Learning Slow Features with Reservoir Computing for Biologically-inspired Robot Localization

Eric A. Antonelo, Benjamin Schrauwen

Department of Electronics and Information Systems, Ghent University, Ghent, Belgium e-mail: eric.antonelo@elis.ugent.be

Abstract

This work proposes a hierarchical biologically-inspired architecture for learning sensor-based spatial representations of a robot environment in an unsupervised way. The first layer is comprised of a fixed randomly generated recurrent neural network, the reservoir, which projects the input into a high-dimensional, dynamic space. The second layer learns instantaneous slowly-varying signals from the reservoir states using Slow Feature Analysis (SFA), whereas the third layer learns a sparse coding on the SFA layer using Independent Component Analysis (ICA). While the SFA layer generates non-localized activations in space, the ICA layer presents high place selectivity, forming a localized spatial activation, characteristic of place cells found in the hippocampus area of the rodent's brain. We show that, using a limited number of noisy short-range distance sensors as input, the proposed system learns a spatial representation of the environment which can be used to predict the actual location of simulated and real robots, without the use of odometry. The results confirm that the reservoir layer is essential for learning spatial representations from low-dimensional input such as distance sensors. The main reason is that the reservoir state reflects the recent history of the input stream. Thus, this fading memory is essential for detecting locations, mainly when locations are ambiguous and characterized by similar sensor readings.

Keywords:
Reservoir Computing, Slow Feature Analysis, Independent Component Analysis, Place cells, Robot localization

1. Introduction

Traditional neural network models were designed to process static spatial input patterns, and are not inherently able to handle time-varying stimuli or dynamic patterns. To cope with temporal problems, these networks model time as an additional spatial dimension by dividing time into bins such that, for example, a 5-bin time window yields an input layer size of $5N_i$ (N_i is the number of input signals). In this way, time is treated as an additional spatial dimension at the level of the inputs, which is not a biologically plausible approach (Buonomano and Maass, 2009). In a second approach for representing time, neural networks models with recurrent connections allowed for computation based on the previous state of the network and the current sensory input, providing a mechanism of temporal context that still considered time as a discrete dimension (Buonomano and Maass, 2009).

The current work is based on the Reservoir Computing (RC) paradigm (Verstraeten et al., 2007), where a non-linear dynamical system (e.g. a recurrent neural network) is used to map the inputs to a high-dimensional space, in which classification or linear regression is easily accomplished. Therefore, the states of this dynamic reservoir are linearly combined in an output layer, which is the sole trained part of the architecture. This type of state-dependent computation has been proposed as a biologically plausible model for cortical processing (Buonomano and Maass, 2009; Maass et al., 2002; Yamazaki and Tanaka, 2007). Such theoretical models include: *Echo State Networks* (Jaeger and Haas, 2004) for analog neurons and *Liquid State Machines* (Maass et al., 2002) for spiking neurons. From a machine learning perspective, a reservoir network, usually randomly generated and sparsely connected, functions as a temporal kernel, projecting the input to a dynamic non-linear space. During simulation, the reservoir states form a trajectory which is dependent on the current external sensory input, but which still contains memory traces of previous stimuli. Computation in the output layer occurs by linearly reading out instantaneous states of the reservoir. In this way, reservoir architectures can inherently process spatiotemporal patterns.

Most reservoir computing models use supervised learning schemes to train the readout output layer. In this case, linear regression is the standard technique used for output training (Jaeger and Haas, 2004). However, biological systems probably learn a great number of tasks in an unsupervised way. Slow Feature Analysis (SFA) (Wiskott and Sejnowski, 2002) is an unsupervised learning method based on the concept of slowness. It extracts invariant or slowly-varying representations of a high-dimensional input signal, and has been shown to be able to model properties of complex cells from the primary visual cortex V1 (Berkes and Wiskott, 2005).

In this work, we propose a hierarchical architecture, where the first layer comprises a sparsely connected reservoir network with internal dynamics, and the second layer consists of SFA units. The short-term memory of the reservoir and its non-linear

projection in conjunction with such an unsupervised learning technique yields a model which possesses advantages from both theoretical models: the inherent spatiotemporal processing capabilities of the reservoir as well as the slowly-varying hidden signal extraction of the SFA model. This architecture is called RC-SFA model, first introduced in Antonelo et al. (2009) and Antonelo and Schrauwen (2009). This paper considerably extends previous works by making an elaborate investigation of the RC-SFA model, a range of insightful experiments (also showing the importance of the reservoir), and a comprehensive analysis of the localization capability mainly with respect to real-world robot experiments.

The slowness extraction mechanism present in SFA allows that high-level concepts, such as the position or orientation of a subject inside a room, which are slowly varying in time, be generated from low-level fast-varying stimuli like vision. In the same way, the location of a mobile robot inside an environment can be predicted from vision, but also from distance sensors, for instance. Supervised learning approaches using RC for robot localization based only on distance sensors have already been developed in Antonelo et al. (2008, 2007). This work goes beyond that by using SFA for learning slowly-varying spatial representations in an unsupervised way from a high-dimensional reservoir space, which is excited by the robot's distance sensors.

In our architecture, SFA units show an activation which is spatially non-localized in the considered environment, that is, after training, their activation is high for multiple and specific locations of an environment, showing low place selectivity. A second step is necessary for producing units which are only active for particular locations. In Franzius et al. (2007a) an additional post-processing step using Independent Component Analysis (ICA) is applied for learning sparse representations from SFA units. Similarly, in this work, ICA is used in the third layer for generating localized representations of a robot environment. The complete architecture is shown in Fig. 1.

Experiments accomplished with freely moving rats in circular or rectangular open fields show the existence of two types of spatial encoding cells in the brain of rodents: hippocampal place cells and grid cells from the entorhinal cortex. Place cells form an implicit spatial representation of an animal's environment, firing whenever the rodent is located at a particular location (O'Keefe and Dostrovsky, 1971; O'Keefe, 1976; Moser et al., 2008), which defines the place field of the cell. Another type of spatial activation is found in grid cells of the entorhinal cortex. The activation of these cells have shown to follow a grid pattern in circular open fields and probably have an important role in the formation of place cells (Moser et al., 2008).

It is assumed that two classes of input are available for spatial encoding cells: idiothetic and allothetic. While allothetic inputs are originated from the external environment (vision, tactile and auditory signals), idiothetic input could be formed by self-motion (proprioceptive) signals. The origin of these idiothetic signals and the mechanisms for the integration of these signals with allothetic input have not yet been determined (Moser et al., 2008).

Although the majority of the existing place cell models are not dependent on the animal's direction of movement, there are few experiments showing that place cells exhibit movement-related firing patterns such that the environment configuration and the animal's behavior can impose a directional structure in the firing of place cells (Eichenbaum et al., 1999). In Brunel and Trullier (1998), it is proposed that place cells are intrinsically directional and that invariance to direction is achieved through generalization. In Frank et al. (2000), using a constrained environment for rats such as W tracks, it is shown that hippocampal CA1 cells and entorhinal cortex (EC) cells code for spatial information on a way dependent to the rat's path or behavior. It assumes that, if the hippocampus and EC are related to path planning over extended trajectories, these structures should reflect where the animal intends to go or where it has come from. Similarly, the proposed RC-SFA architecture in this work encodes positional information on a path-dependent way, where the SFA layer and ICA layer exhibit an activation pattern comparable to that of EC cells and hippocampal CA1 cells found in Frank et al. (2000), respectively.

The current work is inspired by the fact that whiskers of rodents can provide relevant information about the environment and shape of objects (Solomon and Hartmann, 2006). In the same way, the experiments in this paper are based on a small mobile robot which perceives the environment through a limited number of short-range distance sensors. We assume that this low-dimensional input, such as whiskers for rats or distance sensors for robots, can provide interesting information from the environment.

In mobile robotics, the problem of robot localization is usually referred as the SLAM problem, which stands for Simultaneous Localization And Mapping. In our architecture, the robot localization capability emerges in a self-organized way at the upper ICA output layer. Distinct ICA outputs code for different locations in the robot's environment after learning. Is this localization capability comparable to standard robot localization frameworks such as Markov localization (Thrun et al., 2005) or Kalman filtering techniques (Siegwart and Nourbakhsh, 2004)? It is comparable to some extent, with advantages and disadvantages of each approach. Probabilistic approaches to robot localization, usually based on expensive laser-range scanners, have to take into account the following points: the modeling of the noise of each sensor, the kinematic model of the robot, the costly drawing of a priori map of the environment or a mechanism for map building during navigation, matching sensory data to the stored map representation for the correction of position estimation, and so on. In contrast to this, models of biologically-inspired localization and navigation systems, such as artificial neural networks-based models, tend to overcome these limitations and costly processes by learning implicit representations of the environment from sensor data.

In traditional localization algorithms, the representation of the stored map can be grid-based, topological, or hybrids (Siegwart and Nourbakhsh, 2004). Grid-based methods produce metric maps and have high resolution, while topological maps are more abstract and describe the environment as a graph of connected nodes. The advantages of probabilistic models are: easy human interpretation of the robot position in the stored

map; accurate descriptions of the map for grid-based methods (Siegwart and Nourbakhsh, 2004); and efficient planning in topological maps (Thrun et al., 2005). Given some assumptions, state-of-the-art algorithms, such as the FastSLAM method (Thrun et al., 2005), are able to solve the SLAM problem. Although this work does not aim to solve the SLAM problem, it is an interesting biologically-inspired model which can autonomously learn locations in an environment from a limited number of inexpensive infra-red distance sensors. Furthermore, the proposed model does not depend on the use of odometry, but it is based on the powerful temporal processing of the reservoir as a form of short-term memory of previous inputs.

This paper is organized as follows. Section 2 describes the methods used in this work, namely, Reservoir Computing, Slow Feature Analysis and Independent Component Analysis, as well as the robot models and the place cell reconstruction method used in the experiments. Sections 3 and 4 describe the experiments performed with simulated and real robots, respectively, using the RC-SFA architecture. Finally, Section 5 presents the conclusions, related works and future directions for research.

2. Methods

2.1. Reservoir Computing

The Reservoir Computing (RC) model for analog neurons, the Echo State Network (Jaeger, 2001), is modeled by the following state update equation:

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

where: $\mathbf{u}(t)$ denotes the input at time t; $\mathbf{x}(t)$ represents the reservoir state; α is the leak rate (Jaeger et al., 2007);

and $f() = \tanh()$ is the hyperbolic tangent activation function (the most common type of activation function used for reservoirs).

Connections between units in the reservoir are represented by the weight matrix $\mathbf{W}_{\text{res}}$, while the inputs' connections to the reservoir are given by $\mathbf{W}_{\text{in}}$. The initial state of the dynamical system is $\mathbf{x}(0) = \mathbf{0}$. The above equation considers that reservoir units are leaky integrators (Jaeger et al., 2007). A standard reservoir equation is found when $\alpha = 1$.

The input weight matrix $\mathbf{W}_{\text{in}}$ as well as the reservoir weight matrix $\mathbf{W}_{\text{res}}$ are not trained during the simulations but, instead, they are generated from a random distribution, such as a Gaussian distribution or a discrete set. The dynamic regime of the reservoir has traditionally been tuned by scaling the connection matrix $\mathbf{W}_{\text{res}}$ such that its spectral radius $|\lambda_{max}| < 1$, where $|\lambda_{max}|$ is the largest absolute eigenvalue of $\mathbf{W}_{\text{res}}$ (Jaeger, 2001). Although it does not guarantee the Echo State Property (e.g. a reservoir with fading memory), in practice this criterium produces good reservoirs. Besides the reservoir weight matrix, the dynamics of the reservoir is determined by several factors: the input scaling of $\mathbf{W}_{\text{in}}$, an optional bias, the nonlinearity of the nodes and the external input driving the reservoir (Verstraeten and Schrauwen, 2009).

In this work, the dynamics of the reservoir is tuned by empirically choosing the input scaling for $\mathbf{W}_{\text{in}}$ as well as the spectral radius of $\mathbf{W}_{\text{res}}$, which is fixed at $|\lambda_{max}| = 0.99$ for all experiments. Further optimization on these parameters are shown in Section 4.2.4, which also indicates that the chosen empirical values are close to the optimum.

Usually, the connection matrices $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{res}}$ are created considering sparse connectivity, but that is not mandatory (Schrauwen et al., 2008).

There are two ways to increase the memory of a reservoir without output feedback. It is possible to either tune the leak rate $\alpha \in (0, 1]$ of the reservoir (Jaeger et al., 2007) for matching the timescale of the input signal or downsample the input signal. Low leak rates yield reservoirs with more memory which can *remember* the previous stimuli for longer time spans. On the other hand, leak rates close to 1 are suitable for high-frequency input signals which vary in a faster timescale. Sometimes a combination of reservoir units possessing distinct leak rates can improve performance (Antonelo et al., 2008), for example, when the input signal consists of components operating in distinct timescales. Other approaches include the use of band-pass filters in reservoir units (Wyffels et al., 2008) which allow for very specific frequency sensitivity. The configuration of several parameters are given in subsequent sections.

The readout output layer in reservoir systems is usually trained with linear regression methods such as the Least Squares method or Ridge Regression, which are supervised methods. The training uses a matrix whose rows are made of the reservoir's states generated during the stimulation of an input signal in the reservoir. In this work, the output layer is replaced by a SFA layer which learns in an unsupervised way. This method is described in the following section.

Next, consider N_{u} as the number of inputs; N_{res} as the number of neurons in the reservoir; N_{SFA} as the number of SFA units; and N_{ICA} as the number of ICA units.

2.2. Slow Feature Analysis

Most sensory input signals vary in a fast timescale, even though the environment properties may be slowly varying. This is because sensors provide low-level representations of environment features and are prone to fast signal variations, e.g., a human moving inside an office produces fast visual input variation while his/her position in the building changes slowly. Slow Feature Analysis (SFA) (Wiskott and Sejnowski, 2002) is an algorithm which finds output functions $g_i(\mathbf{x}(t))$ which maximize temporal slowness, given a high-dimensional input signal $\mathbf{x}(t)$. It extracts functions which provide a higher level representation of the environment, assuming that they vary in a slower timescale when compared to the raw input. The SFA output is an instantaneous function of the input so that it depends only on the current state, differently from a filter which depends on previous inputs.

In mathematical terms (Wiskott and Sejnowski, 2002), the SFA model tries to find output signals $y_i = g_i(\mathbf{x}(t))$ such that:

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

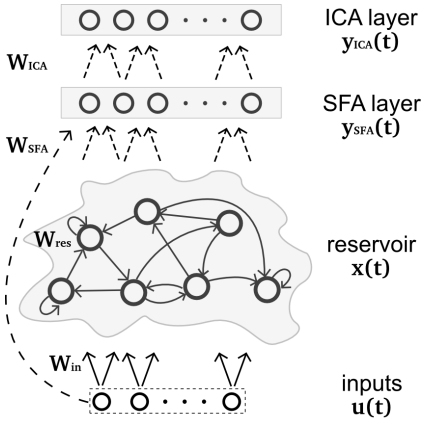


Figure 1: RC-SFA architecture. The reservoir is a recurrent network where the inputs are mapped to a high-dimensional non-linear space. The resulting reservoir trajectory, generated by input stimulation, is used for training the SFA layer, which extracts instantaneous slowly-varying signals from the reservoir after learning. The subsequent ICA layer implements sparse coding on the SFA outputs, extracting independent components from the SFA activation. Training is unsupervised and takes place for the dashed connection lines in the figure. Solid lines represent fixed randomly generated weights.

under the constraints

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

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

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

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

Learning: The first step of the learning process is normalizing the input signal $\mathbf{x}(t)$ to have zero mean and unit variance.

The common step of non-linear expansion of the input signal is not used in this work, but it is replaced by the non-linear reservoir at the first layer of the RC-SFA architecture. It can be shown that SFA learning corresponds to solve a generalized eigenvalue problem (Wiskott and Sejnowski, 2002):

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

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 $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{N_{\text{SFA}}}$ corresponding to the ordered generalized eigenvalues $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{N_{\text{SFA}}}$ solve the learning task, satisfying (3-5) and minimizing (2) (see Wiskott and Sejnowski, 2002, for more details). This algorithm is guaranteed to find the global optimum. Learning and inference is very fast, as there are efficient methods for solving the generalized eigenvalue problem. The decorrelated SFA outputs extract instantaneous slowly-varying signals, which is different from low-pass filtering of the inputs.

Although the eigenvalue problem for solving SFA is biologically unrealistic, biologically plausible implementations of SFA exist(Hashimoto, 2003).

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

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

where $\mathbf{x}_{\text{SFA}}(t)$ 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 stimulating the reservoir with the input signal $\mathbf{u}(t)$ for $t = 1, 2, ..., N_{\text{s}}$ by using (1), where N_{s} is the number of samples.

The weight matrix $\mathbf{W}_{\text{SFA}}$ is a $N_{\text{SFA}} \times (N_{\text{u}} + N_{\text{res}})$ matrix corresponding to the eigenvectors found by solving (6).

2.3. Independent Component Analysis

Independent Component Analysis (ICA) is a method used for sparse coding of input data as well as for *blind source separation* (Hyvärinen and Oja, 2000). The ICA model assumes 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 $\mathbf{W} = \mathbf{A}^{-1}$.

The basic assumption for ICA is that the components s_i are statistically independent. It is also assumed that the independent components have non-Gaussian distributions (Hyvärinen and Oja, 2000).

Learning: In this work the matrix $\mathbf{W}$ is found with the FastICA algorithm (Hyvärinen and Oja, 2000). Before using ICA, the observed vector $\mathbf{x}(t)$ is preprocessed by centering (zero-mean) and whitening (decorrelation and unit variance) (Hyvärinen and Oja, 2000). FastICA uses a fixed-point iteration scheme for finding the maximum of the non-Gaussianity of $\mathbf{wx}(t)$ (where $\mathbf{w}$ is a weight vector of one neuron). The basic form of the FastICA algorithm (for one unit) is described next:

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,

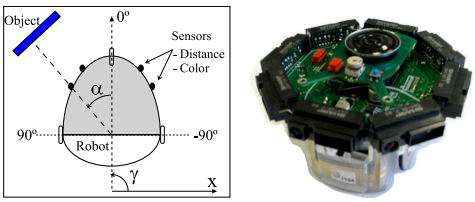
where g is the derivative of a nonquadratic function G (Hyvärinen and Oja, 2000). Convergence means that vectors $\mathbf{w}^+$ and $\mathbf{w}$ point in the same direction. The next units $\mathbf{w}_i$ in $\mathbf{W}$ are found one by one such that the outputs $\mathbf{w}_i^T\mathbf{x}$ are decorrelated.

Whereas the FastICA algorithm may seem biologically unrealistic, an alternative biologically plausible implementation of ICA can be achieved through non-linear Hebbian learning (Hyvärinen and Oja, 1998).

Architecture: The equation for the ICA layer is (by redefining variables):

$$\mathbf{y}_{\text{ICA}}(t) = \mathbf{W}_{\text{ICA}}\mathbf{y}_{\text{SFA}}(t), \tag{10}$$

where: $\mathbf{y}_{\text{SFA}}(t)$ is the input vector at time t (the observed values); $\mathbf{W}_{\text{ICA}}$ is the mixing matrix ($N_{\text{ICA}} \times N_{\text{SFA}}$); and $\mathbf{y}_{\text{ICA}}(t)$ is the output of the ICA layer (the independent components).



(a) SINAR (b) E-puck

Figure 2: (a) SINAR robot model with distance and color sensors positioned in the frontal part of the robot (–90 to 90). (b) E-puck robot with an additional turret with 8 infra-red sensors capable of reading distances from 4 cm to 30 cm.

2.4. Robot Models

2.4.1. SINAR

SINAR is a 2D autonomous robot simulator introduced in Antonelo et al. (2006), where the mobile robot (Fig. 2(a)) interacts with the environment by distance and color sensors; and by one actuator which controls the movement direction (turning). The SINAR robot model has seventeen (17) sensor positions distributed uniformly over the front of the robot, from -90° to +90°. Each position holds two virtual sensors for distance and color perception. The distance sensors are limited in range such that they saturate for distances greater than 300 distance units (d.u.), and are noisy - they exhibit Gaussian noise $N(0, 0.01)$ on their readings. A value of 0 means near some object and a value of 1 means far or nothing detected. At each iteration the robot is able to execute a direction adjustment to the left or to the right in the range [0, 15] degrees and the speed is constant (0.28 distance units (d.u.)/s). The SINAR controller, described in Antonelo et al. (2006), is a reactive intelligent navigation system made of hierarchical neural networks which learn by interaction with the environment. After learning, the robot is able to efficiently navigate and explore environments, during which the signal $\mathbf{u}(t)$ is built by recording the 17 distance sensors of the robot.

2.4.2. Real robot e-puck

The e-puck (Mondada, 2007) is a small differential wheeled robot which was built primarily for education purposes, but has been largely adopted in research as well. The mobile robot has a diameter of 7 cm and is equipped with 8 infra-red sensors which measure ambient light and proximity of obstacles in a range of 4 cm originally, which effectively restricts the ability to read distances to obstacles. Because of this, an extension turret for the e-puck was built with 8 longer-range infra-red sensors capable of measuring distances in the interval [4-30] cm (see Fig. 2(b)).

For recording datasets containing the robot's sensor readings, a controller written in Matlab steers the e-puck robot through a Bluetooth connection. This controller performs basic wall following throughout 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 may generate ellipsoid trajectories inside a room until it finds a wall to follow (see Fig. 8(b)). One iteration, for reading the distance sensors as well as for motor actuation, lasts 200 ms. The actuator sets the speed (steps/second) of a stepper motor, where the maximum speed is 1000 steps per second. In this work, the actuator is limited to the interval ±[15, 385] steps/s (or ±[0.198, 5.08] cm/s).

The eight distance sensors are sequentially read in groups of 2 while the robot is moving, that is, there are 4 cycles of sensors reading, where each cycle corresponds to 2 simultaneous readings. Considering an acquisition time of 25 ms on average for a cycle, the total time spent on sensor reading is between 100 and 120 ms.

Any resulting inconsistencies from this sequential sensor reading during robot movement are not taken into account when using the RC-SFA architecture, so that learning has to cope with this additional problem.

The signal $\mathbf{u}(t)$ is built by recording the eight distance sensors during robot navigation and scaling them to the interval $[0, 1]$.

For analysis purposes, the robot position and orientation are estimated using pictures taken from a fish-eye camera placed on a structure localized above the environment. Robot recognition and pose tracking are accomplished using the ReactiVision software (Kaltenbrunner and Bencina, 2007).

2.5. Place cell reconstruction

It is common to use population vector coding for interpreting activation from hippocampal place cells (Zhang et al., 1998). However, Bayesian methods have shown superior performance for reconstructing the position of freely moving rats and were shown to be biologically plausible (Zhang et al., 1998). In this work, we use a probabilistic method based on Zhang et al. (1998) to estimate the robot position from the activation of ICA units of our architecture. The reconstruction is based on the conditional probability

$$P(\mathbf{x_r}|\mathbf{y}_{\text{ICA}}) = \frac{P(\mathbf{y}_{\text{ICA}}|\mathbf{x_r})P(\mathbf{x_r})}{P(\mathbf{y}_{\text{ICA}})}. \tag{11}$$

The prior $P(\mathbf{x_r})$ is the probability of the robot being at position $\mathbf{x_r} = (x, y)$ and can be computed with the true recorded position during robot navigation. $P(\mathbf{y}_{\text{ICA}})$ is a normalization term which does not need to be explicitly computed. $P(\mathbf{y}_{\text{ICA}}|\mathbf{x_r})$ is the probability of the activation $\mathbf{y}_{\text{ICA}}$ given that the robot is at location $\mathbf{x_r}$. As ICA units $\mathbf{y}_{\text{ICA}}$ are statistically independent, the conditional probability can be computed as:

$$P(\mathbf{y}_{\text{ICA}}|\mathbf{x_r}) = \prod_{i=1}^{N_{\text{ICA}}} P({y^i}_{\text{ICA}}|\mathbf{x_r}) \tag{12}$$

where $P({y^i}_{\text{ICA}}|\mathbf{x_r})$ is the probability of ICA unit i given that the robot is at position $\mathbf{x_r}$, which can be approximated by computing the histogram of the data $P({y^i}_{\text{ICA}}|\mathbf{x_r}) = \langle {y^i}_{\text{ICA}} \rangle_{\mathbf{x_r}} / \mu$, where $\langle {y^i}_{\text{ICA}} \rangle_{\mathbf{x_r}}$ is the mean activation of ICA unit i in position $\mathbf{x_r}$ and μ is a normalization factor.

The reconstructed position is given by:

$$\hat{\mathbf{x}}_\mathbf{r} = \arg\max_{\mathbf{x_r}} P(\mathbf{x_r}|\mathbf{y}_{\text{ICA}}) \tag{13}$$

which is the most probable position given the ICA layer activation.

3. Learning to Localize a Simulated Robot

3.1. Introduction

This section shows that, using the RC-SFA architecture, the capability of self-localization of a small simulated mobile robot emerges in a self-organized way. The SINAR robot model has 17 short-range noisy distance sensors (see Section 2.4.1). Based only on the information from these sensors, the RC-SFA architecture can autonomously learn an internal representation of the environment which allows for spatial coding and self-localization.

3.2. Experiments

The experiments are conducted using environment E1 (Fig. 3), a big maze with 64 predefined locations spread evenly around the environment (represented by small labeled triangles). Additional investigations are also performed with a modified version of the environment containing 11 obstacles which randomly move around the environment, representing an extra source of noise which also changes the robot behavior and trajectory in the environment.

The experiments are built as follows. First, for generating the input signal, the simulated robot navigates in the environment for 180.000 timesteps while its distance sensor measurements are recorded. It takes approximately 13.000 timesteps for the robot to visit most of the predefined locations with the SINAR controller described in Section 2.4.1, which basically makes the robot explore the whole environment. After downsampling the recorded input signal $\mathbf{u}(t), t = 1, ..., 180.000$, the number of samples is reduced to $n_s = 3.600$. Next, the downsampled input signal is used to generate the reservoir states $\mathbf{x}(t), t = 1, 2, ..., n_s$ using (1).

For model optimization, we perform multiple grid search experiments over subsets of the model parameters, using the place cell reconstruction method from Section 2.5 for position estimation.

3.2.1. Settings

The optimal combination of the model parameters are given in the following. The reservoir has $N_{res} = 300$ neurons. The SFA and ICA layers consists of 128 units each. The nonlinearity $g(u) = u^3$ is used for the fixed-point ICA algorithm from Section 2.3.

The input signal is downsampled by a factor of $d_t = 50$ (using the matlab function *resample*), which reduces the number of samples given the low speed of the robot. The leak rate in the reservoir is $\alpha = 0.4$.

The weight matrix $\mathbf{W}_{\text{in}}$ is initialized to -0.9, 0.9 and 0 with probabilities 0.15, 0.15 and 0.7, respectively (which means an input scaling of 0.9). While the connectivity between units is not that important (Schrauwen et al., 2008), the scaling of the input connections have a great influence on the reservoir dynamics (Verstraeten et al., 2007).

A summary of the configuration parameters is given in Table 1, where values in bold denote parameters found by optimization, and d_t is the downsampling rate of the input signal.

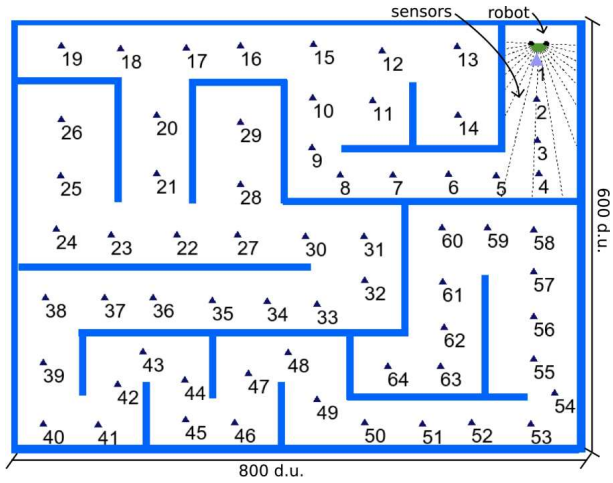


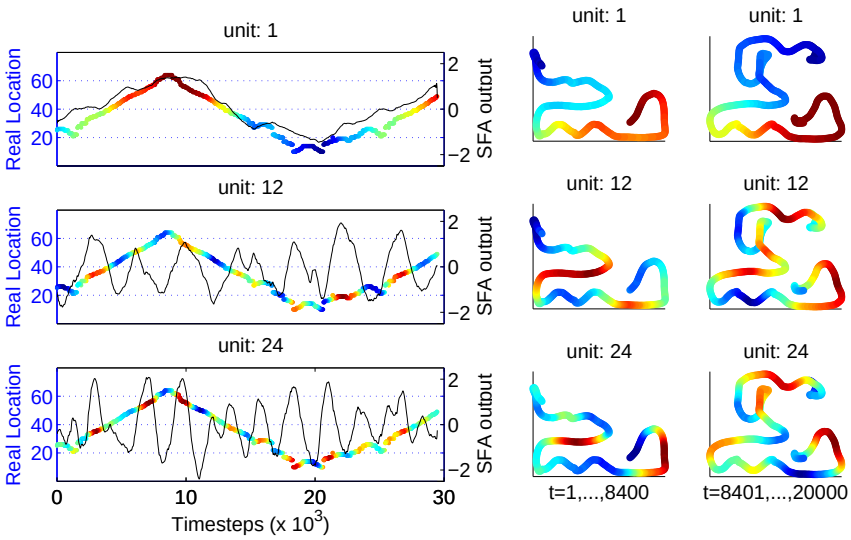
Figure 3: (a) Environment E1. The environment is tagged with 64 labels displayed by small triangles.

The training of the RC-SFA architecture is accomplished in 2 steps and uses 5/6 of the input signal as the training dataset (1/6 for testing). First, the SFA layer is trained by solving (6) where the inputs are the reservoir states and distance sensors (like in (7)). After $\mathbf{W}_{\text{SFA}}$ is found, the output of SFA units $\mathbf{y}_{\text{SFA}}(t), t = 1, 2, ..., N_s$ is generated using (7). The second step corresponds to training the upper ICA layer by applying the FastICA algorithm from Section 2.3 where the inputs for this layer are the output of the SFA units. The output signals $\mathbf{y}_{\text{SFA}}(t)$ and $\mathbf{y}_{\text{ICA}}(t)$ are upsampled to the original sampling rate of $\mathbf{u}(t)$.

3.2.2. Results

Fig. 4 show the output of 3 SFA units for a test input signal. The left plot, Fig. 4(a), show the outputs over time whereas the right plots, Fig. 4(b), show the response of the neurons as a function of the robot position in the environment. In the left plot, the horizontal axis represents the time, the left vertical axis denotes the real robot location (as given by the labeled triangles in Fig. 3), and the right vertical axis denotes the SFA output of the neuron. The colored dots represent the output of the SFA unit (where red denotes a peak response, green an intermediate response, and blue a low response). The SFA output is also shown as a black line in the same plot and as a colored trajectory in the right plot. As SFA units are ordered by slowness, the first SFA unit has the slowest response. It shows a high response for locations 40 to 64 and a low response otherwise. Units 12 and 24 vary much faster, encoding several locations of the environment. In Fig. 4(b), each of the units is shown during two different time intervals, [1, 8400] and [8401, 20000]. In that way, it is possible to observe that units 12 and 24 encode spatial information in a way which is dependent on the robot path, that is, their activation depends on where the robot has come from in the environment (a characteristic comparable to EC cells of rats in Frank et al., 2000).

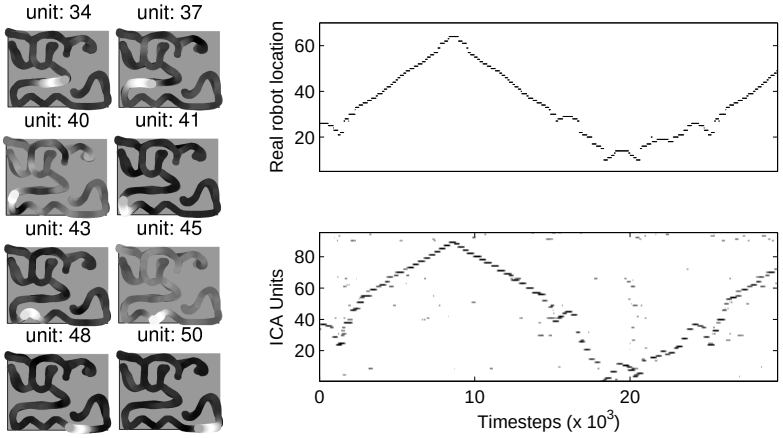
The upper ICA layer builds on the SFA layer. During learning, ICA units seek to maximize non-Gaussianity so that their responses become sparse and clustered, and also as independent as possible. This form of sparse coding leads to the unsu-



(a) SFA output over time

(b) SFA output as a function of the robot position

Figure 4: Response of SFA units 1, 12, and 24 for simulations in Environment E1 on test data. (a) the SFA output over time. For each location (in time) given by the labeled triangles 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 plot as a black line. (b) the same SFA output as a function of the robot position for two distinct time intervals $[1, 8400]$ and $[8401, 20000]$.



(a) ICA output

(b) ICA Activation map

Figure 5: Response of ICA units for simulations in Environment E1 on test data. (a) Response of ICA units as a function of the robot position. White dots denote high activity while darker dots represent lower responses. The results show the localized aspect of place cells or ICA units (the peak response is characteristic of one specific location). (b) The real robot occupancy grid (top) and the respective spatially-ordered ICA activation map (bottom), where black dots denote peak responses and white represent lower responses.

Table 1: Parameter configuration

Environment (Robot)	N_{u}	N_{res}	N_{SFA}	N_{ICA}	d_t	α	$\mathbf{W}_{\mathrm{in}}$
E1 (SINAR)	17	**300**	**128**	**128**	**50**	**0.4**	$\{\pm 0.9, 0\}$
E2 (E-puck)	8	**600**	**128**	**128**	**1**	**0.1**	$\{\pm 2, 0\}$

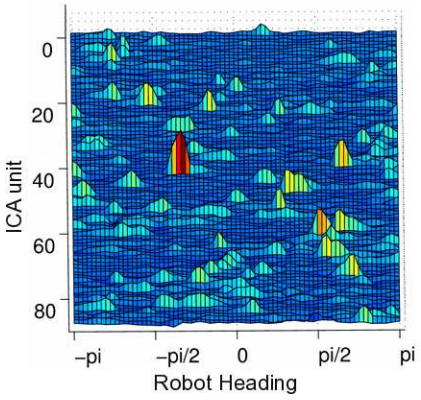


Figure 6: The plot shows the mean activation of ICA units as a function of the robot heading, which reveals the dependence of trained ICA units on the direction of movement. Red denotes high response while blue a low response.

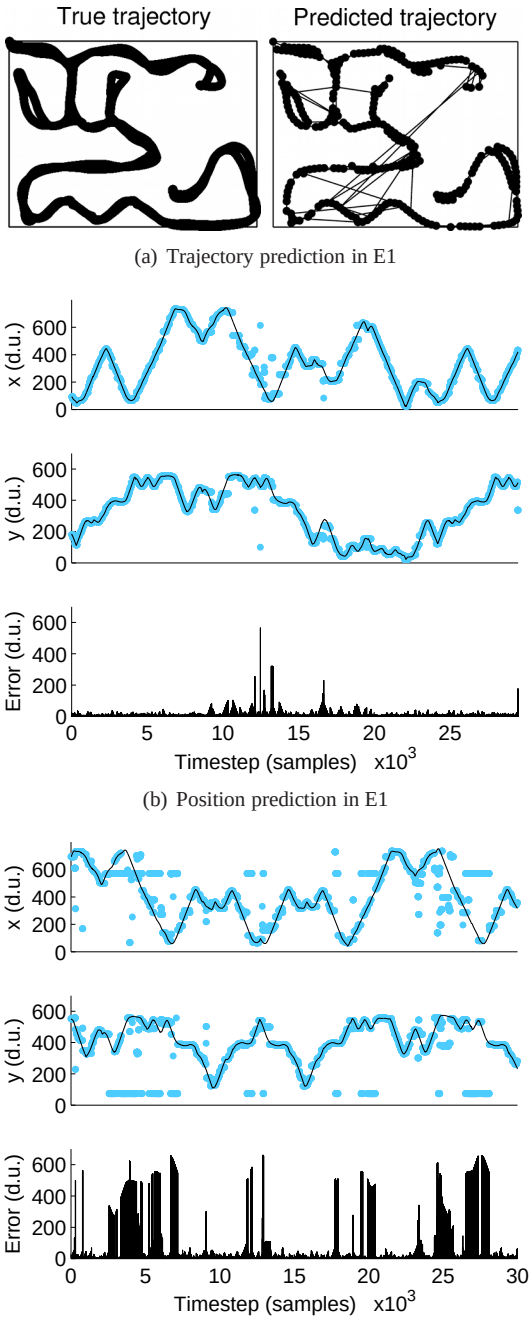
pervised formation of place cells. Fig. 5(a) shows a number of ICA units which code for specific adjacent locations in the environment. The peak response is represented by white dots while lower responses are given gradually in darker colors. As the robot navigates, a sequence of high activity spots (white dots) is observable through these set of ICA units, each one coding for a specific location in the environment.

In order to visualize the localized property of place cells more clearly, the output of ICA units are ordered such that they have a spatial relationship. The reference locations (from 1 to 64), shown in environment E1 (Fig. 3), are used to automatically order the ICA layer. ICA units which do not respond strongly enough in any situation are removed from the vector. Fig. 5(b) shows the real occupancy grid for the robot while it drives in environment E1 and the respective ICA activation map showing the spatially-ordered ICA responses (where $\mathbf{u}(t)$ is a test signal not used during learning). Stronger responses are represented by darker dots in the figure. This activation map is very similar to the real robot occupancy grid showing that the place cells efficiently learned to cover most of the locations in the environment.

We have repeated the experiments shown here 15 times with the same datasets, where each time a different random reservoir is created. These distinct experiments have not shown any significant differences in the quality of the learned place cells.

Most ICA units show an activation which is dependent on the direction of the robot's movement. This can be visually confirmed in Fig. 6, where a blue surface represents a low activation and a red surface means high activation. It shows that the activation of several ICA units (vertical axis in the figure) is, on average, high only for particular robot directions (horizontal axis in the figure). This is comprehensible since the environment is composed of narrow corridors, which shapes the robot trajectory so that the orientations that the robot may have are restricted by the environment configuration. So, as the robot direction is not, in general, a constantly fast-varying feature, it is also learned by SFA and ICA units.

Using the probabilistic place cell reconstruction method from Section 2, the predicted robot position during robot navigation



(a) Trajectory prediction in E1

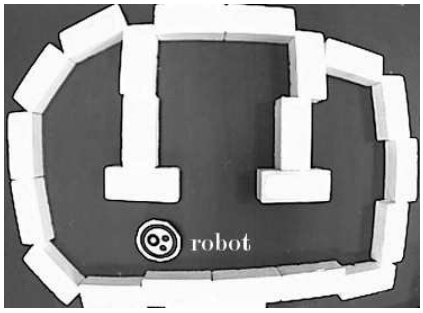
(b) Position prediction in E1

(c) Position prediction in E1 with dynamic obstacles

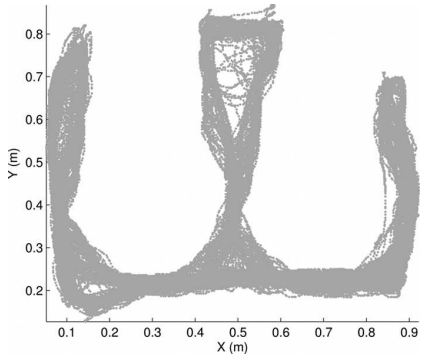
Figure 7: Prediction of the robot position in environment E1 given the activation in the ICA layer using the place cell reconstruction method on test data. (a) The true robot trajectory as connected black points (left) and the corresponding estimated robot trajectory by the place cell reconstruction method given the activation in the ICA layer (right). (b) The true and predicted robot coordinates given by black curves and points in cyan color (gray for black-and-white prints), respectively. The bottom plot shows the error as the Euclidean distance between true and predicted position. (c) The true and predicted robot position in a modified version of environment E1, containing 11 dynamic moving obstacles. Mistakes can be observed when the predicted points in cyan (gray for black-and-white prints) deviate from the black line (also detected by the error plot).

is computed given the activation in the ICA layer. The true and predicted trajectory can be seen in Fig. 7(a) showing that the RC-SFA architecture is able to learn an internal spatial representation of the environment. Some jumps in the predicted position can be observed, which also occur with the estimated position computed with signals recorded from hippocampal place cell of rats (Moser et al., 2008). Fig. 7(b) shows the same true and predicted positions in terms of the x and y coordinates of the robot in the world frame along with the respective error, given by the Euclidean distance between true (black line) and the predicted (points in cyan color) positions. The mean test error is 17.2 distance units - see Table 2.

Another experiment with environment E1 is performed, in which 11 dynamic obstacles were artificially added to the environment. These obstacles were constantly moving around in a random way, possibly closing passages and forcing the robot to follow another way. That yields more stochasticity in the environment and in the robot behavior. The recorded dataset consists of 200.000 samples. Training and parameter configuration are the same as in the previous experiment. The test error of 108 d.u., shown in Table 2, is clearly higher than when the environment is not dynamic. Fig. 7(c) shows the network predictions as points in cyan and the true position as a black curve. Despite this higher error rate, the trained system is robust enough to recover from intense environment stochasticity given by dynamic obstacles and a period of miss-predictions without the use of odometry, being also able to recover from robot kidnapping situations as in Antonelo et al. (2008).



(a) Environment E2



(b) Trajectory in environment E2

Figure 8: (a) Environment E2 (120 cm x 90 cm), made of 3 rooms and a connecting corridor. The position of the robot is tracked with a camera placed above the environment for analysis purposes. (b) Trajectory (in gray) generated by the robot controller in environment E2 for 60.000 timesteps (or 3.3 hours).

4. Learning to Localize a Real Robot

4.1. Introduction

In the previous section, experiments showed that the RC-SFA architecture can learn spatial representations from a maze-like simulation environment based on information from 17 sensors. This section elaborates on experiments with the e-puck robot in real environments. The robot has only 8 infra-red sensors which measure distances to environment walls. This setup is more difficult for two main reasons: stochasticity of the robot controller and a limited number of sensors. If enough training samples can be collected, then the architecture can autonomously learn to encode spatial information.

4.2. Experiments

This section shows results considering a real environment with 3 rooms and a connecting corridor (see environment E2 in Fig. 8(a)). The robot navigates in this environment according to the controller described in Section 2.4.2. So, it can stay navigating in one room for a random time interval, eventually making ellipsoid trajectories or leaving the room towards the corridor (see Fig. 8(b)). The randomness of the robot movement is determined by ρ (see Section 2.4), which is the probability of changing the movement direction at each second. We have made experiments with different settings - $\rho = 0, \rho = 0.02, \rho = 0.03$, and in any case, ICA units would learn to code for locations, although the more random the movement, the more difficult the place cell learning. This section shows results considering the most random behavior, that is, $\rho = 0.03$, which practically means that there is a probability of circa 60% for inverting the direction of movement while the robot is navigating inside one of the rooms.

For model optimization, grid search experiments are performed over a subset of the model parameters as in the previous section, using the place cell reconstruction method from Section 2.5 for position estimation.

4.2.1. Settings

Navigation in environment E2 resulted in $N_s = 192.000$ samples of sensor measurements, which means approximately 11 hours of robot navigation. The number of inputs is $n_{\text{u}} = 8$ corresponding to eight distance sensors. Using the place cell reconstruction method to estimate the robot position from the ICA layer activation, we found the following optimal parameters: $n_{\text{res}} = 600$ neurons in the reservoir; SFA and ICA layers with $n_{\text{SFA}} = n_{\text{ICA}} = 128$ units; reservoir's leak rate $\alpha = 0.1$. The nonlinearity $g(u) = u \exp(-u^2/2)$ is used for the fixed-point ICA algorithm from Section 2.3.

Each experiment during the optimization process is executed 10 times, with each run considering a randomly generated reservoir.

The input weight matrix $\mathbf{W}_{\text{in}}$ is initialized to -2, 2 and 0 with probabilities 0.15, 0.15 and 0.7, respectively (which means an input scaling of 2). The RC-SFA architecture is trained in steps

Table 2: Results using the place cell reconstruction method

Environment (Robot)	Dimensions	Type	Architecture	Test Error
E1 (SINAR)	800x600 d.u.	Simulation	RC-SFA	17.2 d.u.
E1 with dynamic obstacles (SINAR)	800x600 d.u.	Simulation	RC-SFA	108 d.u.
E2 (E-puck)	120x90 cm	Real	RC-SFA	11 cm
E2 (E-puck)	120x90 cm	Real	SFA	23 cm

as already stated, and uses 9/10 of the input signal (172.800 timesteps) as the training dataset and 1/10 (19.200 timesteps) is used for testing. After training, the ICA units are ordered by kurtosis

$$\text{kurt}(y) = E\{y^4\} - 3(E\{y^2\})^2 \quad (14)$$

such that the first unit has the most kurtosis. The above expression simplifies to $E\{y^4\} - 3$ once we assumed y is of unit variance.

4.2.2. Results

This section shows results after training the RC-SFA architecture. The mean activation of 4 SFA units, rescaled to the interval $[0, 1]$, as a function of the robot position is shown in Fig. 9(a). The slowest SFA feature shows a high response in room 1 which gradually decreases as it gets further to room 3. The third slowest feature has a low response in the middle room and a high response otherwise. Faster-varying features, like units 80 and 128, show high responses in multiple locations of the environment, characterizing low place selectivity in a way similar to entorhinal cortex cells of rats in Frank et al. (2000).

The mean activation of ICA units as a function of the robot position $h_i(\mathbf{x_r})$ is computed by averaging out the response of each ICA unit over a discrete grid of evenly spaced robot positions. Four units' mean activation are shown in Fig. 9(b). It is clear that these units learned to code for particular locations in the environment, i.e., the place fields of the cell, presenting a peak response at the center of these locations.

The mean activation does not show whether the unit is invariant to the robot movement direction. To investigate about the directionality aspect of ICA units, Fig. 10(a) shows several plots, where each row corresponds to an ICA unit, and each column considers robot positions with specific robot headings. Each plot displays robot positions associated with a heading θ in cyan color, whereas the robot positions plotted in maroon color represent a strong activation of the corresponding ICA unit. For example, unit 5, in the first row, is strongly activated at the right part of the corridor when the robot is heading right ($\theta = 0 \pm \kappa$). The last column of this figure shows the mean activation as a function of the robot heading, clearly showing the direction dependence of these ICA units. Fig. 10(b) is another plot which indicates the directionality dependence of ICA units, by showing the mean activations of each ICA unit as a function of the robot heading, where the most representative (with most kurtosis) ICA units are direction-dependent.

By using the probabilistic method described in Section 2.5, we can evaluate the capability of trained ICA units in terms of robot localization performance. Fig. 11 shows the estimated robot position using equation (13) as well as the true robot position for 3.000 timesteps. The test error, given by the Euclidean distance between $\mathbf{x_r}$ and $\hat{\mathbf{x}}_\mathbf{r}$ was 0.1188 for these 3.000 timesteps. It can be seen in the figure that the estimated position matches very well with the true robot position, confirming the good localization capability which emerged from the unsupervised learning of the RC-SFA architecture. Furthermore, we can see erratic jumps of the estimated robot position in this figure, which is actually also observed in the estimated position from the activity recorded from hippocampal place cells of rats (Zhang et al., 1998).

4.2.3. Robustness to Noise

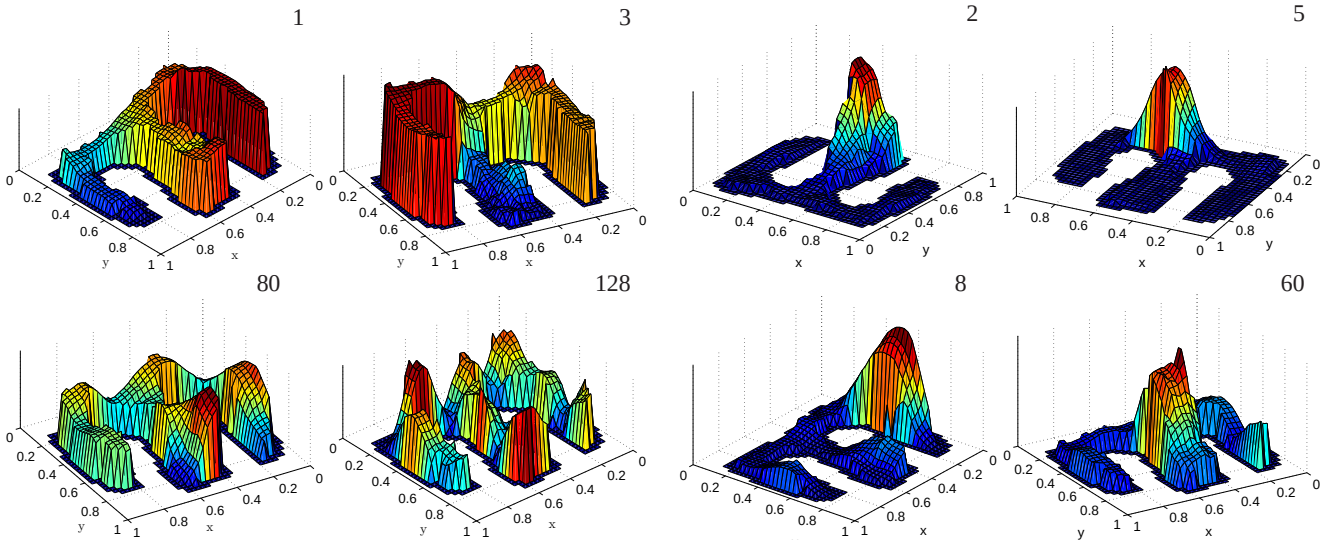
We also tested the robustness of the proposed architecture to different levels of Gaussian noise on sensor measurements. In Fig. 12, the mean and standard deviation of the test error, given by the Euclidean distance between true and predicted positions, are displayed for noise levels ranging from 1% to 50%. The error stays very low even with 10% (up to 15%) noise on sensors. From 20% on, sensors become too noisy and do not convey useful information, which causes the error to be maximum.

4.2.4. The Role of the Reservoir Recurrent Architecture

The dynamics of the reservoir is best fine-tuned by grid searching two parameters: the input scaling of $\mathbf{W}_{\text{in}}$ and the spectral radius $|\lambda_{max}|$ (Verstraeten and Schrauwen, 2009) (see Section 2.1). Fig. 13 shows that the nonlinear reservoir performs better for $|\lambda_{max}| <= 1$, on average. The reservoir's dynamic nonlinear regime is further tuned by choosing an optimal input scaling. Higher values of the input scaling yield an improve in performance, as shown in the figure. The optimal combination is an input scaling of 2.5 and $|\lambda_{max}| = 0.9$.

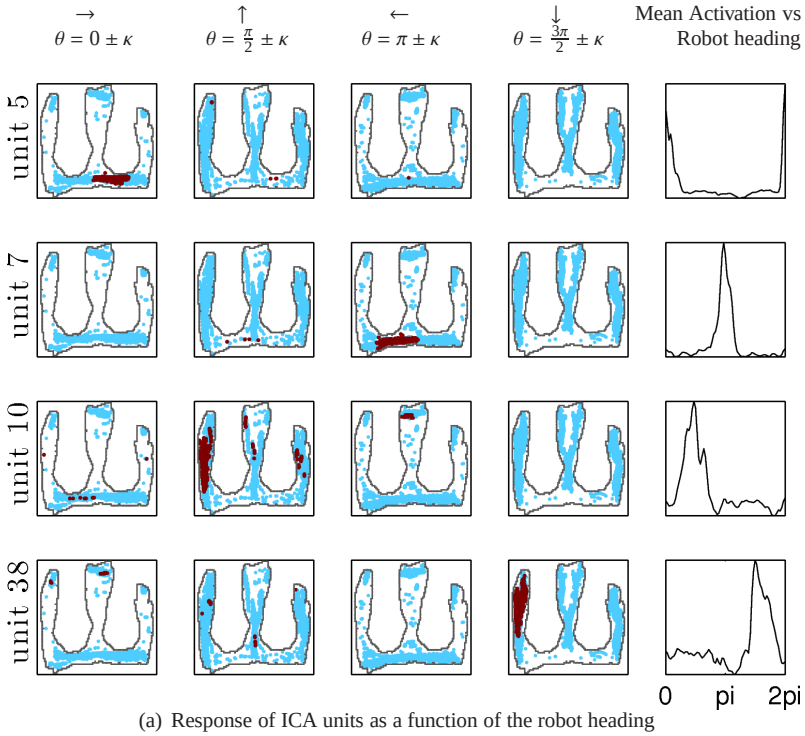
Leaky integrator neurons can also enhance performance if the input timescale does not optimally match the reservoir timescale. The leak rate α in (1) controls how fast (or slow) reservoir units respond to input stimuli. In Fig. 14, it is possible to see that there is an optimum for the leak rate, when it is approximately 0.07. It also shows that the short-term memory in the reservoir is an important characteristic for the learning of the SFA and ICA layers and, therefore, for the performance of the localization capability of the robot.

In order to confirm the importance of the reservoir in our RC-SFA architecture, experiments are performed with an archi-

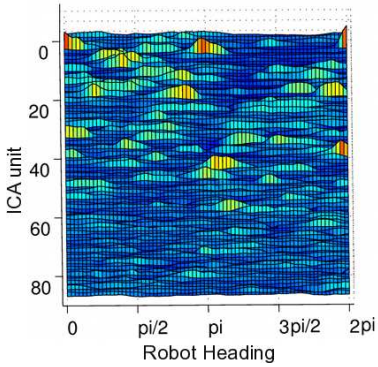


(a) 4 SFA units (b) 4 ICA units

Figure 9: Activation of SFA and ICA units in environment E2. Red denotes a high response whereas blue denotes a low response. (a) Mean activation of SFA units as a function of the robot position in the environment, rescaled to the interval [0,1]. (b) Mean activation of place cells (ICA units) as a function of the robot position in the environment.

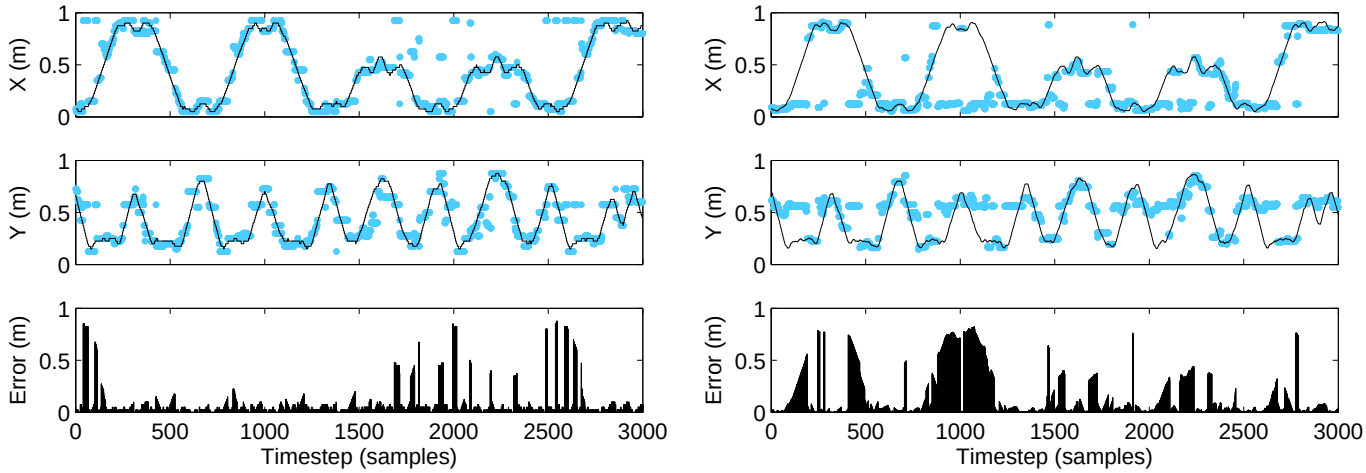


(a) Response of ICA units as a function of the robot heading



(b) Mean activation of ICA units as a function of the robot heading

Figure 10: Results after training with the e-puck robot in environment E2. (a) Directionality dependence of place cell activation for test data. Each row represents a place cell, where points in cyan (lighter) color denote the positions occupied by the robot for given directions θ in the environment and points in maroon (darker) color represent the positions where the place cell responses are higher than a certain fixed threshold. The last column shows the mean activation of a place cell as a function of the robot heading. (b) Mean activation of place cells as a function of the robot heading. The plot shows that the activation of most place cells are dependent on the robot heading. The results are shown for test data. Red denotes a high response whereas blue denotes a low response.



(a) RC-SFA + ICA

(b) SFA + ICA (using a time window and non-linear expansion)

Figure 11: Predicted robot position in environment E2 on test data using the Bayesian place cell reconstruction method for 3.000 timesteps of navigation. (a) Results using the RC-SFA architecture. The true and predicted robot coordinates are given by black curves and points in cyan color (gray for black-and-white prints), respectively. The bottom plot shows the error as the Euclidean distance between true and predicted position. (b) Results using an architecture without the reservoir, but with a time window and non-linear expansion on the input signal for the SFA layer.

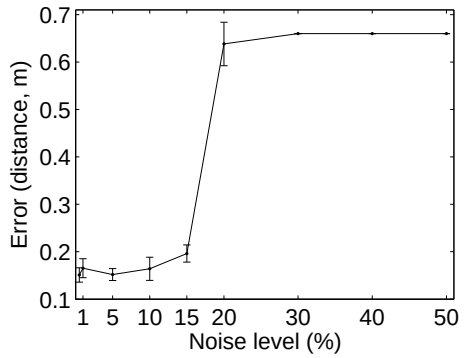


Figure 12: Robustness to Gaussian noise. The plot shows the mean and the standard deviation of the localization error on test data from environment E2 considering different noise levels on all 8 robot distance sensors. Training data uses only 0.5% noise on sensors in all experiments. Each experiment is run for 10 times (the plot shows the mean and the standard deviation).

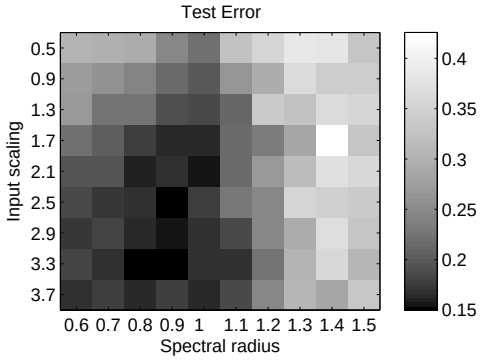


Figure 13: Input scaling vs. Spectral radius. The plot shows the mean and the standard deviation of the test localization error in environment E2 for different combinations of input scaling in $\mathbf{W}_{\text{in}}$ and the reservoir's spectral radius $|\lambda_{max}|$. Each experiment is executed 5 times with randomly generated reservoir weights. Input scaling of 2.5 and $|\lambda_{max}| = 0.9$ yields the minimum test error (black surface represents low error while white surface corresponds to higher error).

tecture which replaces the reservoir by a non-linear expansion on a time-delayed downsampled input signal. The non-linear function expands the input signal in the space of polynomials of degree 2. We optimized the model by grid searching the following parameters: downsampling rate d_t and size of the time window t_w. The best performance in terms of the Euclidean distance between true and predicted robot position, which is 0.21, is attained for $d_t = 32$ and $t_w = 2$.

So, this architecture uses a reduced and smoothed input signal by a downsampling process which converts $N_s = 192.000$ samples into $N_s = 6000$ samples. It also uses a time window of size 2 which effectively produces $n_i = 16$ inputs after the non-linear expansion step. Table 2 shows that this model has a test error which is almost double of the error using the RC-SFA architecture. In Fig. 11(b), the predicted robot position using this model and its associated error are shown. Although it learned to code for some locations in the environment, the precision is not as good as with the RC-SFA model in Fig. 11(a). Intermediate locations are not coded at all: note that the predicted X coordinate often presents big jumps.

5. Discussion

This work has proposed a biologically-inspired hierarchical architecture with three layers for learning sensor-based spatial representations of a robot environment in an unsupervised way. The proposed model does not use any idiothetic signals for path integration (or odometry) as most models do (Burgess et al., 2007; Hasselmo, 2008; Arleo et al., 2004; Stroesslin et al., 2005; Milford, 2008), and is the first to rely solely on a limited number of raw distance sensors for unsupervised learning of place cells.

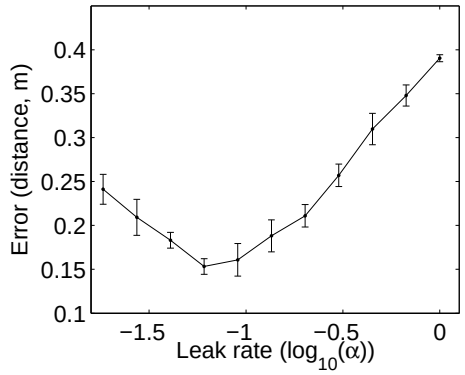


Figure 14: Influence of the reservoir's leak rate on performance. The plot shows the mean localization error (and its standard deviation) on test data from environment E2 considering different leak rates α of the reservoir. Each experiment is executed 10 times with randomly generated reservoir weights. $\alpha = 0.07$ yields the lowest test error.

The first layer of the architecture is a reservoir of recurrent nodes, which is used as a form of temporal kernel for projecting low-dimensional inputs (e.g., a small number of distance sensors) to a dynamic high-dimensional space. It integrates the noisy distance sensory input for making it possible to infer the robot position from the history-dependent trajectory of the dynamic reservoir. The second layer learns in an unsupervised way to derive slowly-varying features from the reservoir states and also possibly from the input layer, using the Slow Feature Analysis (SFA) algorithm. These slow features indicate latent signals present in the input signal which vary in a slower timescale, such as the position or the orientation of a robot in its environment. If the position can be inferred from given inputs (e.g., the reservoir state), SFA can extract it based on the slowness concept. The top layer produces independent components which are a linear combination of the SFA features, using Independent Component Analysis (ICA). It learns a sparse coding on the SFA outputs, resulting in units which are activated only for a specific position in the robot environment.

Using a probabilistic place cell reconstruction method (Zhang et al., 1998), the robot position (coordinates in the world's frame) is estimated from the activation in the ICA layer. This estimated position has shown that the ICA layer in the RC-SFA architecture has an activation correlated to the robot position, confirming the powerful capability for sensor-based unsupervised learning of spatial representations. These results are obtained in simulated environments considering a robot model with 17 distance sensors, as well as in real environments using the e-puck robot with 8 infra-red distance sensors.

The SFA layer in the proposed model has shown low place selectivity, with similarities to entorhinal cortex (EC) cells of rats in W tracks (Frank et al., 2000), whereas the ICA layer has presented high place selectivity and an activation which is dependent on the robot path, similarly to hippocampal cells of rats in Frank et al. (2000).

The directionality aspect of place cells in the ICA layer is in accordance with other works in the literature which show that environment shape and robot behavior affect the directionality component of place cells (Frank et al., 2000; Eichenbaum et al., 1999; Brunel and Trullier, 1998). Most of these computational models implement path integration using idiothetic input, despite the current lack of knowledge with respect to the mechanisms of path integration in the brain like the integration of self-motion signals with allothetic input (Eichenbaum et al., 1999). On the other hand, in the proposed RC-SFA architecture, the reservoir integrates the allothetic input (distance sensors), forming a trajectory in state space which SFA units use to learn spatial features from a given environment. This computation can be compared with optical flow, in the sense that the reservoir provides a temporal memory of the stimuli stream which can be used for distance estimation, that is, the reservoir is involved in maintaining an estimate of the robot location for a temporary time period in the absence of the distance sensory input.

Learning in SFA is comparable to Principal Component Analysis (PCA) in terms of complexity. Furthermore, while biologically plausible implementations of SFA exist (Hashimoto, 2003), there is experimental evidence showing that the slowness learning principle of SFA is present in the visual cortex (Li and DiCarlo, 2008).

Although the current ICA implementation may seem biologically unrealistic, a more biologically plausible learning scheme for generating place cells at the top ICA layer from the non-localized representation of SFA units can be implemented by competitive learning (Franzius et al., 2007b) or non-linear Hebbian learning (Hyvärinen and Oja, 1998).

5.1. Related works

It has been shown in Franzius et al. (2007a) that a hierarchy of SFA layers with increasing receptive fields at upper layers and a top ICA layer can be trained to code for either the rat's position or the rat's head direction depending on the movement pattern of a simulated rat. Their model is based on the high-dimensional input from a camera which simulates the 320^o field of view of the rat. Simple environments such as linear tracks or rectangular arenas with distinct textures set for each wall make it possible to infer the rat's position from a single image. The similarities between their model and the one proposed in this paper refer to the layers which learn by SFA and ICA. The main differences are that we use a dynamic reservoir at the first layer, which projects a low-dimensional input into a high-dimensional non-linear space and which proved to be essential for learning spatial representations with such a small number of distance sensors. Moreover, our model copes with sensor aliasing, where multiple environmental states map to the same perceptual sensory input.

This means that it is not sufficient to consider only the current time step to determine the robot location, but the history of the input stream - a property which the reservoir naturally has.

In Wyss et al. (2006), a cortical hierarchy of layers is proposed which learn by optimizing an objective function that takes into account temporal stability and temporal decorrelation between units. All units are leaky integrators providing them with a local memory trace.

Their method is similar to Slow Feature Analysis in the sense that it maximizes temporal stability or slowness of the output signal. However, their learning method is iterative and, as so, prone to convergence to local optimum (unlike SFA). Their experiments are made with a mobile robot driving randomly in a rectangular environment with predefined cues. The continuous stream of a 16x16 pixels image feeds the architecture which, after learning, shows properties of hippocampal place cells.

The SFA algorithm has also been used as the training method of a linear readout output layer of reservoirs of spiking neurons in Klampfl and Maass (2009). They show that these linear readouts can learn to discriminate isolated spoken digits in an unsupervised way.

Other models of hippocampal place cells and biologically-inspired navigation exist in the literature. In Arleo et al. (2004), unsupervised growing networks are used to build an architecture with idiothetic and allothetic components that are combined in a hippocampal place cell layer to support spatial navigation (validated using a Khepera mobile robot with 2D vision sensors). Their model explicitly uses dead-reckoning to track the robot position and associates place cell firing with the estimated position.

In Milford (2008), a hippocampal place cell model is designed to solve the SLAM problem. They choose a pragmatic approach, favoring functionality over biologically plausibility. Their model, called RatSLAM, have a 3D structure for pose cells (representing beliefs for the robot position and orientation) which learn associative connections with view cells (allothetic representation). They validate their model with several mobile robots, equipped with a camera, in indoor and outdoor environments. For a further review on biologically-inspired localization models, see Filliat and Meyer (2003) and Trullier et al. (1997).

5.2. Future Work

Future research could be done on generative models which predict the robot perceptual input given the activation of the ICA layer. This can be implemented in a supervised fashion using a reservoir with the location as inputs (ICA units) and the sensory input as desired output (see Antonelo et al., 2007, for an example). Although the training of this generative model is supervised, the whole learning process is still unsupervised, because no labels for the locations are needed. By predicting the sensory input, the localization in the ICA layer may be improved in situations where sensors are broken or faulty (Antonelo et al., 2007). In the context of robot navigation and localization, future work include the integration of the allothetic representation of the RC-SFA architecture with the robot position estimated by a path integration module (odometry). With this setup, the new architecture could be compared to a basic form of biologically-inspired SLAM (as in Milford, 2008).

In principle, the proposed RC-SFA architecture would scale to larger environments, but likely not to very random robot behaviors. On the other hand, a hierarchical SFA architecture could be used for scaling to larger environments and richer sensory data (e.g., a camera) (Franzius et al., 2007a).

Acknowledgment

The authors would like to thank the anonymous referees for their contributions to the improvement of this article and to Michiel D'Haene for helping on issues with the e-puck robot. This research is partially funded by the EC FP7 project ORGANIC (FP7-231267). Eric Antonelo is sponsored by the Special Research Fund of Universiteit Gent (BOF).

References

Antonelo, E. A., Baerlvedt, A.-J., Rognvaldsson, T., Figueiredo, M., 2006. Modular neural network and classical reinforcement learning for autonomous robot navigation: Inhibiting undesirable behaviors. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN). Vancouver, Canada, pp. 498– 505.

Antonelo, E. A., Schrauwen, B., 2009. Towards autonomous self-localization of small mobile robots using reservoir computing and slow feature analysis. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 3818–3823.

Antonelo, E. A., Schrauwen, B., Campenhout, J. V., 2007. Generative modeling of autonomous robots and their environments using reservoir computing. Neural Processing Letters 26 (3), 233–249.

Antonelo, E. A., Schrauwen, B., Stroobandt, D., 2008. Event detection and localization for small mobile robots using reservoir computing. Neural Networks 21, 862–871.

Antonelo, E. A., Schrauwen, B., Stroobandt, D., 2009. Unsupervised learning in reservoir computing: Modeling hippocampal place cells for small mobile robots. In: ICANN '09: Proceedings of the 19th International Conference on Artificial Neural Networks. Vol. 5768. Springer-Verlag, Berlin, Heidelberg, pp. 747–756.

Arleo, A., Smeraldi, F., Gerstner, W., May 2004. Cognitive navigation based on nonuniform gabor space sampling, unsupervised growing networks, and reinforcement learning. IEEE Transactions on Neural Networks 15 (3), 639–652.

Berkes, P., Wiskott, L., 2005. Slow feature analysis yields a rich repertoire of complex cell properties. Journal of Vision 5, 579–602.

Brunel, N., Trullier, O., 1998. Plasticity of directional place fields in a model of rodent ca3. Hippocampus 8, 651–665.

Buonomano, D., Maass, W., 2009. State-dependent computations: Spatiotemporal processing in cortical networks. Nature Reviews Neuroscience 10 (2), 113–125.

Burgess, N., Barry, C., O'Keefe, J., 2007. An oscillatory interference model of grid cell firing. Hippocampus 17 (9), 801–812.

Eichenbaum, H., Dudchenko, P., Wood, E., Shapiro, M., H, H. T., 1999. The hippocampus, memory, review and place cells: Is it spatial memory or a memory space? Neuron 23, 209–226.

Filliat, D., Meyer, J.-A., 2003. Map-based navigation in mobile robots:: I. a review of localization strategies. Cognitive Systems Research 4 (4), 243 – 282.

Frank, L. M., Brown, E. N., Wilson, M., 2000. Trajectory encoding in the hippocampus and entorhinal cortex. Neuron 27 (1), 169 – 178.

Franzius, M., Sprekeler, H., Wiskott, L., August 2007a. Slowness and sparseness lead to place, head-direction, and spatial-view cells. PLoS Computational Biology 3 (8), 1605–1622.

Franzius, M., Vollgraf, R., Wiskott, L., 2007b. From grids to places. Journal of Computational Neuroscience 22, 297–299, 10.1007/s10827-006-0013-7.

Hashimoto, W., 2003. Quadratic forms in natural images. Network: Comput. Neural Syst. 14, 765–788.

Hasselmo, M. E., 2008. Grid cell mechanisms and function: Contributions of entorhinal persistent spiking and phase resetting. Hippocampus 18 (12), 1213–1229.

Hyvärinen, A., Oja, E., 1998. Independent component analysis by general nonlinear hebbian-like learning rules. Signal Processing 64 (3), 301 – 313.

Hyvärinen, A., Oja, E., 2000. Independent component analysis: algorithms and applications. Neural Networks 13, 411–430.

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

Jaeger, H., Haas, H., April 2 2004. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. Science 308, 78–80.

Jaeger, H., Lukosevicius, M., Popovici, D., 2007. Optimization and applications of echo state networks with leaky integrator neurons. Neural Networks 20, 335–352.

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

Klampfl, S., Maass, W., 2009. Replacing supervised classification learning by slow feature analysis in spiking neural networks. In: Proc. of NIPS 2009, Advances in Neural Information Processing Systems. Vol. 22. MIT Press, pp. 988–996.

Li, N., DiCarlo, J. J., 2008. Unsupervised natural experience rapidly alters invariant object representation in visual cortex. Science 321 (5895), 1502–1507.

Maass, W., Natschläger, T., Markram, H., 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14 (11), 2531–2560.

Milford, M., 2008. Robot Navigation from Nature. Springer Tracts in Advanced Robotics.

Mondada, F., September 2007. E-puck education robot. Http://www.e-puck.org/.

Moser, E. I., Kropff, E., Moser, M.-B., 2008. Place cells, grid cells and the brain's spatial representation systems. Annual Reviews of Neuroscience 31, 69–89.

O'Keefe, J., 1976. Place units in the hippocampus of the freely moving rat. Experimental Neurology 51 (1), 78 – 109.

O'Keefe, J., Dostrovsky, J., 1971. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. Brain Research 34, 171–175.

Schrauwen, B., Busing, L., Legenstein, R., 2008. On Computational Power and the Order-Chaos Phase Transition in Reservoir Computing. In: Proceedings of NIPS.

Siegwart, R., Nourbakhsh, I. R., 2004. Introduction to Autonomous Mobile Robots. Bradford Book.

Solomon, J. H., Hartmann, M. J., 2006. Sensing features with robotic whiskers. Nature 443, 525.

Stroesslin, T., Sheynikhovich, D., Chavarriaga, R., Gerstner, W., 2005. Robust self-localisation and navigation based on hippocampal place cells. Neural Networks 18 (9), 1125–1140.

Thrun, S., Burgard, W., Fox, D., 2005. Probabilistic Robotics. The MIT Press.

Trullier, O., Wiener, S. I., Berthoz, A., Meyer, J.-A., April 1997. Biologically-based artificial navigation systems: Review and prospects. Progress in Neurobiology 51 (5), 483–544.

Verstraeten, D., Schrauwen, B., 2009. On the quantification of dynamics in reservoir computing. In: ICANN '09: Proceedings of the 19th International Conference on Artificial Neural Networks. Vol. 5768. Springer-Verlag, pp. 985–994.

Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D., 2007. A unifying comparison of reservoir computing methods. Neural Networks 20, 391–403.

Wiskott, L., Sejnowski, T. J., 2002. Slow feature analysis: Unsupervised learning of invariances. Neural Computation 14 (4), 715–770.

Wyffels, F., Schrauwen, B., Verstraeten, D., Stroobandt, D., 2008. Band-pass reservoir computing. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN). pp. 3204–3209.

Wyss, R., Knig, P., Verschure, P. F. M. J., 2006. A model of the ventral visual system based on temporal stability and local memory. PLoS Biol 4 (5), e120.

Yamazaki, T., Tanaka, S., 2007. The cerebellum as a liquid state machine. Neural Networks 20, 290–297.

Zhang, K., Ginzburg, I., McNaughton, B. L., Sejnowski, T. J., Feb 1998. Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. J Neurophysiol 79 (2), 1017–1044.