Discovering the Phase of a Dynamical System from a Stream of Partial Observations with a Multi-map Self-organizing Architecture

Bassem Khouzam, Hervé Frezza-Buet
Information, Multimodality and Signal
Supélec, UMI 2958 Georgia Tech/CNRS, Metz, France
{*Bassem.Khouzam,Herve.Frezza-Buet*}*@supelec.fr*

***Abstract*—This paper presents a self-organizing architecture made of several maps, implementing a recurrent neural network to cope with partial observations of the phase of some dynamical system. The purpose of self-organization is to set up a distributed representation of the actual phase, although the observations received from the system are ambiguous (i.e. the same observation may correspond to distinct phases). The setting up of such a representation is illustrated by experiments, and then the paper concludes on extensions toward adaptive state representations for partially observable Markovian decision processes.**

***Keywords*-Dynamical Systems; Recurrent Neural Networks; Self-Organization.**

I. INTRODUCTION

In the design of artificial agents evolving in some environment, one has to deal with information *streams*. Sensors provide input streams to the information processing system of the agent, and actuators actually produce a stream of actions performed in order to exploit the environment. Such an action stream is actually the output of the agent to the environment. The coupling of the agent and the environment through such information and energy streams is obvious for any biologist who analyzes the behavior of some animal. Nevertheless, in the field of Computer Science and Machine Learning, computation is often considered off-line, for technical reasons. The typical case is the use of data sets to train the models, before actually using the trained models on-line.

This paper is a contribution to the part of Computer Science that is rather involved in the design of situated systems, thus actually focusing on the process of *streams* of information. In that sense, it is related to reinforcement learning approaches, that deal with sequential decision making of an agent continuously interacting with its environment, as well as temporal systems like recurrent neural networks that handle sequences of input. Indeed, the model proposed here allows to extract the phase of some dynamical system from a sequence of observations computed from that phase. Let us illustrate the need for such a feature from a straightforward toy example.

Let us consider an animal perceiving the temperature T of the floor. Let us suppose that any temperature $T > T_0$ is dangerous to it. In our example, the temperature oscillates periodically between high and low values (for night and day). The whole solar system configuration is the *phase* of the environment, noted x^t here. It evolves in a deterministic way, according to Newton's law. The phase evolution is thus driven by a transition function ϕ such as $\forall t, x^{t+\mathrm{d}t} = \phi(x^t)$. The temperature perceived by the animal is an *observation* of the solar system, that can be expressed as $T^t = O(x^t)$ where O is the observation function, see figure 1. The sun position in the sky $P^t = O'(x^t)$ would have been another observation of the solar system phase. Let us now consider a time t for which the temperature $T^t = T_0 - \epsilon$ is just below the threshold. Should the animal try to hide away from the sun? The answer depends on the phase x^t, from which the animal could know if the temperature is currently decreasing or increasing. The decision would have been easier from the perception of sun position P^t, since O' is a bijective function, and thus the values of P^t allow to take the decision directly from the current perception. If only T^t is perceived by the animal, an efficient behavior requires that the animal is able to represent internally a value $\hat{x}^t$ from which it can take the right decision, since values of temperature may be ambiguous (similar temperatures are observed twice a day). T is thus said to be a partial observation of x. The current value $\hat{x}^t$ is inferred and updated from the successive observations T^t. It is not required that $\hat{x}$ be the exact representation of the phase x, i.e. the animal do not need to know where the planets are, but $\hat{x}$ has to be set up such that a bijective observation function implicitly exists from x to $\hat{x}$.

Partially observed environments are of interest in reinforcement learning domain. While the general trend is to find $\hat{x}^t$ before computing the corresponding value function of each state, other works like [1] implement evaluation with a recurrent network, but without explicitly extracting some $\hat{x}^t$.

The neural architecture presented in this paper relies on self-organizing neural networks in order to build such an internal representation from ambiguous sequences of observations. Many works in the literature try directly or indirectly to find $\hat{x}^t$. Concerning the use of self-organizing maps, many enhancements on Kohonen basic map [2] like in [3], [4], [5] consider the temporal dimension of input sequences but they deal with the recognition of manually

 ISBN: 978-1-61208-155-7

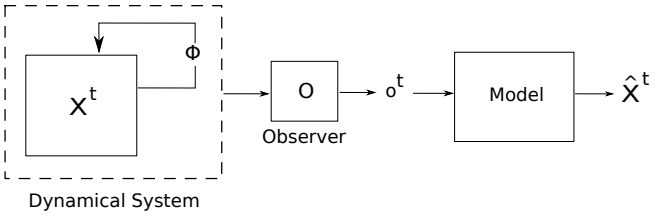


Figure 1. Dynamical system phase extraction.

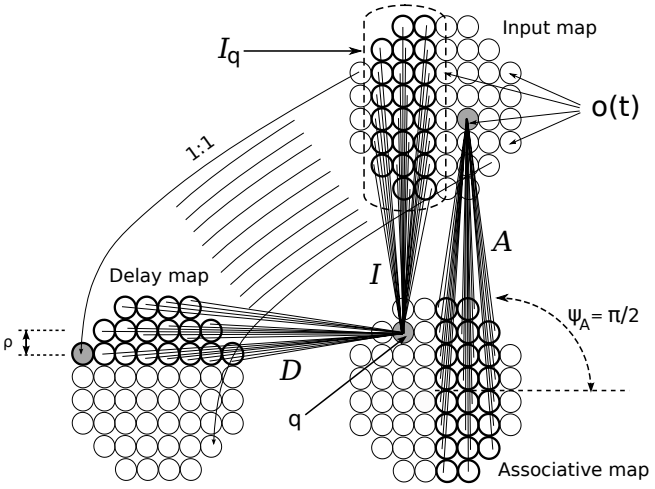


Figure 2. Model architecture. Input map and delay map are connected via one-to-one connections, other connections are one-to-many connections organized in strips. For unit q, $\mathcal{I}_q$ is the strip of kind $\mathcal{I}$ handled by q.

extracted sequences, rather than on-line stream of inputs.

Reservoir computing methods [6] are closer to that purpose, since they handle inputs from a stream one by one. Readout units then search the huge reservoir states in order to locate few significant states. The significant states represents the phases of the system that provide the inputs.

For the best of our knowledge, the present work is the first attempt to find a mapping of a dynamical system phase space using on-line self-organizing recurrent neural networks.

II. The model

As mentioned above, our goal is to design a system that generates an internal representation of the dynamical system phases from the stream of observations emerging from it. Our model is an effort in this direction, inspiring from biological information. It is based on the `bijama` model [7], proposed and developed in our team. It enables building computational cortical-like 2D-neural assemblies called *maps*, made of computational units representing cortical columns. Units allow to process in parallel external entry and internal signals carried out by connections. A schematic of the model is shown in Fig.s 2 and 3. The architecture comprises three maps, namely the input map, the delay map, and the associative map. The input map receives the external input stream. Its activity is expected to represent at time t the coding $\hat{x}^t$ of the actual dynamical system phase x^t.

The two other maps, the delay map and the associative map are auxiliary maps that play the role of intermediate structures for extracting $\hat{x}^t$ from the input stream o^t. Their purpose is to re-inject the delayed activity of the input map into its dynamics. This recurrent pathway actually reveals the temporal dimension of the input stream. Map activities are computed by a neural field. Each unit has lateral recurrent connections to other units within the map, implementing an on-center/off-surround connectivity [8], [9]. The field performs lateral competition between units and computes the activity of each one, so that the global map activity has the shape of a bump (see dark meshes in Fig. 4). The bump positions are actually the response of the map to its input. Lateral competition, from which activity bumps emerge, is used in the model in order to guide the process of self-organization of inputs over the map surface. This is indeed difficult with neural fields as explained in [9] from which the neural field process used in this paper is taken. The whole architecture evolution is controlled by successive *time steps*. A time step is a discrete time instance at which the activities of all units in all maps are evaluated once, using an asynchronous evaluation scheme [9]. Another kind of connectivity in the model is the inter-map connectivity. A unit at the bi-dimensional position p in some local map can be connected to a whole strip-shaped region $\mathcal{S}_p$ in some remote map. Then unit p handles connections from the units at positions $q \in \mathcal{S}_p$ in the remote map, as shown in Fig. 2. Each connection in a strip between p and a remote unit q handles a weight whose current value is $\bar{s}^t_{pq}$, so that the strip $\mathcal{S}_p$ owned by p handles a vector of weights $\bar{S}^t_p = \left(\bar{s}^t_{pq}\right)_{q \in \mathcal{S}_p}$. Let us note $\mathcal{S}$ the set of strips received by local map ($\mathcal{S} = \mathcal{A}, \mathcal{I}, \mathcal{D}$ in Fig. 2). Inter-map connections are referred to as *cortical connections*. Strips are characterized by their width $\rho_{\mathcal{S}}$ and direction $\psi_{\mathcal{S}}$ relative to the horizontal axis connecting the centers of the local and remote map. Let us note the activity of the unit at position p at time t as u^t_p, and the vector of remote unit activities perceived at p through the strip $\mathcal{S}_p$ as $S^t_p = \left(u^t_q\right)_{q \in \mathcal{S}_p}$. The computation of unit activity using `bijama` is achieved using a stack of modules (see Fig. 3). Each one handles a scalar value that may be received as input or computed from lower modules in the stack. The higher module (here, the neural field module) handles the unit output that is the one actually accessed through cortical connections.

Let us first describe the stack of modules used for the units in the input map, see Fig. 3 and 4 while reading the definitions which follow. In general terms, the input map, in the one hand, receives external observations o^t. In the other hand it also receives strips (noted $\mathcal{A}$) from the associative map. It outputs an activity bump as a response. This activity represents $\hat{x}^t$ as will be shown later. The lower module is

referred to as the *thalamic module*. It handles the external input o^t, and matches it against a stored prototype ω_p^t and computes the similarity θ_p^t.

$$\theta_p^t = e^{-\frac{\left(o^t - \omega_p^t\right)^2}{2\sigma^2}} \quad (1)$$

The second module is referred to as the *cortical module*. It handles the strip $\mathcal{A}_p$ emerging from the associative map, and computes the matching $c_{p,\mathcal{A}}^t$ between the strip weight vector $\bar{A}_p^t$ and the activity vector A_p^t. The matching is computed as follows, where B is a numerical constant:

$$c_{p,\mathcal{A}}^t = \frac{\left\langle A_p^t . \bar{A}_p^t \right\rangle}{\max\left(\left\|\bar{A}_p^t\right\|^2, B\right)} \quad (2)$$

The third module is referred to as *cortico-thalamaic merging*. It merges θ_p^t and $c_{p,\mathcal{A}}^t$ into one scalar ν_p^t as follows:

$$\nu_p^t = \sqrt{\theta_p^t . \beta + (1-\beta) . c_{p,\mathcal{A}}^t} \quad (3)$$

Where β is a constant. The value of ν_p^t forms the final input ready to use by the neural field, i.e. the upper module, to compute the unit activity u_p^t.

The unit activity is used to modulate the learning. Thalamic learning implies moving θ_p^t towards o^t proportionally to u_p^t, as shown in (4).

$$\omega_p^{t+1} = \omega_p^t + \alpha_\omega . u_p^t . (o^t - \omega_p^t) \quad (4)$$

Where α_ω is a fixed thalamic learning rate for all units.

On the other hand, cortical learning implies moving the weight $\bar{a}_{pq}^t$ of each connection included in the strip $\mathcal{S}_p$ towards the cortical input u_q^t, as shown by 5.

$$\bar{a}_{pq}^{t+1} = \bar{a}_{pq}^t + \alpha_\mathcal{S} . u_p^t . (u_q^t - \bar{a}_{pq}^t) \quad (5)$$

Where $\alpha_\mathcal{S}$ is a fixed cortical learning rate for all model connections. The previous rule means that learning occurs only in connections to active units in local maps.

The next map in the model is the associative map. Its receives the actual activity of the input map as well as its delayed activity exhibited by the delayed map, then it performs lateral competition via the neural field, and re-injects the result into the input map through the previously mentioned $\mathcal{A}$ strips. The first module of a unit q of this map handles the strip $\mathcal{I}_q$ emerging from input map and computes the matching $c_{q,\mathcal{I}}^t$. The second module handles the strip $\mathcal{D}_q$ emerging from delay map and computes the matching $c_{q,\mathcal{D}}^t$. Both matching values are computed similarly to what was shown in input map units, according to 2. The third module is referred to as *cortico-cortical merging*. It merges $c_{q,\mathcal{I}}^t$ and $c_{q,\mathcal{D}}^t$ in one scalar μ_q^t as follows:

$$\mu_q^t = \sqrt{c_{q,\mathcal{I}}^t . c_{q,\mathcal{I}}^t} + \text{noise}_\mu \quad (6)$$

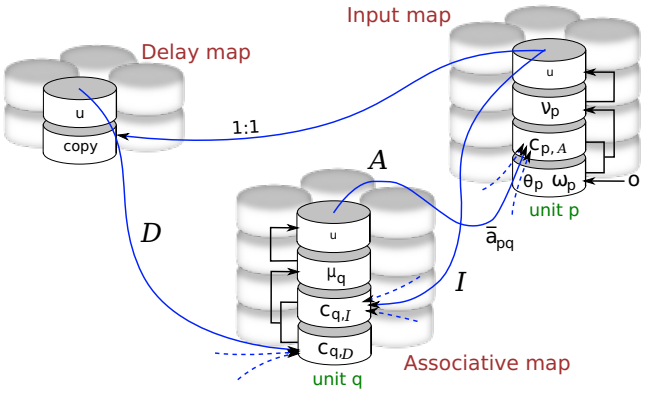


Figure 3. Module stacks of the units in the three maps, and their cortical inter-connections.

The purpose of noise is to boost the associative map activity in units receiving null cortical activity before being injected into the input map. The value of μ_q^t is actually the input to the associative map neural field module which computes the unit activity u_q^t.

The last map in the model is the delay map. It receives a unit-to-unit copy of the input map activity and delays it for some period of T time steps, using a T-length FIFO queue. Units in this maps have two modules. The first is the *copy* module that copies u_p^t from the input map. The second is the *FIFO* module. Thus $u_p^t = u_q^{t-T}$ where q is a position in the input map and p the same position in the delay map. There is no need for a neural field in this map.

As can be seen, the proposed architecture requires no prior conditions on the input stream, nor on the underlying dynamical system, it is thus a model free architecture. The model does not require to adjust any parameter during execution. Learning rates are thus constant. Moreover, there is no need for resetting output bump activities u when a new observation $o^{\tau+1}$ is presented.

III. EXPERIMENTS

In this section, the model is tested to validate its capability to find an internal state representation of some unknown dynamical system providing observations. A toy example of dynamical system is used here to test the capacity of spatio-temporal organization of the model. Let $x \in \mathbb{C}$ represent the system phase, and let the transition function be $\phi(x) = x.e^{i\varphi}$. Let us consider that the system transition occurs at instant τ, thus we can write $x^{\tau+1} = \phi(x^\tau)$ with $x^0 = 0.5$. Let the partial observation fed to the model be $O(x^\tau) = 0.5 + \Re(x^\tau) + \text{noise}_o$. Its values are kept in $[0, 1]$. The sampled stream values are perturbed by noise to test the robustness to noisy observations.

The dynamical system phase can be thought of as the position of a point moving in a steady speed on a circle, and the observation is its noisy abscissa. This sinusoidal observation is obviously ambiguous.

 ISBN: 978-1-61208-155-7

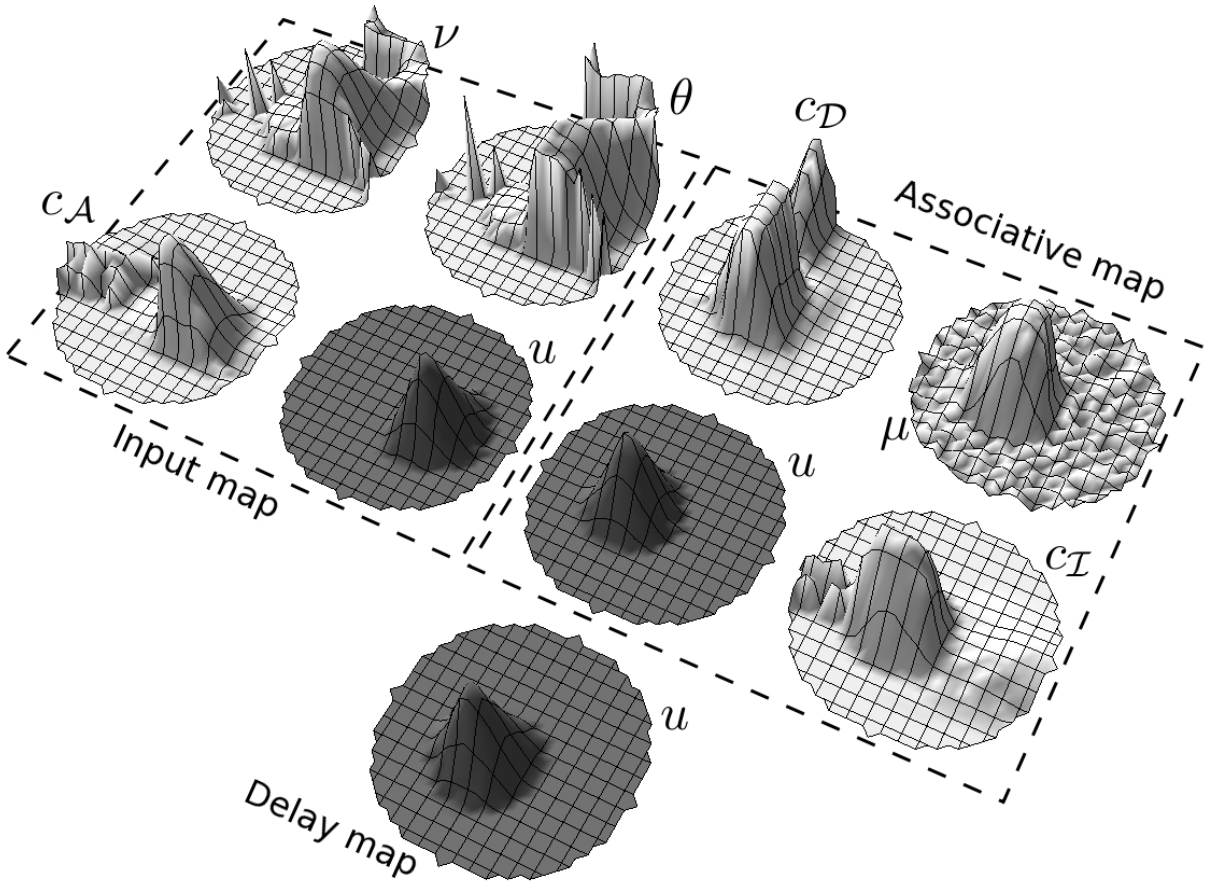


Figure 4. Activity of model maps modules.

In this experiment, τ is incremented every T time steps, and each observation $O(x^\tau)$ is fed to the model during T time steps, i.e. $o^t = o^{t+1} = \cdots o^{t+T-1} = O(x^\tau)$ and $o^{t+T} = o^{t+T+1} = \cdots o^{t+2T-1} = O\left(x^{\tau+1}\right)$, etc. The reason for that is to give enough time to the neural field to relax and form an appropriate bump, as well as for cortical and thalamic learning to influence significantly the weights. T value is the same as FIFO length used in delay map units, thus, it delays input maps activity until the next $O(x^\tau)$ is sampled.

The input stream value o^t is presented to all the units in the input map as for Kohonen maps [2]. The map response is computed as the barycenter position G^τ of the u activity bump at the end of each chunk of T successive time steps. It is computed as follows:

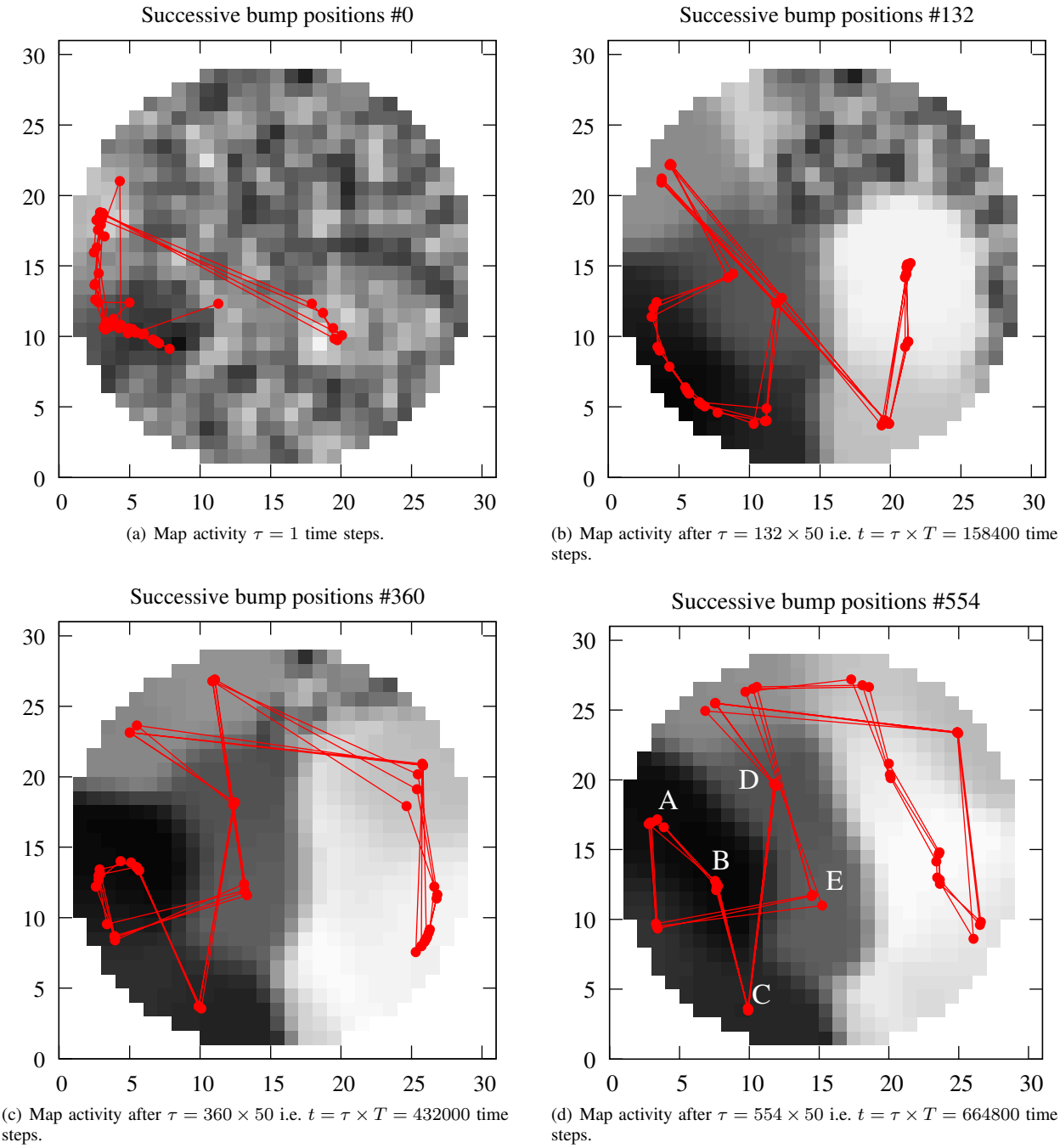
$$G^\tau = \sum_p u_p^t.p / \sum_p u_p^t : p \in [1, M]^2 \qquad (7)$$

Where M is the dimension of the square surrounding the round map. Each time that l barycenters are computed, they are organized in a list $P^\tau = \{G^{\tau-l+1}, G^{\tau-l+2}, \cdots, G^\tau\}$ of positions forming l-length paths over the map as sketched in Fig. 5. Successive paths allow to track the evolution of the map state $\hat{x}^t$ through time.

Fig. 5 shows the evolution through time of the input map. At the experiment start, thalamic values ω_p^t are random and the activity is located in a limited regions on the map as illustrates Fig. 5(a). Activity bumps start to disperse in Fig. 5(b). At this stage, due to thalamic learning guided by the neural field, thalamic values start to exhibit spatial organization as the grey-scaled regions show. The reason is that ω_p^t mainly organize according to values of o^t which are in $[0, 1]$. Fig. 5(c) shows a better organization of the thalamic values, and different regions appear, each handling some different range of the o^t values. Besides, it exhibits a better dispersion of activity bumps in the each region.

At the end of learning, Fig. 5(d) shows that different grey-scaled regions can be distinguished, indicating the spatial self-organization of thalamic values.

The poly-line in the figure is formed by $l = 50$ points, corresponding each to the representation $\hat{x}^t$ of a dynamical system phase x^t. As can be seen, points are clustered along



(a) Map activity $\tau = 1$ time steps.

(b) Map activity after $\tau = 132 \times 50$ i.e. $t = \tau \times T = 158400$ time steps.

(c) Map activity after $\tau = 360 \times 50$ i.e. $t = \tau \times T = 432000$ time steps.

(d) Map activity after $\tau = 554 \times 50$ i.e. $t = \tau \times T = 664800$ time steps.

Figure 5. Status of the input map during the system evolution. Grey-scaled values are the ω prototypes (white for 0, black for 1). P^τ is represented with a poly-line, linking successive positions $G^{\tau-l+1}, G^{\tau-l+2}, \cdots, G^\tau$, that are localized on the figures with red dots.

the poly-line. One cluster is formed by repeated visits of the same system state. This indicates the stable representations of states in the map space. Each thalamic region corresponds to a range of close observations values. Within each region there exists the representation of 2 or more states. For example, the black region corresponds to observation values close to 1, nevertheless, it contains 3 distinct successive state representations marked A,B,C. When a range of observations is located in the middle of the input values range, points (like D,E) express non-successive states corresponding to the same observation range, but in different temporal context. Such duplications of states, related to the same value of O, are progressively performed while the whole architecture gets organized. It removes observation ambiguity. Thus, the

 ISBN: 978-1-61208-155-7

ensemble of state representations $\hat{x}$ (i.e. bump positions) expresses a bijective mapping between the map surface and the dynamical system phase space. This was possible because the model allowed the previous state of the input map to be considered in computing its new state, integrating this way, its state history. The added noise to the input stream did not affect the model capability to extract the mapping.

The experience was launched with numerical values for the dynamical system as follows: $\varphi = 2\pi/15$ and noise_o is sampled from a uniform random noise $\mathcal{U}[-0.05, 0.05]$.

Model numerical values was initialized as follows: $M = 30$ for all maps, $u_p^0 = 0$, ω_p^0 and $\bar{a}_{pq}^0, \bar{i}_{pq}^0, \bar{d}_{pq}^0$ are initialized to uniform random values from $\mathcal{U}[0, 1]$, $\sigma = 0.07$, $\alpha_\omega = \alpha_\mathcal{S} = 0.0416$, $B = 10$, $\beta = 0.25$, noise_μ is sampled from a uniform random noise $\mathcal{U}[0, 0.1]$, $\rho_\mathcal{I} = \rho_\mathcal{A} = \rho_\mathcal{D} = 5$. $\psi_\mathcal{I} = 90$, $\psi_\mathcal{A} = -90$, $\psi_\mathcal{D} = 0$, $T = 24$, and $l = 50$.

IV. Conclusion and future work

In this paper, a recurrent neural architecture is proposed for setting up a representation of the phases of a dynamical system from the stream of partial observations of that dynamical system. The phase extraction relies on three self-organizing modules, whose self-organizing processes are coupled via strip-like connections, according to the `bijama` model. Experiments show that this fully unsupervised architecture is able to self-organize so that the token of hidden phases of the dynamical system are explicitly built in the input map. Indeed, for each bump position in that map, a phase of the dynamical system can be assigned. Moreover, the topology preservation that is expected from self-organizing maps actually stands here, since the input map is still a continuous mapping of the space where the observation lives (here the interval $[0, 1]$).

In the one hand, seminal works by Elman and Jordan [10], [11] have already addressed the learning of a dynamical system from the stream of observation, but this was obtained from a supervised approach. In the other hand, as mentioned, reservoir computing approaches relies on high dimensional representation spaces to build an a priori set of states, from which the ones corresponding to the actual phases of the system can be extracted by readout units. In both cases, the setting up of a phase representation is not explicit. Here, the whole architecture adapts for extracting explicit phase representation by self-organization.

Future work will investigate the potential of the model self-organization features in both space and time when applied to systems exhibiting non-stationary dynamics. The goal is to see if the model would be able to recruit new regions in the input map or release useless regions when the dynamic change. Future work consists also in using the representation built in the input map as a state space for taking decisions within Markovian decision processes. In a more integrated model, such cortical representations could indeed feed actor and critic neural modules, inspired from basal ganglia modeling [12], [13], with a Markovian state space representation that is updated from the current partial information provided by the robot sensors.

Acknowledgments

This work is part of the InterCell project supported by INRIA, Région Lorraine and Supélec.

`http://intercell.metz.supelec.fr`

References

[1] I. Zhenzhen Liu; Elhanany, "A scalable model-free recurrent neural network framework for solving pomdps," in *Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007. IEEE International Symposium on*, 2007, pp. 119–126.

[2] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Springer-Verlag, Inc., 2001.

[3] G. J. Chappell and J. G. Taylor, "The temporal kohonen map," *Neural Netw.*, vol. 6, pp. 441–445, March 1993.

[4] M. Varstal, J. Milln, and J. Heikkonen, "A recurrent self-organizing map for temporal sequence processing," in *Artificial Neural Networks ICANN'97*. Springer Berlin / Heidelberg, 1997, pp. 421–426.

[5] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, "Time series prediction using recurrent som with local linear models," *Int. J. of Knowledge-Based Intelligent Engineering Systems*, pp. 60–68, 1997.

[6] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, Aug. 2009.

[7] O. Ménard and H. Frezza-Buet, "Model of multi-modal cortical processing: Coherent learning in self-organizing modules," *Neural Networks*, vol. 18, no. 5-6, pp. 646 – 655, 2005, iJCNN 2005.

[8] S. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biol Cyb*, vol. 27, pp. 77–87, 1977.

[9] L. Alecu, H. Frezza-Buet, and F. Alexandre, "Can self-organization emerge through dynamic neural fields computation? ," *Connection Science*, vol. 23, no. 1, pp. 1–31, 2011.

[10] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[11] M. I. Jordan, "Attractor dynamics and parallelism in a connectionnist sequential machine," in *Proc. 8th Annual Conference of the Cognitive Science Society*. Erlbaum, 1986, pp. 112–127.

[12] K. Doya, "What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?" *Neural Networks*, vol. 12, no. 7-8, pp. 961–974, 1999.

[13] A. Leblois, T. Boraud, W. Meissner, H. Bergman, and D. Hansel, "Competition between feedback loops underlies normal and pathological dynamics in the basal ganglia," *Journal of Neuroscience*, vol. 26, no. 13, pp. 3567–83, 2006.

 ISBN: 978-1-61208-155-7