Tracking Non-stationary Dynamical System Phase using Multi-map and Temporal 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 multi-map recurrent neural architecture, exhibiting self-organization to deal with the partial observations of the phase of some dynamical system. The architecture captures the dynamics of the system by building up a representation of its phases, coping with ambiguity when distinct phases provide identical observations. The architecture updates the resulted representation to adapt to changes in its dynamics due to self-organization property. Experiments illustrate the dynamics of the architecture when fulfilling this goal.**

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

I. Introduction

Artificial agents often deal with environments with no known models. The environment can be considered as a dynamical system, whose phase changes over time. However, the agent should execute the suitable action corresponding to each phase of the dynamical system. The agent gets observations on-line from a perceptive stream, and performs a sequence of actions as a response to the input observations.

Observations may sometimes be ambiguous in the sense that the agent perceive similar observations for different system phases. However, the action to be taken relies rather on the system phase than on the observation itself. For the agent to perform the suitable action in the presence of such ambiguity, it should know the system phase at each time, or at least keep a reliable representation of it. Reliable representations should be obtained from a bijective mapping that can be implicit between the dynamical system phase space and the agent's representation space. Such built-in representational space is required for the agent to take the right action, using for example reinforcement learning techniques.

Moreover, the underlying environment dynamics is not always fixed, and its dynamics may be non-stationary. If the agent builds a fixed representation of the world dynamics, its performance will become poor when the dynamics changes. Non-stationary evolution of the system phase requires the agent to update its representation of the environment in an on-line and unsupervised manner.

Let us consider the toy example of a simplified autonomous power system where an artificial agent drives the process of a hydroelectric station.

The electrical load changes with time, depending on the consumption profile of the region it serves. The latter is a dynamical system whose phase x^t at a specific time t is the consumers activity. From current phase x^t of the system, the agent gets an observation $o^t = O\left(x^t\right)$ that is the current power consumption value. The phase changes over time and is ruled by a evolution function ϕ_t so that the next phase is given by: $x^{t+1} = \phi_t\left(x^t\right)$.

The consumption in short periods follows a regular dynamics, for example, consumption in day and night periods almost repeats its same values on 24-hours intervals within the same month, thus ϕ_t can be considered as fixed on short time intervals like one month. However, the consumption profile changes between summer and winter months. This is related to the use of warming systems, and to the change in consumption for lighting according to daytime length. On the year scope, the evolution function ϕ_t is changing and the dynamical system is non-stationary.

Let us first consider the case of short periods where ϕ_t can be considered as fixed. Let us also suppose that the consumption increases normally in the daytime to and decreases at night. This means that the agent will get the same observation $o^{t1} = o^{t2}$ in two different times within the 24-hours interval, each corresponding to a different phase $x^{t1} \neq x^{t2}$, one when consumption increases and the other when it decreases. Such observation is ambiguous, because knowing it is not sufficient for the agent to anticipate the system behavior which -in turn- is necessary to drive the station. Therefore, the first task of the agent is to distinguish the actual phase starting from ambiguous observations. A straightforward solution is to build distinct internal representations $\hat{x}$ corresponding to distinct system phases x in such a away that $\hat{x}^{t1} \neq \hat{x}^{t2}$ correspond to $x^{t1} \neq x^{t2}$ although the agent observes $o^{t1} = o^{t2}$. This means setting up a bijective mapping between the internal representation of the agent and the dynamical system phase space. Fig. 1 shows a schematic of the process of a dynamical system phase representation extraction.

On the other side, the evolution function ϕ_t changes on a year scope. The second task of the agent is to follow the changes in the system dynamics throughout the year, i.e.

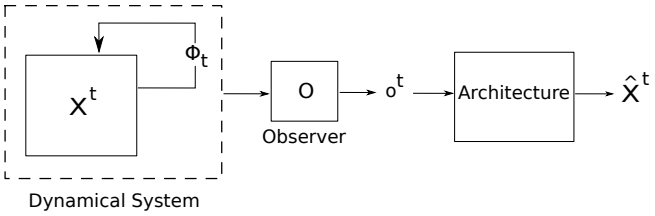


Figure 1. Dynamical system phase extraction.

to update its internal representations of the succession of phases while the dynamics changes, thus tracking the non-stationary behavior of the dynamical system.

This work presents an unsupervised method for the on-line extraction of an adaptive representation of a non-stationary dynamical system phase. The extracted representation forms a bijective mapping between the representation space and the system phase space. The proposed architecture depends on recurrent neural networks employed to build self-organizing maps (SOMs). The latter uses a neural field as a competition process that controls learning.

In the literature of recurrent neural networks considering the temporal dimension of the input, most works using self-organization rather focused on the clustering of input sequences [1], [2] than on setting up a mapping of the system dynamics generating these sequences. Some other works [3] focused on setting up an on-map representations to inputs, and used the difference in representations on the map to compare input temporal sequences. Reservoir computing approach [4] aims basically to set up a mapping between the input phase space and the reservoir state space. This resides in the mapping in a supervised way the relevant states in the large reservoir state space to the input space. However, the use of supervised learning leaves it necessary to re-train the model each time the system dynamics changes.

II. The Architecture

The proposed architecture is based on `bijama` model [5] developed in our team. `bijama` is inspired from biological information about the cerebral cortex. It enables building 2D-neural assemblies called *maps*, analogical to cerebral cortex structure. Each map contains a group of units inspired from a functional view of cortical columns. Units process external and internal signals carried out by connections. The proposed architecture consists of three maps, interconnected as illustrated in Figs. 2 and 3. The main map in the architecture is called the *input* map. This map receives the input stream values o^t and builds a spatial coding of the state $\hat{x}^t$ corresponding to the actual dynamical system phase x^t. In fact, this coding resides in the map activity profile, as show further. The other two maps are the *delay* map and the *associative* map. They are intermediate structures that form with the input map a macroscopic recurrent pathway. This pathway re-injects the input map activity

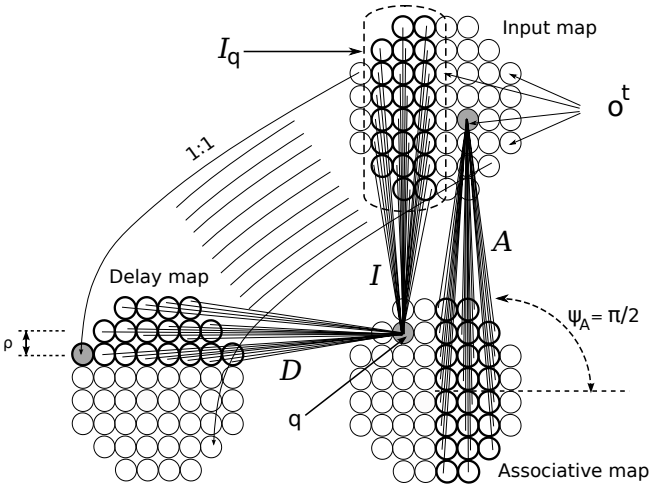


Figure 2. Model architecture. Input map and delay map are connected via one-to-one connections, other connections are one-to-many connections organized in strips. For unit q, $\mathcal{I}_q$ is the strip of kind $\mathcal{I}$ handled by q.

in its dynamics via strip-like connections. The interest of the intermediate architecture is to consider the past of the input stream in current map response. Units within a map are interconnected. Each unit is connected to others by an on-center/off-surround kernel [6], [7], giving the map a neural field structure. Neural fields are differential equations that describe the spatio-temporal evolution of a competitive process within a population. The neural field used here [7] executes competition between map units activities. It is parametrized so that lateral competition results in a single activity bump on the map. The bump-shaped global activity is used to guide the process of self-organization. This is rather difficult as explained in [7]. In `bijama` model, the evaluation of units activities follows an asynchronous and parallel scheme [5] not detailed here. Evaluation of all the units occurs at discrete time instances which are called *time steps*.

In addition to intra-map connectivity between units, there exist inter-maps connections as mentioned previously. Each unit at a position p in a map is connected to units at positions q belonging to a partial region in the remote map. This region has the shape of a strip as shown in Fig. 2. The strip-shaped region related to p is referred to as $\mathcal{S}_p$, and the positions within the strip as $q \in \mathcal{S}_p$.

The connection between a position p in the local map and a position q in the remote map handles a weight $\bar{s}^t_{pq}$ which is modified via a learning rule at every time step. These connections are called *cortical connections*. The unit p, manipulating cortical connections in strip $\mathcal{S}_p$, handles a vector of weights $\bar{S}^t_p = \left(\bar{s}^t_{pq}\right)_{q\in\mathcal{S}_p}$. The set of strips $\mathcal{S}_p$ for p in some map is referred to as $\mathcal{S}$ generally, but in Fig. 2 $\mathcal{S}$ should be replaced by $\mathcal{A}, \mathcal{I}, \mathcal{D}$ according to the name of the map the strips originate from.

Each strip has an orientation $\psi_{\mathcal{S}}$ measured as the angle

of the axis connecting the center position of both local and remote maps. In figure 2, $\psi_{\mathcal{A}} = 90°$ is represented for strip $\mathcal{A}$, and it can be seen that $\psi_{\mathcal{I}} = \psi_{\mathcal{D}} = 0°$ since strips $\mathcal{I}$ and $\mathcal{D}$ are horizontal. The width of the strip-shaped region is referred to as the $\rho_{\mathcal{S}}$ (see Fig. 2). The remote value of some unit q read through a connection is a scalar activity noted u_q^t. The unit p thus perceives a vector $S_p^t = \left(u_q^t\right)_{q \in \mathcal{S}_p}$ of some remote units activities via the strip $\mathcal{S}_p$.

In `bijama` model, a unit is modeled as a stack of modules as shown in Fig. 3. Each module handles a set of scalar values. These values can be computed from external inputs provided to the unit, or from scalars handled by other modules in the unit. Module stacks are the same for all units within a map, providing the map with a functional homogeneity. Fig. 3 facilitate the reading of the following description.

Let us start by describing the stack of the input map units in light of its function. Units in this map receive two entries: external input o^t and strips from the associative map (see strips noted $\mathcal{A}$ on Fig. 2). The map output is an activity bump which current position will be used to represent $\hat{x}^t$ as explained later.

The unit stack is composed of several modules. The lower one is the *thalamic module* which computes a similarity value θ_p^t by matching the external input o^t against some stored prototype ω_p^t:

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

Above the thalamic module is the *cortical module*. It computes the similarity $c_{p,\mathcal{A}}^t$ between the strip weight vector $\bar{A}_p^t$ and the strip activity vector A_p^t of the strip $\mathcal{A}_p$ handled by unit p:

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

Where B is a numerical constant.

θ_p^t and $c_{p,\mathcal{A}}^t$ are combined in a third module called *cortico-thalamic merging*, in order to determine the participation of thalamic and cortical modules in the competition described next. The combination is given by:

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

Where β is a constant.

The latest value is passed to the neural field module which computes the unit activity u_p^t, so that the u_p^t profile within the map is a bump arising around the highest ν_p^t.

The value of u_p^t is then used to modulate learning in thalamic and cortical modules. For thalamic learning, weights are updated as follows:

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

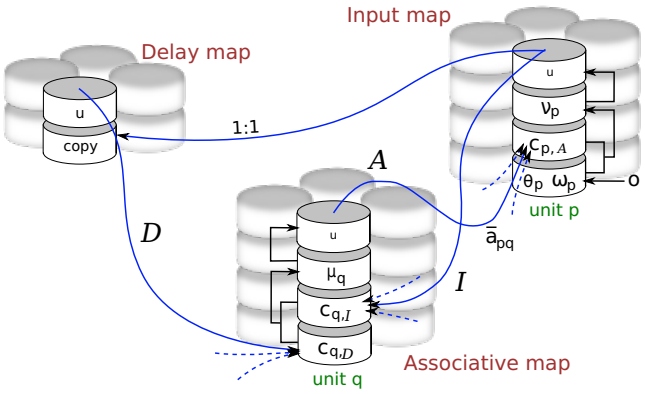


Figure 3. Module stacks of the units in the three maps and their cortical inter-connections.

Where α_ω is a fixed thalamic learning rate for all input map units. On the other hand, the same rule is applied for cortical learning for each connection weight in the strip $\mathcal{A}_p$:

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

$\alpha_{\mathcal{S}}$ being a unified fixed learning rate for all $\mathcal{S} \in \{\mathcal{A}, \mathcal{I}, \mathcal{D}\}$.

The second map in the architecture is the delay map. Its units are connected to the input map units via one-to-one scheme, so that each unit in the delay map is connected to the unit in the input map having the same coordinates. The role of the delay map is to copy the input map activity and delay it for a period T of time. There is no lateral connection neither neural field modules in this map. The stack of a unit in this map has two modules: a *copy* module that reads its value from an input map unit on the other side of the cortical connection and a *FIFO* module which implements a T-length queue. For each position q in the input map and the corresponding position p in the delay map the delay map activity is $u_p^t = u_q^{t-T}$.

The last map in the architecture is the associative map. Units in this map receive as inputs the activity of the input map (via strip $\mathcal{I}$ in Fig. 2) in addition to a delayed copy of this activity from the delayed map (via strip $\mathcal{D}$ in the same figure). The neural field of the associative map performs lateral competition between its units. The resulting activity is then re-injected in the input map dynamics through $\mathcal{A}$ strip.

In this map, the two lower modules of a unit q handle the strips $\mathcal{I}_q$ and $\mathcal{D}_q$ emerging from the input map and the delay map respectively. They compute the corresponding matches $c_{q,\mathcal{I}}^t$ and $c_{q,\mathcal{D}}^t$ similarly to equation 2. These two modules are then combined in a third module called *cortico-cortical merging*, and computes a scalar as follows:

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

The value μ_q^t is the actual input to the neural field of the associative map, which in turn computes the unit activity

u_q^t. Cortical learning for strip $\mathcal{I}$ and $\mathcal{D}$ connections occurs similarly to equation 5. Noise helps boosting the associative map activity in units receiving null cortical activity. This is necessary to avoid drastic bump eliminations in the input map.

The proposed architecture does not require any knowledge about the dynamical system neither about the observation stream, it is thus model free. As there is no need for any parameter adjustment during learning, the architecture is able to follow the changes in the dynamical system. This is why the learning rates in the architecture are fixed values: There is no coarse-mapping and fine-tuning learning stages that are often used in architectures based on self-organizing maps [8].

III. EXPERIMENTS

In this section, the goal is to test the architecture capability to find a suitable representation of the dynamical system phases, as well as the adaptation of such a representation to some changes in the evolution function ϕ_t. The architecture should set up a representation from the ambiguous observations provided by the dynamical system. The building up of such representation actually occurs in the input map. This is performed by the spatio-temporal self-organization of both the thalamic and the cortical weights [9]. Let us consider the system phase at time τ to be x^τ. When the phase is observed it gives the observation $O\left(x^\tau\right)$, provided as an input to the input map in the architecture. Like in Kohonen maps [10], each input value is presented to all the units in the input map. A single input $O\left(x^\tau\right)$ is presented and maintained during several time steps. This gives time for the neural field to form the bumps and for thalamic and cortical learning to occur. It appeared experimentally that a quite small $T = 24$ value was suitable for the experiment. In this experiment, τ is incremented every T time steps. and the input $O\left(x^\tau\right)$ is presented to the input map during these T time steps, so that:

$o^t = o^{t+1} = \cdots o^{t+T-1} = O\left(x^\tau\right)$ and $o^{t+T} = o^{t+T+1} = \cdots o^{t+2T-1} = O\left(x^{\tau+1}\right)$, etc.

The T used here is the same as the delay map FIFO length. This means that the input map activity is delayed until the next input $O\left(x^{\tau+1}\right)$ is sampled.

At the end of each T time steps, the neural field results in a stable bump. For further analysis of the map behavior, let us compute the barycenter G^τ of the activity u of all units in the map. It is computed as follows:

$$G^\tau = \sum_p u_p^t.p / \sum_p u_p^t \tag{7}$$

Giving $p \in \left\{(i,j) : \sqrt{(i-R)^2 + (j-R)^2} < R\right\}$, R is the radius of the input map, and t is the time step at the end of T interval.

When a group of l barycenter ($l = 50$) are computed, they are organized in a list $P^\tau = \{G^{\tau-l+1}, G^{\tau-l+2}, \cdots, G^\tau\}$ that form a path of successive bump positions. This will be drawn on the map snapshot as shown in Fig. 4.

In experiments, artificial observations are used as inputs. They are two series of values in the range $[0..1]$. Values in each series are fed to the architecture one by one periodically. The first series is ($S_1 = ABCDEFEDCB$) and the second one is ($S_2 = ABCBAFEDEF$) where the values are coded as follows: ($A = 0, B = 0.20, C = 0.40, D = 0.60, E = 0.80, F = 1$).

Ambiguous values exist in both input series. Some input values (D for example in S_1) are preceded by different values (once by C and once by E). Thus, the same observation could corresponds to two distinct states of the dynamical system. The proposed architecture is expected to resolve this ambiguity and find distinct representations for the same input value according to its temporal context, more specifically, in S_1 it should find two representations for each input except for A and F which are always preceded by the same series of values. Similarly, for S_2 it should find two representations for each value except for C and D. Furthermore, the architecture should update the way it maps the real system phase x^t over its surface, i.e. the way it determines the states $\hat{x}^t$ representing x^t, when the input series switches from S_1 to S_2. The robustness to noise is tested by perturbing each observation by a noise_o value before being presented to the architecture.

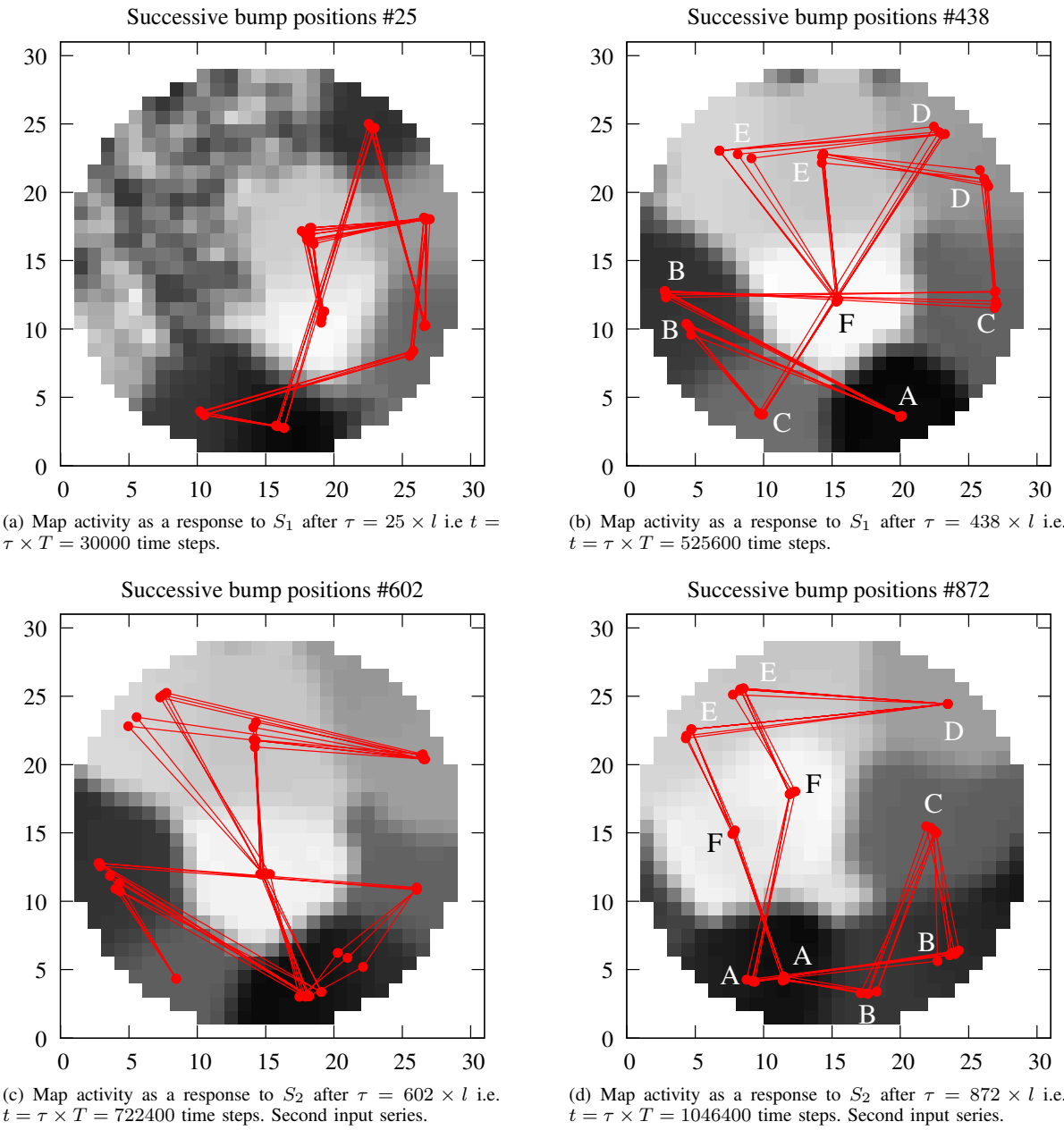
The architecture is trained on the first series S_1 for a period τ^s of time, then the input is switched to the second series S_2 for the rest of experiment time. This simulates the change in the dynamical system evolution function at time τ^s.

System evolution as a response on both input series is shown in Fig. 4. Fig. 4(a) shows the first stages of the self-organization process. On the left of the map, initialization with random ω_p^t is still visible (noisy grey-scaled area), while on the right, ω_p^t values are spatially organized (continuous variation of shade).

Along the experiment time, the duplication of states representation related to the same o^t occurs as self-organization sets up. This can be seen by comparing Fig. 4(b) to Fig. 4(a).

Fig. 4(b) shows that the map exhibits better spatially dispersed thalamic regions handling each a different range of o^t values. Six grey-scale regions can be distinguished, they correspond to the six values of input. Each unit in the map belongs to some region handling some o^t. This means that the map is fully recruited.

Each input is assigned to a thalamic region corresponding to its value. A region could be split into two regions corresponding to the same o^t value (as for the region encoding C values). However, the representation held by each region resolves the ambiguity of the observation in the input series. For example, the two phases that give the D observation



(a) Map activity as a response to S_1 after $\tau = 25 \times l$ i.e $t = \tau \times T = 30000$ time steps.

(b) Map activity as a response to S_1 after $\tau = 438 \times l$ i.e. $t = \tau \times T = 525600$ time steps.

(c) Map activity as a response to S_2 after $\tau = 602 \times l$ i.e. $t = \tau \times T = 722400$ time steps. Second input series.

(d) Map activity as a response to S_2 after $\tau = 872 \times l$ i.e. $t = \tau \times T = 1046400$ time steps. Second input series.

Figure 4. Status of the input map during the system evolution as a response for both input series. Grey-scaled values are the ω prototypes (black for 0, white 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 higher figures represent the map response on the first input series $S_1 = ABCDEFEDCBA$. The two lower figures represent its response on the second input series $S_2 = ABCBAFEDEF$. Step number t are computed with $l = 50$ and $T = 24$.

have two representations in a region corresponding to the same thalamic value. Each representation corresponds to the observation in a different context, one representation for D preceded by C and the other for D preceded by E. Therefore, the architecture has built two distinct representations for two different system phases although they correspond to the same D observation. Nevertheless, A and F have unique representations as they are not ambiguous. Thus, successive bump positions (red dots) actually correspond to each term in the input sequence (i.e. to the phases x), in spite of the redundancy of some terms in the sequence. More precisely, the poly-line that appears in the figure is formed by $l = 50$ points corresponding each to one G^τ. As sequences contain 10 items, $l = 50$ points shows five consecutive repetitions of full sequences. The motivation for computing G^τ points is to show the temporal succession of states. Each point in

the poly line corresponds to a representation $\hat{x}^t$ of a single phase of the dynamical system x^t. Clusters of points can be distinguished along the poly-line. Clusters result from the repeated visits of some state. At some position in the map (e.g. point E-left on Fig. 4(b)), there are rather clusters of G^τ. They are always well localized, meaning that every visit of the corresponding phase in the sequence leads to similar bump positions. This is an indication of the stability of $\hat{x}^t$ representation.

Fig. 4(c) shows the map state after switching to the next input series S_2. The past organization of the map which was fitting S_1 does not fit S_2 anymore. For example, the input F was corresponding to one representation on the map, while it is ambiguous now in S_2.

In Fig. 4(d), it can be seen that the map has re-organized to re-assign its regions to different thalamic values than in the case of S_1. New stable clusters of points are found, and the path formed by them expresses a stable correct representation of S_2. Two special cases should be emphasized: while F was not ambiguous in S_1 and was corresponding to one representation on the map, it is ambiguous in S_2, and the map has re-organized to find two distinct representations. The same is true for A. The second case is the case of D which was ambiguous is S_1 and corresponding to two representations, but it corresponds to one representation in S_2 where it is not ambiguous. Noise was always present in both input series values and didn't affect the architecture ability to find a suitable mapping.

The experience was launched with numerical values for the dynamical system as follows: $\tau^s = 25000$ and noise_o is sampled from a uniform random noise $\mathcal{U}[-0.05, 0.05]$.

Architecture numerical values was initialized as follows: $R = 15$ 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 was sampled from a uniform random distribution $\mathcal{U}[0, 0.1]$, $\rho_\mathcal{I} = \rho_\mathcal{A} = \rho_\mathcal{D} = 5$. $\psi_\mathcal{I} = 90°$, $\psi_\mathcal{A} = 90°$, $\psi_\mathcal{D} = 0$, $T = 24$, and $l = 50$.

IV. Conclusion and future work

Considering the obtained results, it was experimentally shown that the proposed architecture is able to self-organize in order to set up a suitable representation that maps in a bijective way to the phase of a dynamical system, and to overcomes the ambiguity of observations. The architecture was also able to adapt on-line to the changes in the system dynamics, without the need to re-parametrization during simulation. There is no conditions on the dynamical system, meaning that it is a model-free architecture. The architecture has also exhibited the full recruitment of the resource map units, besides to the capability to re-organize them to adapt to the changing dynamics of the concerned system, ending by setting up an adaptive representation.

Future work will be oriented in two main directions. First, the robustness of the spatio-temporal self-organizing process has to be investigated with much larger maps since scalability condition still need to be investigated, allowing the representation of more complex dynamical systems. With more available space on the map surface, the architecture should be able to represent richer systems, or several separate systems at once. Second, the longer term purpose of our work is to set up Markovian state space from partial observations, allowing an agent to schedule actions in Partially Observable Markovian Decision Processes framework. This implies considering actions rather than just observing the world transitions, which requires extensions of the architecture proposed in this paper.

Acknowledgments

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

http://intercell.metz.supelec.fr

References

[1] 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.

[2] 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.

[3] T. A. Mcqueen, A. A. Hopgood, and T. J. Allen, "A recurrent self-organizing map for temporal sequence processing," in *in: Proceedings of Fourth International Conference in Recent Advances in Soft Computing (RASC2002*, 2002.

[4] 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.

[5] 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.

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

[7] 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.

[8] O. A. S. Carpinteiro, "A hierarchical self-organizing map model for sequence recognition," *Neural Process. Lett.*, pp. 209–220, 1999.

[9] B. Khouzam and H. Frezza-Buet, "Discovering the phase of a dynamical system from a stream of partial observations with a multi-map self-organizing architecture." ThinkMind Digital Library, 2011, COGNITIVE 2011, to appear.

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