (eds.) DCOSS 2009. LNCS, vol. 5516, pp. 72–86. Springer, Heidelberg (2009)
7. Crossbow Technology Inc.: `http://www.xbow.com`
8. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, L., Burgelman, J.: Scenarios for Ambient Intelligence in 2010. Tech. rep., IST Advisory Group (February 2001)
9. Gallicchio, C., Micheli, A.: Architectural and markovian factors of echo state networks. Neural Networks 24(5), 440–456 (2011)
10. Gallicchio, C., Micheli, A., Barsocchi, P., Chessa, S.: User movements forecasting by reservoir computing using signal streams produced by mote-class sensors. In: Del Ser, J., Jorswieck, E.A., Miguez, J., Matinmikko, M., Palomar, D.P., Salcedo-Sanz, S., Gil-Lopez, S. (eds.) Mobilight 2011. LNICST, vol. 81, pp. 151–168. Springer, Heidelberg (2012)
11. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Tech. rep., GMD - German National Research Institute for Computer Science (2001)
12. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304(5667), 78–80 (2004)
13. Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks 20(3), 335–352 (2007)
14. Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review 3(3), 127–149 (2009)
15. Rodan, A., Tiňo, P.: Minimum complexity echo state network. IEEE Transactions on Neural Networks 22(1), 131–144 (2011)
16. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks 20(3), 391–403 (2007)