Towards Autonomous Self-localization of Small Mobile Robots using Reservoir Computing and Slow Feature Analysis

Eric Antonelo, Benjamin Schrauwen
Department of Electronics and Information Systems
Ghent University
Ghent, Belgium
eric.antonelo@elis.ugent.be

***Abstract*—Biological systems such as rats have special brain structures which process spatial information from the environment. They have efficient and robust localization abilities provided by special neurons in the hippocampus, namely place cells. This work proposes a biologically plausible architecture which is based on three recently developed techniques: Reservoir Computing (RC), Slow Feature Analysis (SFA), and Independent Component Analysis (ICA). The bottom layer of our RC-SFA architecture is a reservoir of recurrent nodes which process the information from the robot's distance sensors. It provides a temporal kernel of rich dynamics which is used by the upper two layers (SFA and ICA) to autonomously learn place cells. Experiments with an e-puck robot with 8 infra-red sensors (which measure distances in [4-30] cm) show that the learning system based on RC-SFA provides a self-organized formation of place cells that can either distinguish between two rooms or to detect the corridor connecting them.**

***Index Terms*—reservoir computing, slow feature analysis, place cells, robot localization**

I. Introduction

Studies carried on rodents show that these animals can efficiently process spatial information from their environment [1]. They are able to get acquainted with a new environment and through learning its features they can construct a spatial representation of their environment (i.e., a cognitive map). The part of the rat's brain which is involved in this spatial representation is the hippocampus, and more specifically, areas CA1 and CA2 [1].

Intelligent autonomous navigation systems can be classified in reactive and deliberative systems. Whereas reactive systems consists of instantaneous sensory-motor mappings, deliberative systems have more abstract planning capabilities and they usually take into account the history of sensory inputs (or past events) for taking the next action. The ability of a robot to self-localize in its environment is clearly important once it provides basic cognitive information to a deliberative navigation system. In other words, it allows for planning and higher level cognitive behavior in the context of mobile robots as well as biological systems.

Traditional robot localization systems are designed mostly by probabilistic methods which can perform SLAM (Simultaneous Localization And Mapping) under suitable assumptions [2] and are usually built for robots having high-resolution expensive laser scanners. Biologically inspired systems for robot localization can be considered a competitive alternative that works well for small mobile robots. Robustness, learning and low computation time are some characteristics of these biological inspired systems. Most systems are based on visual input from a camera [3]–[6] and model hippocampal place cells from rats [3]–[7].

These place cells are the main components of the spatial navigation system in rodents. Each place cell codes for a particular location of the rat's environment, presenting a peak response in the proximities of that location (the place field of that cell). Other components of the brain's spatial representation system includes head-direction cells, which encode the orientation of the animal in its environment, and grid cells, which are non-localized representations of space (having a grid-like structure of activations in space) [1].

There are two classes of stimuli used by place and grid cells, namely, idiothetic and allothetic. The former corresponds to inner system signals, i.e., proprioceptive sensors such as odometry signals (e.g., from encoders), inertia sensors, etc. The latter (allothetic input) is represented by external information coming from the environment such as distance sensors, camera, touch sensors, etc. Most systems use allothetic information to correct the odometry signals from dead reckoning [3], [5]–[7].

Previous work has focused on supervised learning approaches for robot localization using solely short-range noisy distance sensors [8]. In this work, we propose a hierarchical architecture with an unsupervised learning technique which is built on the concept of slowness [9]. Our architecture is composed of three layers, where the first layer is a randomly created recurrent neural network (the reservoir) which functions as a temporal kernel that projects the input to a high dimensional dynamic space. The second layer receives the signals from the reservoir and the input, and extracts the slow features from it using Slow Feature Analysis (SFA) [9]. The SFA layer learns to encode spatial representations of the environment which are invariant over time. The third layer performs sparse coding on the output of the SFA layer using Independent Component Analysis (ICA) [10], generating

localized representations of space (as the place cells from biological systems).

Similarly to [4], we use SFA to model grid cells. While they use several SFA layers and high-dimensional input from a camera, we use only few noisy distance sensors and a RC-SFA based architecture. As the reservoir weights remain fixed in our RC-SFA architecture, only the upper layers learn. The reservoir concept is first used in the form of Echo State Networks [11] and Liquid State Machines [12]. These two learning paradigms for recurrent neural networks have been recently termed under the common name of Reservoir Computing [13]. Both techniques do not train the recurrent neural network, but only a linear readout output layer (supervised learning).

As we will show in this paper, a small mobile robot (the e-puck) can learn to self-localize in real environments based solely on short-range noisy distance sensors. The main advantages of this work are five-fold: the environment is unstructured; the robot has small dimensions and only 8 short-range distance sensors (4 cm - 30 cm); odometry information is not used; self-localization emerges from an unsupervised learning process; and the model can correct itself from kidnapping situations without any pre-design decision (the kidnapping is not seen during learning).

The proposed architecture is biologically plausible and models the place cells found in rodents. Previous work has shown the modeling of place cells with the proposed architecture only for simulated mobile robots [8]. Now we extend it to real environment settings using the e-puck robot. In addition, we show the importance of the timescale of the reservoir for learning of place cells.

II. Methods

A. Reservoir Computing

Reservoir Computing (RC) is used here to model the first layer of our architecture. The basic building block of RC is the reservoir, which is a randomly created recurrent neural network. This network is composed of sigmoidal neurons and is modeled by the following state update equation [11]:

$$\mathbf{x}(t+1) = f((1-\alpha)\mathbf{x}(t) + \alpha(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}_{\text{res}}\mathbf{x}(t))), \quad (1)$$

where: $\mathbf{u}(t)$ denotes the input at time t; $\mathbf{x}(t)$ represents the reservoir state; α is the leak rate [14], [15]; and $f() = \tanh()$ is the hyperbolic tangent activation function (common type of activation function used in reservoirs). The connections between the nodes of the network are represented by weight matrices: $\mathbf{W}_{\text{in}}$ is the connection matrix from input to reservoir and $\mathbf{W}_{\text{res}}$ represents the recurrent connections between internal nodes. The initial state of the dynamical system is $\mathbf{x}(0) = \mathbf{0}$. A standard reservoir (without the leak rate) is found when $\alpha = 1$.

The connection matrices $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{res}}$ are randomly generated and remain always fixed (non-trainable weights). The recurrent connections $\mathbf{W}_{\text{res}}$ are generated from $N(0,1)$ and rescaled such that the system is stable and the reservoir has the echo state property (i.e., it has a fading memory [11]). This can be done by rescaling the matrix so that the spectral radius $|\lambda_{max}|$ (the largest absolute eigenvalue) of the linearized system is smaller than one [11]. In this work, reservoir weights ($\mathbf{W}_{\text{res}}$) are rescaled to achieve a spectral radius of $|\lambda_{max}| = 0.99$ which is an arbitrarily chosen value which sets the reservoir to the edge of stability. The initialization of $\mathbf{W}_{\text{in}}$ is given in Section III-A.

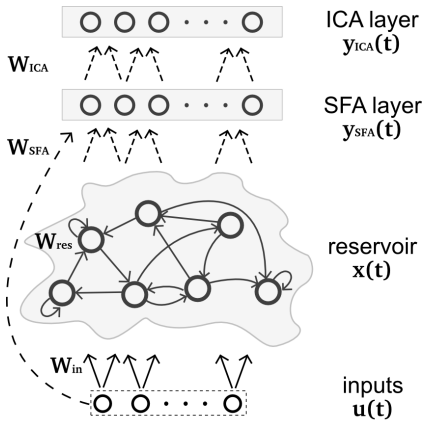

Fig. 1. RC-SFA architecture.

The reservoir dynamics can also be tuned to match the input signal dynamics by changing the leak rate of the reservoir $\alpha \in (0, 1]$ [14], [15]. So, low leak rates make the reservoir function in a slow timescale, effectively increasing its memory capacity but reducing its ability to respond quickly to input signals. On the other hand, leak rates close to 1 yield reservoirs with less memory to hold past stimuli but with more agile processing of the input.

RC-based systems are usually trained in a supervised way. In these systems, the reservoir states are mapped to the desired output with a readout matrix which is usually found by standard linear regression methods on the reservoir states [13]. In this paper, we propose a RC-SFA architecture which is a hierarchical network of nodes where the lower layer is the reservoir and the upper layers are composed of SFA and ICA units, respectively (Fig. 1). The function of the reservoir is basically to expand the input signals to a high-dimensional dynamic space. The randomly created reservoir can be understood as a temporal non-linear kernel which extract features from the input signal. Because of its recurrent connections, the reservoir states contain *echoes* of the past inputs, providing a short-term memory to our model. The SFA layer receives signals from the input nodes $\mathbf{u}(t)$ and from the reservoir nodes $\mathbf{x}(t)$. This layer learns instantaneous functions of the input which are slowing-varying or invariant representations [9] of the reservoir states. The ICA layer learns a sparse and local representation of the SFA features and is the last layer in our architecture. The following sections focus on these upper layers.

Next, consider the following notation:

n_{u} : number of inputs;
n_{res}: number of neurons in the reservoir;
n_{sfa}: number of SFA units;
n_{ica}: number of ICA units.

B. Slow Feature Analysis

Slow Feature Analysis (SFA) is an unsupervised learning method which finds instantaneous functions of the input based on the concept of slowness [9]. It is able to extract slowing-varying features of an input signal. In [16], it is shown that SFA is able to reproduce qualitative and quantitative properties of complex cells from the primary visual cortex V1. In [4], grid cells (from the entorhinal cortex of rats) and hippocampal place cells are modeled with an hierarchy of non-linear SFA layers.

The learning problem can be defined as follows. Given a high-dimensional input signal $\mathbf{x}(t)$, find a set of scalar functions $g_i(\mathbf{x}(t))$ so that the SFA output $y_i = g_i(\mathbf{x}(t))$ varies as slowly as possible and still carries significant information. Mathematically, find output signals $y_i = g_i(\mathbf{x}(t))$ such that [9]:

$$\Delta(y_i) := \langle \dot{y}_i^2 \rangle_t \qquad \text{is minimal} \tag{2}$$

under the constraints

$$\langle y_i \rangle_t = 0 \qquad \text{(zero mean)} \tag{3}$$

$$\langle y_i^2 \rangle_t = 1 \qquad \text{(unit variance)} \tag{4}$$

$$\forall j < i, \langle y_i y_j \rangle_t = 0 \qquad \text{(decorrelation and order)} \tag{5}$$

where $\langle . \rangle_t$ and $\dot{y}$ denote temporal averaging and the derivative of y, respectively.

Algorithm: As a pre-processing step, the input signal $\mathbf{x}(t)$ is normalized to have zero mean and unit variance. We consider the linear case $g_i(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, because the input signal is already non-linearly expanded by the reservoir in the first layer. The SFA learning algorithm is as follows:
Solve the generalized eigenvalue problem:

$$\mathbf{AW} = \mathbf{BW}\Lambda, \tag{6}$$

where $\mathbf{A} := \langle \dot{\mathbf{x}}\dot{\mathbf{x}}^T \rangle_t$ and $\mathbf{B} := \langle \mathbf{x}\mathbf{x}^T \rangle_t$.
The eigenvectors $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{n_{sfa}}$ corresponding to the ordered generalized eigenvalues $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{n_{sfa}}$ solve the learning task, satisfying (3-5) and minimizing (2) (see [9] for more details). This algorithm is guaranteed to find the global optimum.

Architecture: The SFA layer in our architecture (Fig. 1) is denoted by $\mathbf{y}_{\mathrm{sfa}}(t)$:

$$\mathbf{y}_{\mathrm{sfa}}(t) = \mathbf{W}_{\mathrm{sfa}}\mathbf{x}_{\mathrm{sfa}}(t), \tag{7}$$

where: $\mathbf{x}_{\mathrm{sfa}}(t)$ is the input vector at time t consisting of a concatenation of input $\mathbf{u}(t)$ and reservoir states $\mathbf{x}(t)$. Note that the states $\mathbf{x}(t)$ are generated by feeding the input $\mathbf{u}(t)$ for $t = 1, 2, ...n_{\mathrm{s}}$ in equation (1), where n_{s} is the number of samples. The matrix $\mathbf{W}_{\mathrm{sfa}}$ is a $n_{\mathrm{sfa}} \times (n_{\mathrm{u}} + n_{\mathrm{res}})$ matrix found by solving (6). After learning, $\mathbf{y}_{\mathrm{sfa}}(t)$ generates non-localized representations of the environment, in a similar way as grid cells of the entorhinal cortex of rats [1].

C. Independent Component Analysis

Independent Component Analysis (ICA) is a method for sparse coding of an input signal as well as blind source separation [10]. The learning problem of ICA [10] can be defined as follows. Assume that a linear mixture of signals $x_1, x_2...x_n$ can be used for finding the n independent components or latent variables $s_1, s_2...s_n$. The observed values $\mathbf{x}(t) = [x_1(t), x_2(t)...x_n(t)]$ can be written as:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) \tag{8}$$

where $\mathbf{A}$ is the mixing matrix; and $\mathbf{s}(t) = [s_1(t), s_2(t)...s_n(t)]$ is the vector of independent components (both $\mathbf{A}$ and $\mathbf{s}(t)$ are assumed to be unknown). The vector $\mathbf{s}(t)$ can be generated after estimating matrix $\mathbf{A}$:

$$\mathbf{s}(t) = \mathbf{W}\mathbf{x}(t) \tag{9}$$

where $\mathbf{W}$ is the inverse matrix of $\mathbf{A}$. The basic assumption for ICA is that the components s_i are statistically independent and have nongaussian distributions [10].

Algorithm: We use the FastICA algorithm for finding $\mathbf{W}$ [10]. The observed vector $\mathbf{x}(t)$ is preprocessed by centering (zero-mean) and whitening (decorrelation and unit variance) [10]. The algorithm tries to maximize the nongaussianity of ICA neurons $\mathbf{w}\mathbf{x}(t)$. A sketch of the algorithm (for one unit) is found below:
1. Initialize $\mathbf{w}$ randomly
2. Let $\mathbf{w}^+ = E\{\mathbf{x}g(\mathbf{w}^T\mathbf{x})\} - E\{g'(\mathbf{w}^T\mathbf{x})\mathbf{w}\}$
3. Let $\mathbf{w} = \mathbf{w}^+/\|\mathbf{w}^+\|$
4. Do steps 2 and 3 until convergence,
where g is the derivative of a nonquadratic function G (in this work, $G(u) = u^3$) (see [10] for details). Convergence means that vectors $\mathbf{w}^+$ and $\mathbf{w}$ point in the same direction.

Architecture: The ICA output is:

$$\mathbf{y}_{\mathrm{ica}}(t) = \mathbf{W}_{\mathrm{ica}}\mathbf{y}_{\mathrm{sfa}}(t), \tag{10}$$

where: $\mathbf{y}_{\mathrm{sfa}}(t)$ is the input vector at time t (the observed values); $\mathbf{W}_{\mathrm{sfa}}$ is the mixing matrix ($n_{\mathrm{ica}} \times n_{\mathrm{sfa}}$); and $\mathbf{y}_{\mathrm{ica}}(t)$ is the output of the ICA layer, which, in this work, learns to generate localized outputs which model hippocampal place cells of rats [1].

D. Robot and Controller

The robot used in the following experiments is the e-puck robot [17] extended with 8 infra-red sensors which can measure distances in the range [4-30] cm (see Fig. 2). For generating datasets with recorded sensor readings, we use a robot controller written in Matlab that communicates with the e-puck through a Bluetooth link. The controller performs basic wall following in the environment and it switches randomly to left or right wall following with a certain probability ρ. When the robot switches from right to left wall (or vice-versa), it can generate ellipse-like trajectories inside a room until it finds a wall to follow (see Fig. 4). One iteration (for sensing and acting) lasts 200 ms on average. The speed of the robot is not constant. The e-puck motor actuator sets the speed (steps/second) of a stepper motor (the maximum speed

Fig. 2. E-puck robot extended with longer-range infra-red sensors which can measure distance in the range 4 cm - 30 cm.

is 1000 steps per second). In this work, the actuator is limited to the interval $\pm$[15, 385] steps/s (or $\pm$[0.198, 5.08] cm/s). The 8 distance sensors are sequentially read while the robot is moving (our architecture is not modeled to correct these delays between sensor readings, but instead the reservoir may autonomously solve inconsistencies involved in this process because of the memory its recurrent connections provide). The signal $\mathbf{u}(t) \in [0, 1]$ is built by recording the 8 distance sensors during robot navigation and scaling them to the interval $[0, 1]$.

III. Experiments

A. Introduction

This section shows experiments with the e-puck robot in an unstructured environment with 2 rooms connected by a corridor (Fig. 3). The robot navigates in this environment according to the controller described in previous section. So, it can stay navigating in one room for a random time interval, eventually making ellipse-shaped trajectories or leaving the room towards the corridor (see Fig. 4). The randomness of the robot movement is determined by ρ (see previous section). We have made experiments with different settings ($\rho = 0, \rho = 0.004, \rho = 0.008$) and in all of them, place cells could be learned (the more random the movement, the more difficult the place cell learning). This section shows results considering the most random behavior ($\rho = 0.008$). We recorded the robot sensor readings in a dataset during 67 minutes of navigation in the environment (generating 18120 iterations of sensing-acting). The complete robot trajectory is shown in Fig. 4. For making this trajectory, we use a camera placed at the top of the environment and the ReactiVision recognition software [18] to record the position and heading of the robot during navigation. Fig. 4 also shows 25 labeled positions which partition the environment in smaller locations (strictly for analysis purposes).

In the following, we describe the initialization of parameters for the experiments in this section. The number of inputs are $n_{\mathrm{u}} = 8$ corresponding to the robot's distance sensors. The reservoir has $n_{\mathrm{res}} = 600$ neurons. The upper layers have each 4 units ($n_{\mathrm{sfa}} = n_{\mathrm{ica}} = 4$). The size of the SFA and ICA layers has a direct consequence on what type of place cells are learned. In [8], 70 units for each of the upper layers are used, generating around 60 place cells after learning (for a big maze). In the current work, 4 units are used because the environment is smaller and contains only 2 rooms. The SFA layer is associated with grid cells found in the entorhinal cortex of rats [1], for instance, in the simulated rat's experiment in [4]. As grid

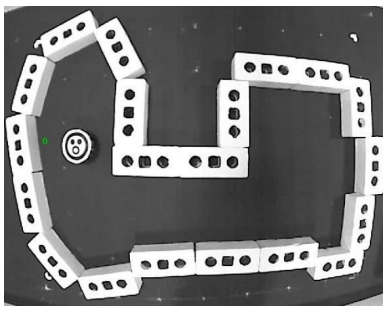

Fig. 3. Environment used for the experiments shown in this paper (width: 120 cm; height: 90 cm). It is composed of two rooms connected by a corridor (built with painted bricks).

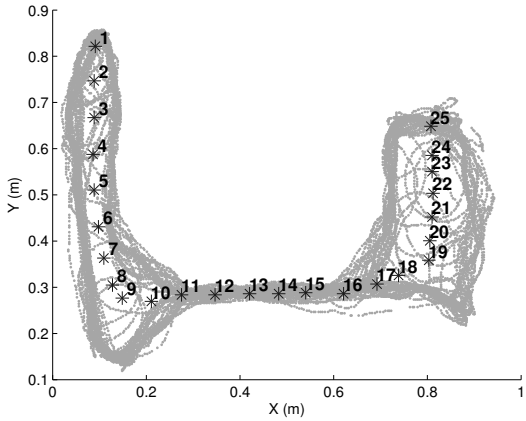

Fig. 4. Trajectory (in gray) generated by the robot controller in environment from Fig. 3 for 67 minutes. Twenty five (25) labeled asterisks are manually defined in this environment for analysis purposes.

cells are non-localized representation of space (they fire for more than a single location) [4], [8], sparse coding is used for learning place cells (the ICA layer). Previous work [8] only uses one timescale in the reservoir. For the current experiments with a real robot, better results were achieved with a reservoir having multiple timescales (leak rates set to $\alpha_1 = 0.05$ for half of the reservoir and $\alpha_2 = 0.15$ for the other half). These settings were necessary because the current robot has a variable speed and a more random movement behavior, thus requiring a reservoir with slow-processing and fast-processing neurons (in [8], the robot speed is constant and the robot behavior less random). The matrix connecting the input to the reservoir ($\mathbf{W}_{\mathrm{in}}$) is initialized to -2, 2 and 0 with probabilities 0.075, 0.075 and 0.85, respectively.

The RC-SFA architecture learns in steps, from the bottom SFA layer to the top ICA layer, and uses $7/8$ of the input signal (15855 timesteps) as the training dataset and $1/8$ (2265 timesteps) is used for testing. First, the SFA layer learns by solving (6) where the inputs are the reservoir states and distance sensors (like in (7)). After $\mathbf{W}_{\mathrm{sfa}}$ is found, the output of SFA units $\mathbf{y}_{\mathrm{sfa}}(t), t = 1, 2, ..., n_s$ is generated using (7). Afterwards, the ICA layer learns its connections weights using the FastICA algorithm from Section II-C where the inputs for

this layer are the output of the SFA units.

B. Results

This section shows the results after training the SFA and ICA layers with the setup presented in the previous section. Four different SFA and ICA units are generated after learning. We have observed that some units are most sensitive to the robot movement direction (not shown). We only show here the place cell which learned to be position invariant (see Fig. 5). This unit could distinguish between the two rooms of the environment even though the robot could stay in one room for a random time interval and generating random movements in it. In this sense, this place cell could be used for room detection (robot localization). Fig. 5 shows the response of the place cell (ICA unit 2) as a function of the robot position in the environment (first plot) and the place cell output over time (second plot). The colored dots represent the output of the ICA unit (where red denotes a peak response, green an intermediate response, and blue a low response). It is possible to see that this unit learned to detect in which room the robot is located, that is, high responses (red and yellow) in right room and low responses (blue) in left room.

Results on test data (trajectories unseen during learning) are shown in Fig. 6 for same ICA unit 2. It also shows that the learned place cell can distinguish between the rooms for new robot trajectories, which indicates its generalization capability. Note that when the robot first enters the right room, the ICA output is green for some time period until it turns to red. So, it takes some time until the unit responds with a high activity in this room.

We also tested the capability of RC-SFA architecture to recover from a kidnapping situation. For that, we used the same test dataset as in Fig. 6, but we kidnapped the robot twice at timesteps 150 and 640 and placed it back at timesteps 400 and 980 respectively. In other words, we shifted the robot from the right room to the left room and after some timesteps

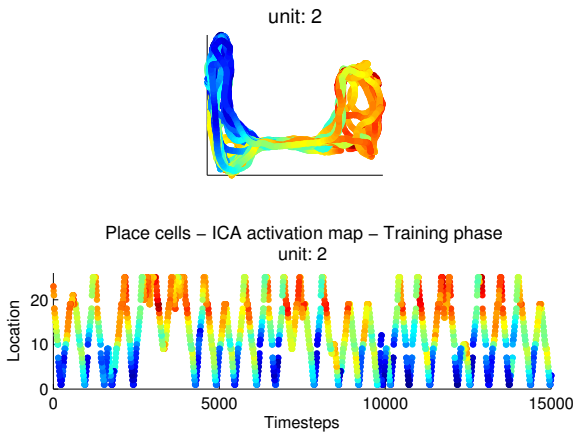

Fig. 5. Place cell representations on training data. Dots in red denote a peak response, in green an intermediate response, and in blue a low response. Top: Response of ICA units as a function of the robot position in the environment. Bottom: the ICA output over time. For each location (in time) given by the labeled asterisks in Fig. 3, there is a colored dot representing the ICA output.

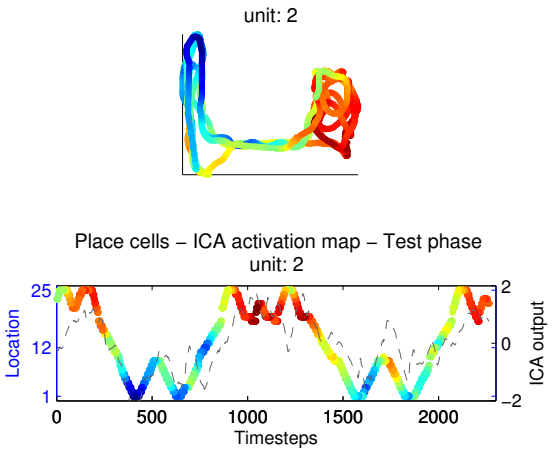

Fig. 6. Place cell representations on test data. Top: Response of ICA units as a function of the robot position in the environment. Bottom: the ICA output over time. The output is also plot as a black dashed line.

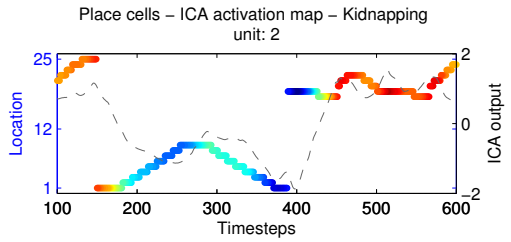

Fig. 7. Kidnapping the robot. The ICA output over time. For each location (in time) given by the labeled asterisks in Fig. 3, there is a colored dot where red denotes a peak response, green an intermediate response, and blue a low response. The output is also shown as a black dashed line.

from the left room to right room. The output of ICA unit 2 over time is shown in Fig. 7. It takes approximately 20 to 30 timesteps until the place cell recover itself from the kidnapping event and output a correct response. It is possible to note that the response transition of the unit is smooth despites the abrupt kidnapping event.

During the experiments, we have seen that the timescales present in the reservoir (leak rates) are very important for learning place cells. By tuning two distinct timescales in the reservoir and, in this way, empowering it with fast and slow processing neurons, we get the best performance in terms of the generated place cell. Currently, this fine-tuning is performed manually (there is no automated way yet). However, by ranging over different values for the leak rates, different place cells emerge from the unsupervised learning process. For instance, by setting the leak rates to $\alpha_1 = 0.11$ for one half of the reservoir and $\alpha_2 = 0.14$ for the other half, we get a place cell which can detect the corridor of the environment. Fig. 8 shows this place cell (ICA unit 1) which presented a peak response (red) when the robot crossed the corridor, but a low response (blue) otherwise.

The experiments presented in this section were repeated for many distinct randomly generated reservoirs. In each of these

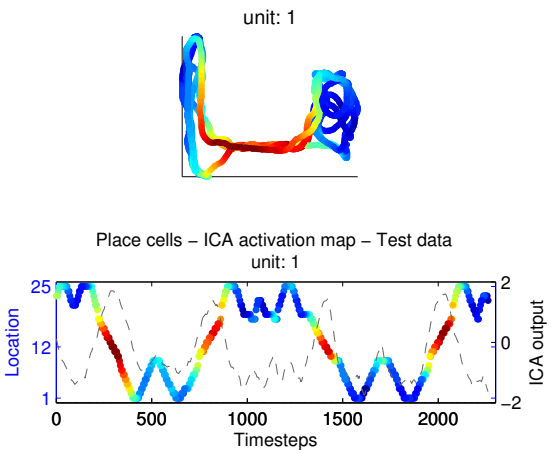

Fig. 8. Corridor detection with different timescales in the reservoir (on test data). Top: Response of ICA units as a function of the robot position in the environment. Bottom: the ICA output over time. The output is also plot as a black line.

runs, the learned place cells were qualitatively similar.

IV. Conclusion and Future Work

This work presented a new biologically inspired architecture (RC-SFA) which in principle can be applied to a wide range of applications (from speech recognition to modeling of several aspects of intelligent robots). By using RC-SFA, the current paper modeled place cells for autonomous learning of locations using an e-puck robot extended with 8 longer-range (4 cm - 30 cm) infra-red sensors.

The RC-SFA architecture is a hierarchical network composed of an untrained and randomly generated reservoir of recurrent neurons in the first layer, and two upper layers which learns by Slow Feature Analysis (SFA) [9] and Independent Component Analysis (ICA) [10], respectively. The SFA layer extracts the slow features from the reservoir states and distance sensors, generating signals which are invariant to fast-varying input signals but dependent on slowing-varying input signals. It is in this way that the robot position in the environment can be extracted from the distance sensors and the reservoir states. Finally, the ICA layer, by performing sparse coding, generates independent signals with spike-like responses for learning place cells.

In this work, place cells learned either to distinguish in which room the robot was or to detect the corridor connecting these two rooms. The type of place cell which can emerge from the unsupervised learning process depends on two main things: the timescales present in the reservoir; and the movement pattern of the robot. We have also shown that the place cells can recover nicely from kidnapping situations which are not present in the training dataset.

The advantages of the current approach include: the environment is unstructured; it works for robots of small dimensions and with cheap sensors; and no proprioceptive information (odometry) is necessary for predicting the location of the robot. This work is just a short investigation of the applications of the RC-SFA architecture, particularly to autonomous learning of the self-localization ability for small robots. Future work may be performed in modeling bigger environments with more rooms, including dynamic environments. Additionally, it would be very desirable to implement on-line learning techniques for the SFA and ICA layers so that the robot could learn while it moves in the environment.

Acknowledgment

This research is partially funded by the EC's project ORGANIC (FP7-231267). Eric Antonelo is sponsored by the Special Research Fund of Universiteit Gent (BOF).

References

[1] E. I. Moser, E. Kropff, and M.-B. Moser, "Place cells, grid cells and the brains spatial representation system," *Annual Reviews of Neuroscience*, vol. 31, pp. 69–89, 2008.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[3] A. Arleo, F. Smeraldi, and W. Gerstner, "Cognitive navigation based on nonuniform gabor space sampling, unsupervised growing networks, and reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 639–652, May 2004.

[4] M. Franzius, H. Sprekeler, and L. Wiskott, "Slowness and sparseness lead to place, head-direction, and spatial-view cells," *PLoS Computational Biology*, vol. 3, no. 8, pp. 1605–1622, August 2007.

[5] T. Stroesslin, D. Sheynikhovich, R. Chavarriaga, and W. Gerstner, "Robust self-localisation and navigation based on hippocampal place cells," *Neural Networks*, vol. 18, no. 9, pp. 1125–1140, 2005.

[6] M. Milford, R. Schulz, D. Prasser, G. Wyeth, and J. Wiles, "Learning spatial concepts from ratslam representations," *Robot. Auton. Syst.*, vol. 55, no. 5, pp. 403–410, 2007.

[7] R. Chavarriaga, T. Strsslin, D. Sheynikhovich, and W. Gerstner, "A computational model of parallel navigation systems in rodents," *Neuroinformatics*, vol. 3, pp. 223–241, 2005.

[8] E. A. Antonelo, B. Schrauwen, and D. Stroobandt, "Unsupervised learning in reservoir computing: Modeling hippocampal place cells for small mobile robots," in *International Conference on Artificial Neural Networks (ICANN)*, 2009, (accepted).

[9] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural Computation*, vol. 14, no. 4, pp. 715–770, 2002.

[10] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, pp. 411–430, 2000.

[11] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 148, 2001.

[12] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[13] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the European Symposium on Artifical Neural Networks (ESANN)*, 2007.

[14] H. Jaeger, M. Lukosevicius, and D. Popovici, "Optimization and applications of echo state networks with leaky integrator neurons," *Neural Networks*, vol. 20, pp. 335–352, 2007.

[15] B. Schrauwen, J. Defour, D. Verstraeten, and J. Van Campenhout, "The introduction of time-scales in reservoir computing, applied to isolated digits recognition," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, 2007.

[16] P. Berkes and L. Wiskott, "Slow feature analysis yields a rich repertoire of complex cell properties," *Journal of Vision*, vol. 5, pp. 579–602, 2005.

[17] F. Mondada, "E-puck education robot," September 2007, http://www.e-puck.org/.

[18] M. Kaltenbrunner and R. Bencina, "reactivision: a computer-vision framework for table-based tangible interaction," in *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*. New York, NY, USA: ACM, 2007, pp. 69–74.