INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

A DSOM hierarchical model for reflexive processing: an application to visual trajectory classification

Claudio M. Privitera and Lokendra Shastri

TR-96-011

June 1996

Abstract

Any intelligent system, whether human or robotic, must be capable of dealing with patterns over time. Temporal pattern processing can be achieved if the system has a short-term memory capacity (STM) so that different representations can be maintained for some time. In this work we propose a neural model wherein STM is realized by leaky integrators in a self-organizing system. The model exhibits *compositionality*, that is, it has the ability to extract and construct progressively complex and structured associations in an hierarchical manner, starting with basic and primitive (temporal) elements. An important feature of the proposed model is the use of temporal correlations to express dynamic bindings.

1 Introduction

Many cognitive processes must be *compositional*, that is, they must be capable of extracting and constructing progressively complex and structured representations in a hierarchical manner, starting with simple primitive elements [4], [3], [26].

For example, visual recognition of complex objects and events requires the rapid assembly of diverse information about the subcomponents of the object or event. In this case, primitive elements are combined into more complex associations until a complete meaningful representation of the visual input is achieved [5], [8], [14]. Another phenomena where compositionality plays an ubiquitous and critical role is language where a sequence of acoustic input is mapped into a complex and high dimensional description of events and states, and motor control where abstract representation of motor programs are decomposed during movement generation into more primitive motor schemas [2], [9].

A central requirement of compositionality and hierarchical processing is the maintenance and propagation of dynamic *bindings* between appropriate components of a multi-level and distributed representation [6], [26], [27]. Recent neurological data highlights the potential role of the temporal structure of neural activity in the expression and propagation of dynamic bindings [28], [27]. A particularly promising hypothesis is that all information pertaining to a single entity is bound together as a result of the synchronous firing of the various nodes encoding information about this entity [30], [26]. Thus compositionality, binding, and temporal pattern processing are interrelated and seem to play an essential role in cognition [3].

In this work, compositionality is achieved by a hierarchical model of Self-Organizing Maps (SOM) and the temporal processing ability is achieved by exploiting neurons capable of supporting short-term memory (STM). Each map in the hierarchy (which we refer to as Dynamic SOM — DSOM) represents a specific abstraction of information expressed in maps that are lower in the hierarchy. The unification among different maps is realized by the synchronization of activity of appropriate nodes across these maps.

Figure 1 shows a generalization of this mechanism. In this case, all the neuronal areas involved in the current execution are represented by means of specific and distinct zones on the map surface: the general idea is to appropriately set an initial sequence (in term of phase difference and frequency) of dynamic activity in the lowest levels of the hierarchy and to propagate this activity up to the top map and thereby achieving a complex description of what is occurring in the input scene. The temporal activity is represented by a sequence of impulses, where each impulse corresponds to the firing of a particular bunch of neurons in the map.

The relationship between an output impulse at level L_i and constituent input impulses from level L_{i-1} is bi-univocal in the sense that, the analysis of impulsive activity at the upper level provides the necessary cue to deterministically identify the constituent impulses in the lower map.

Section 2 presents an overview of the neural paradigm used in the model and section 3 defines the dynamic extension of this paradigm. Section 4 discusses an example of input scene analysis and presents some experimental results. Finally section 5 discusses additional properties of the model.

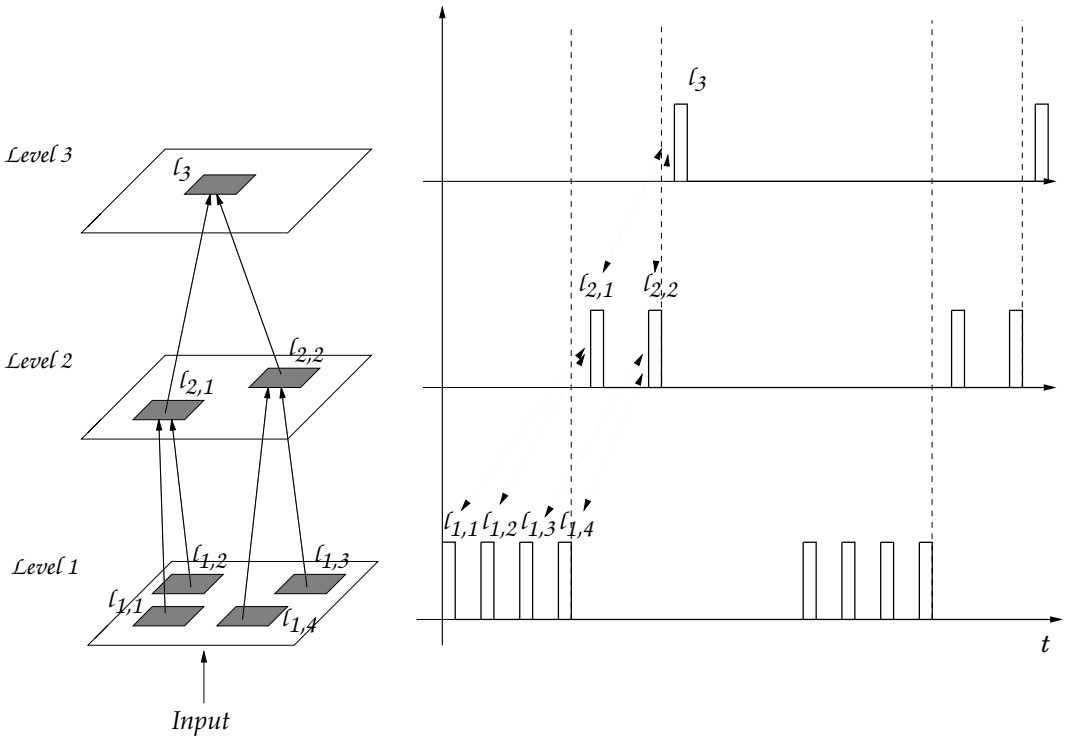

Figure 1: *Three different levels of maps representing a sort of hierarchical knowledge representation: the lower level represents primitive instances whereas the higher levels represent combinations of these instances. Binding is represented by time occurrence between the input and the output sequence of impulses. This is a bidirectional relationship.*

2 Self-organizing Mapping

From a formal point of view, a Self-Organizing feature Map (SOM) is a systematic parallel assemblage with a fixed topology of processing elements (or neurons), each of which is characterized by a prototype vector $\boldsymbol{\pi}_i \in \Re^k$ representing the center of a receptive field.

Every element of the map reads a common input vector $\boldsymbol{\epsilon} \in \Re^k$ and consequently determines its own activation level by means of its activation function $u_i(\boldsymbol{\epsilon})$ [1].

The activation function can also embody a competition among the elements in order to enhance the selectivity of the map (a classical example of this competitiveness is the well-known *winner_take_all* technique wherein, only the most resonant element represents the output of the map).

The weight vector defining each element is automatically learned by the following well-known Hebbian learning rule:

$$\Delta\boldsymbol{\pi}_i(\boldsymbol{\epsilon}) = \eta(\boldsymbol{\epsilon} - \boldsymbol{\pi}_i)u_i(\boldsymbol{\epsilon}) \tag{1}$$

where $\boldsymbol{\epsilon}$ represents one of the input examples, η is the learning rate and $u_i(\boldsymbol{\epsilon})$ is the activation function.

The parameter η and the activation function $u_i(\boldsymbol{\epsilon})$ together define many key properties of the map: for example, in a traditional Kohonen model ([10], [11]), $u_i(\boldsymbol{\epsilon})$ is a simple window function centered on the most resonant element so that each weight vector of the map is updated during each step of the learning phase on the basis of its proximity to the *winner* element: in this way the map preserves the topology of the input data manifold in the sense that, neighboring elements in the lattice store neighboring patterns in the input space. The inverse topological property (neighboring patterns to neighboring elements) cannot hold unless the dimensionality of the lattice is the same of the input space; moreover, is very difficult to accurately approximate non-convex input data manifold.

Some of the drawbacks mentioned above can be overcome by representing the input topology with a connection net evolved during learning: in this case, the function $u_i(\boldsymbol{\epsilon})$ is not evaluated in the map but in the input vector space so that finally, the neighborhood between two elements is highlighted by the presence of a connection between them and not by their proximity in the lattice (see [13] and [12]).

An interesting generalization of the self-organizing process is to associate with each element of the map a second *output* vector $\boldsymbol{\mathcal{Y}}_i \in \Re^Q$ (see Figure 2) so that the map can finally be seen as a system with two layers of weights defining a generic mapping $f : \Re^K \rightarrow \Re^Q$, where usually $K > Q$ ([25], [24]). In this case, the input feature vector making up the training set is a pair $(\boldsymbol{\epsilon}, \boldsymbol{\mathcal{Y}})$ and the training of the network is achieved with the following learning equation used in parallel with equation (1):

$$\Delta\boldsymbol{\mathcal{Y}}_i(\boldsymbol{\epsilon}, \boldsymbol{\mathcal{Y}}) = \eta(\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{Y}}_i)u_i(\boldsymbol{\epsilon}) \tag{2}$$

In general, at the end of the training, each element of the map represents a specific part (often called *tile* referring to the well known Voronoi tessellation) of the input training

[1] The value of the activation function $u_i(\epsilon)$ is classically called the *resonance level* of the element because it reflects the similarity between the weight vector associated to a element and the current input vector (for instance, the activation function can be defined by the distance $\| \pi_i - \epsilon \|$). In this sense, an input vector fed to the map involves an internal place-coded probability resonance distribution having the maximum peak just located on the most similar element (see for instance [29]).

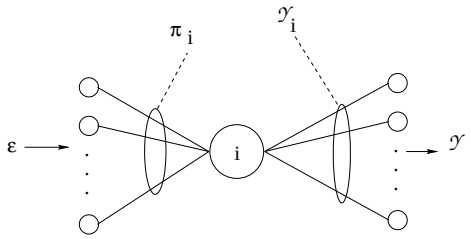

Figure 2: *The input-output structure of the weights vector of a neuron.*

manifold: this *tile* is centered around the weight vector $\boldsymbol{\pi}_i$. The corresponding vector $\boldsymbol{\mathcal{Y}}_i$ represents the output label of the element so that, if i is the most resonant element related to an input $\boldsymbol{\epsilon}$ $(u_i(\boldsymbol{\epsilon}) = max_j\ u_j(\boldsymbol{\epsilon})))$, then $\boldsymbol{\mathcal{Y}}_i$ represents the output mapping $f(\boldsymbol{\epsilon}) = \boldsymbol{\mathcal{Y}}_i$ for that input vector.

Both biological plausibility and statistical properties of self-organizing process can be enhanced by using as the activation function the following normalized Gaussian or softmax function:

$$u_i(\boldsymbol{\epsilon}) = U_i(\boldsymbol{\epsilon}) = \frac{G(\parallel \boldsymbol{\epsilon} - \boldsymbol{\pi}_i \parallel}{\sum_j G(\parallel \boldsymbol{\epsilon} - \boldsymbol{\pi}_j \parallel} \tag{3}$$

where, $G(\cdot)$ is a gaussian with fixed variance σ [1].

In this case, the optimal estimation of a continuous smooth function $f : \Re^K \rightarrow \Re^Q$ can be achieved by means of the following population coding:

$$f(\boldsymbol{\epsilon}) \approx \sum_i \boldsymbol{\mathcal{Y}}_i U_i(\boldsymbol{\epsilon}) \tag{4}$$

where both the vectors $\boldsymbol{\pi}_i$ and $\boldsymbol{\mathcal{Y}}_i$ are learned using the classical parallel Hebbian rule above introduced.

It is worth highlighting that, the previous equation is a sum of *radial basis function* [18] and it can be also proved that Eq. (4) is a solution of the *regularization problem* [16], [17].

The self-organized map represents a distributed model where a parallel assembly of preset processors can operate concurrently on a common afferent signal handling large amounts of information rapidly and efficiently: moreover, the modality of information representation and processing, makes additional interactions with other modules remarkably easier.

3 Dynamic Self-Organizing computational Maps

Temporal processing requires a STM capacity in order to represent different temporal components over a window of time. We use an extended form of SOM that exhibits the necessary STM property. Unlike SOM, the extended model, D-SOM, uses a dynamic version of a classical neuron model.

In the D-SOM model, the activation of an element in the map is represented by a membrane potential $P_i(t)$ which in turn is a function of the input activation function $u_i(\boldsymbol{\epsilon}(t)))$ which measures the degree of match between the element's weights vector and the temporal input vector $\boldsymbol{\epsilon}(t)$ (for example, $u_i(\boldsymbol{\epsilon}(t))$ could be the softmax function). $P_i(t)$ also takes into

account the dynamic properties of the biological membrane of a cell which are approximated by a generic RC circuit:

$$\frac{dP_i}{dt} = -\frac{P_i(t)}{\tau_i} + u_i(\boldsymbol{\epsilon}(t)) \tag{5}$$

where τ_i is the time constant of the RC.

Note that the input activation function $u_i(\cdot)$ of a map element is a function of time $u_i(\boldsymbol{\epsilon}(t))$. Thus the membrane potential of an element is correlated with the temporal evolution of its input and the element can not only analyze a static input vector $\boldsymbol{\epsilon}$, it can also perform a temporal integration of the function $u_i(\boldsymbol{\epsilon}(t))$ (hence the name, Leaky Integrator element (see for example [23])). If this temporal integration depends on the occurrence of certain spatio-temporal features/sub-events in the evolving temporal input, then the activation of the element indicates the occurrence of an external event composed of these sub-events. Thus one can view each element as a recognizer of a complex event.

In view of the above, each element is composed of a set of sub-elements — classically called *taps* [19] — each of which represents different critical points in the temporal evolution of the input vector. Specifically, if an element recognizes an event composed of m temporal features, it consists of m distinct taps. If the input vector is k-dimensional, $\boldsymbol{\epsilon}(t) \in \Re^K$ then, each tap is characterized by a weight vector $\boldsymbol{\pi} \in \Re^K$. At each time instance, the input vector $\boldsymbol{\epsilon}(t)$ is processed by all the taps and the membrane potential of the element is evaluated on the basis of the tap outputs:

$$\frac{dP_i}{dt} = -\frac{P_i(t)}{\tau_i} + \sum_{j \in Act_i(t)} u_{ij}(\boldsymbol{\epsilon}(t)) \tag{6}$$

where, $u_{ij}((\boldsymbol{\epsilon}(t))$ is the activation function of the $j-th$ taps of the $i-th$ neuron and τ_i is the time-constant of the neuron.

The term $Act_i(t)$ represents the set of active taps of the $i-th$ neuron at a generic time instant. As soon as the activation function of a tap exceeds a pre-fixed threshold it is inhibited and excluded from the set of active taps. Different elements may have different activation strategies for taps. In some cases, all taps may be active initially. In other cases, taps may become active one at a time, and hence, respond to temporal features occurring in a specific order. This is achieved by connecting the taps in a sequence as shown in Figure 3. Notice that taps are represented by neurons, $tap_1, \ldots tap_m$, that read a common input vector. A tap can either be in an enabled state, wherein it actively processes its input, or it can be in a disabled state. Initially only the first tap is enabled. When an enabled tap receives a matching input signal, it crosses its threshold and fires, enables its successor tap, and at the same time inhibits itself thereby entering a disabled state.

The neuron n_i (see Figure 3) integrates the tap outputs and maintains a continuous potential according to Eq. (7). This neuron eventually fires if all the taps have been activated by the temporal input. The time constant τ_i limits the delay between successive temporal features and the STM of the integrating neuron.

In other words, the resonance of each element of the map not only represents a planar projection of a general point belonging to a multi-dimensional input manifold but it also represents the evolution of this point such that the input space acquires another coordinate namely, *time*. The final resonance of an element highlights that the input vector has *followed*

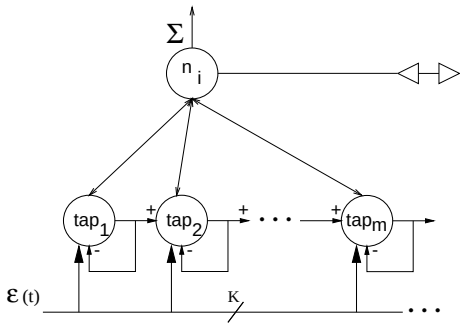

Figure 3: The general structure of partitioned leaky integrator neuron. The taps can be organized by an input layer of neurons connected each other in a way representing the activation policy of the model: once a generic tap match the input, it inhibited itself and sequentially enables the next. The integrating neuron n_i keeps memory of all the sequence of matching.

a specific trajectory in the input space defined by the sequence of the taps of the element. The dynamic property of the element enables it to preserve memory during the occurrence of two consecutive input temporal features; if the input trajectory has excessive delay between two consecutive sub-events, (the delay sensitivity is of course a function of the time constant of the element) the resonance approaches the zero value (the RC is discharged). In this case everything is reset to the initial values and finally the element starts to analyze the input stream again taking into account all the taps (or starting from the first one again, depending on the activation strategy of that element).

The learning of tap weights can be done in a traditional Hebbian manner exploiting a unique vector $\boldsymbol{\pi} \in \Re^{K \times m}$ for each iteration. In [19] an alternative learning method has been presented where the taps are learned in real time using a sequence of input vectors. In [31] another method is proposed where taps are pre-determinated and the learning process only effects the connections between taps and neuron.

Figure 4 illustrates the behavior of D-SOM. The Figure shows an input manifold at three distinct time instants and the arrows on the left side of the Figure indicate three input temporal features of the map for each of the three instants: the exact sequence of these input features represents a specific *event* that has been learned during the learning phase. In particular, with the occurrence of the first temporal feature, the map responds with a resonance distribution located around the element with a tap that matches the most with that feature (see Figure 4, right side: in this example the map also preserves the topology so that the resonance distribution corresponds to a compact resonance bowl in the map surface). With the evolution of the input motion, the resonance not only maintains a significant value but it also increases as soon as the input trajectory exhibits all the temporal features represented by the taps of that element [2].

This temporal property of D-SOM can be exploited in the context of motor control, time series analysis, temporal sequence representation and recognition (see [15], [19], [20],

[2]Some basic simplification has been assumed in this case for the sake of exposition. The simplification is that the first reference point of the event in progress, is specific to that event. Usually, during the first part of an input temporal stream, the resonance distribution observed is located in zones that could be far away from the final localization: this is because some different temporal events can share initial segments.

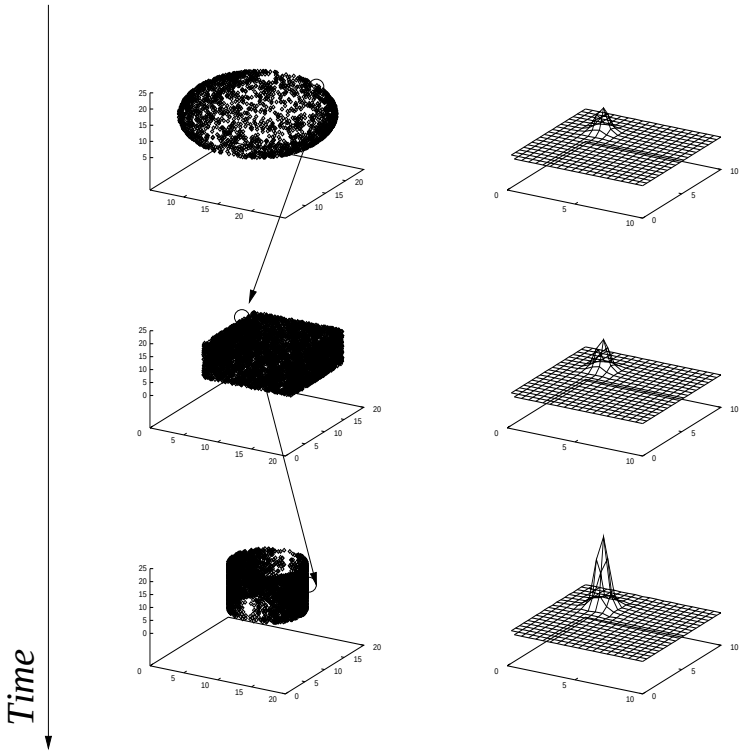

Figure 4: *The evolution of an input manifold in the time: the arrows indicates three different inputs of the map in three different non consequent time instants. On the right, the corresponding resonance distribution on the map is displayed.*

[22], [21]).

We have exploited the D-SOM paradigm to analyze an input train of impulses where each impulse represents information about an input scenario: the occurrence of a specific series of impulse has to be interpreted and transformed into an output impulse finally achieving a hierarchical structure where the binding between the incoming and outgoing impulses of the map is represented by their time occurrence.

4 Firing process

As discussed in the introductory section, synchronous activation seems to be the mechanism that binds different cortical instantiations of information pertaining to an event; in this section, we review this property in the light of Dynamic SOM computation.

In the previous section we have analyzed the dynamic behavior of an Leaky Integrator element and studied the correlation between the membrane potential $P_i(t)$ and a generic external phenomenon that can be viewed as a temporal function. The membrane potential represents the activation level of a neuron and can be evaluated on the basis of a pre-fixed threshold which consequently defines the firing time of that neuron.

The firing of an element in the map will be referred to as an output *impulse*. Each impulse is characterized by a label that corresponds to the output weight vector of the firing element. In this sense, we will often use the term *weighted* impulse.

Many elements can fire in different time instants so that the temporal output function of a generic map corresponds to a train of *weighted* impulses that is fed to the upper map. At each level, the input train is translated into an output train keeping appropriate bindings between input/output impulses based on their time of occurrence.

The threshold of each element plays a key role in this process and certain negative side effects on the global map activity depend on this threshold. For example, let us consider two generic elements i and j each composed of two taps and representing two different input events E_1 and E_2. Assume that E_1 is composed of subevents e_1 and e_3 and E_2 is composed of subevents e_2 and e_3. Now, if we consider the input impulse train e_1, e_2, e_3 which occurs at time t_1, t_2, t_3 respectively then, the two elements receive the initial matching impulse in two different instant t_1 and t_2 (see Figure 5) but receive the final impulse at the same time instant t_f. Consequently, the two elements will fire simultaneously at time t_f [3].

This possibility, referred to as *spike-overlapping*, can obviously lead to an ambiguous output train of impulses wherein any distinction between the input events E_1 and E_2 cannot be detected.

In order to achieve the opposite behavior (referred to as *spike-non-overlapping*) so that, different spikes corresponding to different input events are separated in time, we define a global mutual-exclusion between all the elements of the map using a common threshold

[3] In this case, the input function $u_i(t)$ has a threshold mechanism built in so that equation 6) becomes:

$$\frac{dP_i}{dt} = -\frac{P_i(t)}{\tau_i} + \sum_{j \in Act_i(t)} sgn(u_i j(\epsilon(t)) - \lambda) \tag{7}$$

where, λ is the internal tap threshold of the element which tunes the desidered matching level between the tap and the input impulse.

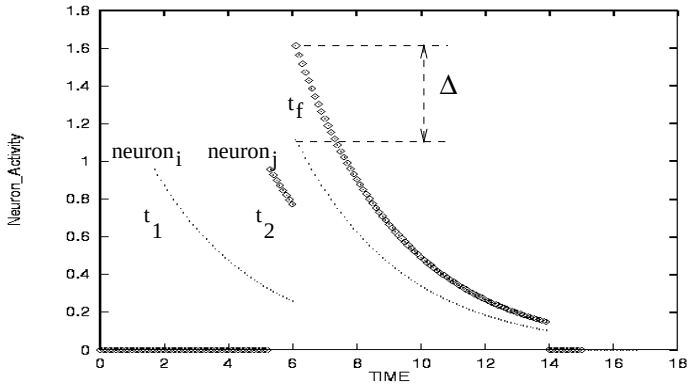

Figure 5: *The Figure shows the membrane potential of two different neurons receiving the first impulse in different instants and the last common impulse at the same time: at this moment the two neurons (representing different features composition) will spike together. The difference Δ of the membrane potential can be exploited in order to appropriately discriminate the spikes of the two neurons.*

function $\Lambda(t)$. Figure 6 shows this function where the time period ARP represents the global absolute refractory period of the map and the 0 time instant represents the firing of a generic element of the map.

Figure 5 shows the time course of the membrane potential of the two elements i and j (the dotted and the squared line): the difference Δ between the membrane potentials of the two elements at time t_f can be exploited for separating the spiking of the two elements. In fact, only the element identified by the minimum membrane potential is allowed to fire, and the firing of the second element is delayed by an amount of time depending on the global refractory period [4]. This is done by using the common threshold function $\Lambda(t)$. The second element that does not fire, is required to maintain a significant potential value during the global refractory period ARP: this is the reason why, the minimum membrane potential element is allowed to fire first.

The impulse output function $\epsilon(t)$ of the map can be finally represented by the following

[4] In order to detect the minimum membrane potential in the map we can assume that the sign of the membrane potentials is inverted and then the maximum of the inverted values is evaluated.

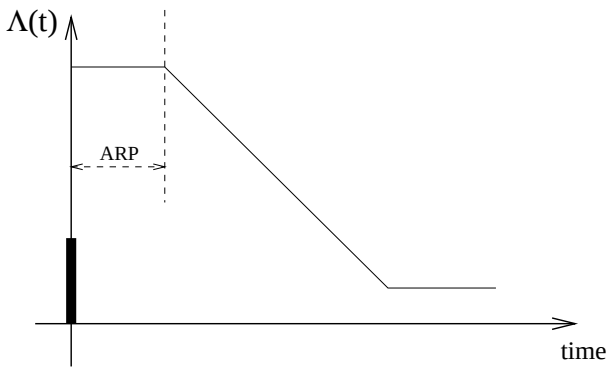

Figure 6: *The time course of function* $\Lambda(t)$ *representing the global threshold function of the map after a spike occurred at the* 0 *time instant.*

population code computed over all the elements of the map:

$$\epsilon(t) = \begin{cases} \mathcal{Y}_i & \text{if } A_i(t) = 1 \text{ and } P_i(t) = min_{j \,:\, A_j(t)=1,\;\; P_j(t)>\Lambda(t)} \;\; P_j(t) \\ 0 & \text{if } \forall i \quad A_i(t) = 0 \end{cases} \tag{8}$$

where, $\mathcal{A}_i(t)$ represents the total inhibition of the element that equals 1 only if all the taps of the element have been matched during the analysis of the input impulse train (and consequently inhibited) and 0 otherwise, and $\mathcal{Y}_i$ is the output weight vector of the element. In this way, each output impulse is synchronized with the time instant when at least one element has been completely matched by the impulse input function. The comparing of the membrane potential of the involved elements with the common threshold function $\Lambda(t)$ implicitly achieves the mutual-exclusion process and avoids *spike-overlapping* between elements representing different events.

5 A vision task example

The general behavior of the model can be tested in a visual context where, for example, the system is asked to analyze the motion of multiple points in a two dimensional input space and eventually recognize the occurrence of certain coordinated patterns of motion. This task involves three levels of representation: the trajectory of the points in the visual scene, the relationship between different points and the evolution of this relationship over time. Those, three D-SOM levels are defined below.

5.1 Level I: trajectory analysis

In the first level, simple information about line tilting and direction of a specific point motion is detected and classified. All the elements of the map share the same visual field and the input activation function of these elements can be directly derived from some neurological hypothesis: for example [7] suggests the existence of motion detectors with elongated spatio-temporal receptive field tilted in a preferred direction. A temporal summation determinates the energy level of the neuron.

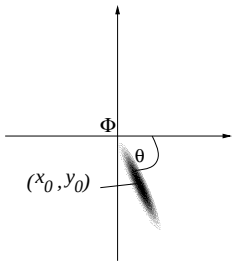

Figure 7: *An example of the trajectory detector filter exploited in the first level: the center of the receptive field* Φ *and the center of the bi-dimensional gaussian are translated each other in order to detect the direction of the input movement.*

With some simplifications, we model the input anatomic conformation of the motion detector neuron with the following non-symmetric and bi-dimensional gaussian filter:

$$\mathcal{GF}(x, y, x_0, y_0) = exp[\frac{-[(x - x_0)^2 + (y - y_0)^2]}{2\sigma_1^2} + \frac{-(xcos\theta - ysin\theta)^2}{2\sigma_2^2}] \tag{9}$$

where σ_1 and σ_2 are variances with $\sigma_1 >> \sigma_2$, x_0 and y_0 are the centers of the gaussian and θ represents the tilt of the filter. The center (x_0, y_0) of the non-symmetric gaussian is translated by a specific radius from the center $\Phi = (\Phi_x, \Phi_y)$ of the neuron receptive field so that, if the tilt θ is oriented along the segment connecting the two centers, $tan(\theta) = \frac{x_0 - \Phi_x}{y_0 - \Phi_y}$, we end up into a filter capable of representing the trajectory as well as the direction. An example of such a filter conformation is displayed in Figure 7.

Consequently, each element of the map corresponds to a specific filter characterized by a common receptive field dimension but with its own conformation in terms of tilt θ.

The input visual field is assumed to be a bi-dimensional white space where a black dot is moving. When the motion starts, the receptive field centers Φ_i of all the elements in the map are placed on the starting point. Consequently, the input function activation $u_i(t)$ of every element can be expressed by the following equation:

$$\frac{du_i(t)}{dt} = \mathcal{GF}(x(t), y(t), x_{i_0}, y_{i_0}) \tag{10}$$

where $(x(t), y(t))$ represents the current coordinate of the moving point.

Any clue concerning the velocity magnitude is obviously neglected because the input visual field area is always the same and we assume that the velocity is constant among different observations. At the end of the observation, the most active neuron classifies the input trajectory and direction and is allowed to fire.

The training of the above structure can be performed simply by feeding the map with a uniform distribution of pairs $< x_0, y_0 >$ which define the characteristic of the filters; each element is finally labeled with the corresponding angle value $\theta = arctg(\frac{x_0 - \Phi_x}{y_0 - \Phi_y})$.

5.2 Level II: basic relationship between pair of trajectories and dynamic coupling

Consider the problem of analyzing the trajectories of two moving points and determining if the pair of points are diverging or converging or, if they are moving in parallel, the direction

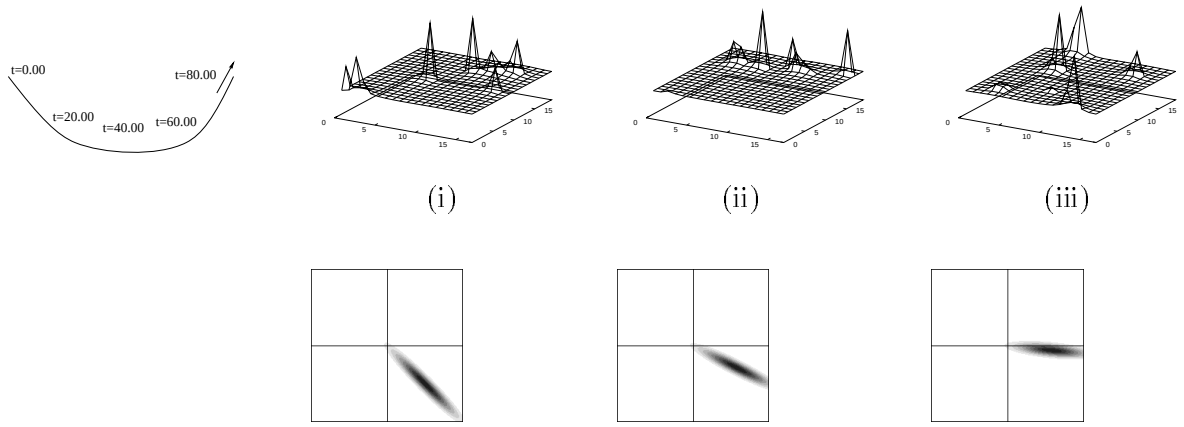

Figure 8: *An example of an input trajectory (top left) sampled in different time instants and the corresponding activity distribution on the first map surface (on the right, in sequence); just below each activity distribution snapshot, the filter conformation of the most active neuron is displayed.*

in which they are directed. The ability to carry out such an analysis would be of obvious significance.

The input visual field can be sampled at regular time intervals φ referred to as a *phase*. During each sampling, the receptive fields of the first level map are calibrated on a specific moving point and at the end of the sampling period, the most active neuron represents the corresponding direction and fires. After the impulse, the *attention* of the map shifts to another moving dot and the same process is repeated.

The elements of the second level are consequently composed of two input taps corresponding to all possible input pairs of directions. The output vector $\mathcal{Y} \in \Re^2$ is simply set equal to the difference and the average values of the two input taps. In the following simulation, the first level is a grid of 18×20 elements and the second level of 100×100 elements. Both of the maps have been trained by the softmax rule so that, any topological information is detectable from the map activation.

Figure 8 shows an example of input motion. The motion corresponds to the mouse movement in a graphic window of a workstation screen: the arrow shows the direction of the movement whereas the sequence of bracketed windows shows the sequence and the magnitude of the sampling periods. For each sampling, the corresponding initial time instant is specified at the top of the window. The corresponding global activity of the map at the end of the first three sampling (Figure 8 on top right, (i)-(iii)) is displayed with the filter of the most active element (just below each global activity snapshot).

The general behavior of the model is analyzed with three different motions evolving at the same time in the input scene. The sampling frequency is kept constant for all the input motions so that each impulse is simply identified by a phase.

Figure 9 shows the experiment: the three different motions are analyzed by means of a sequence of sampling (highlighted by small arrows along the trajectories on the left side of the Figure) which are synchronized in three distinguished phases $\varphi_1, \varphi_2, \varphi_3$. The graph on

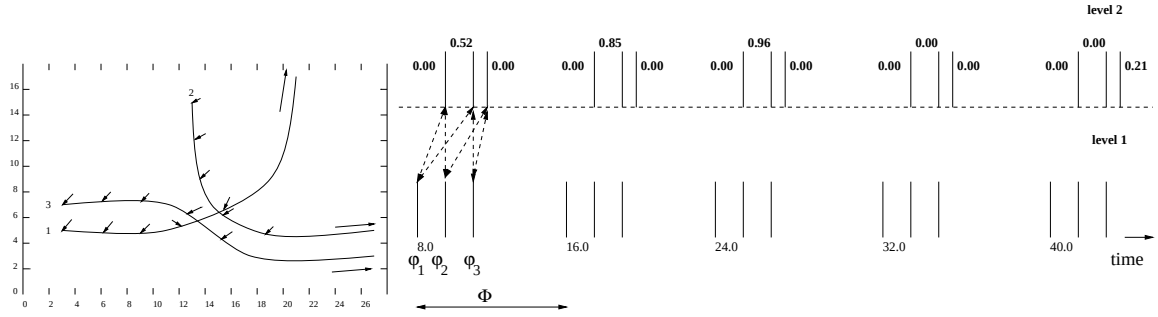

Figure 9: *Analysis of three simultaneous trajectories. Left: the arrows highlighted the sampling points. Right: the sequence of output impulses in the two level of maps; the numbers associated with the upper sequence represent the difference of the two input angles (in particular the gaussian of the difference).*

the right shows the output impulse trains in levels 1 and 2: in particular, for each impulse on the second level, the output value of the firing neuron is specified in the graph. This value is the difference of the two input angles (corresponding to the two different taps) and represents the degree of *parallelism* of the two composing motions. In particular, in Figure 9, the gaussian of the difference is exploited instead of the difference.

The binding between upper and lower impulses is simply achieved by keeping track of the time instant when each tap of the firing element in the upper level is matched by an input impulse. This key mechanism has been studied as a function of the time constant of each element and the refractory period ARP of the map. As we expected, the overall behavior of the model deteriorates outside of a specific range of phase values. In fact, if the input impulses are too close to each other, then the output activation of the involved elements is almost the same even if they represent different events so that at the end, the *spike-non-overlapping* capability no longer holds. On the other hand, it is reasonable to expect some retroactive effects if the time delay between two consecutive impulses exceeds the memory capabilities of the neurons.

All the phase values between these two extremes, (called *working area*) should generate correct synchronization between the two levels of the map and the magnitude of the working area can be modified by tuning the time constant of the elements and the ARP value of the global threshold function.

Figure 10 shows the lower and the upper bound of the working area as a function of ARP and for two different values of the time constant during a computer simulation of the model. The two bounds represent the cut off values of the phase, that is, bounds below and above which the model loses its binding capabilities. Figure 10 shows that, if the ARP value increases, the working area shrinks. The magnitude of the working area can also be tuned by modifying the time constant of the RC (see Eq. 6).

5.3 Level III: complex movement detecting

The third last level of the hierarchy analyzes and recognizes global coordination of a pair of trajectories evolving at the same time in the input space.

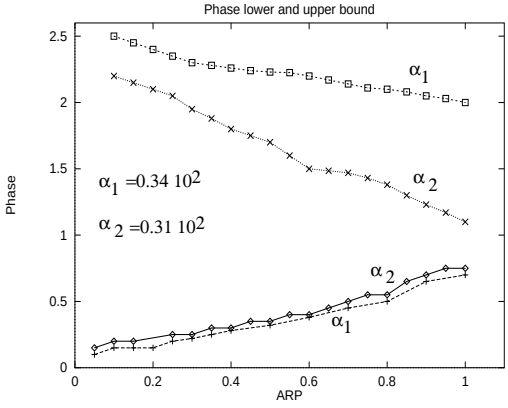

Figure 10: *The lower and upper bound (defining the working area of the model) of the phase delay between two consecutive impulses in function of ARP and the time constant α of the neurons.*

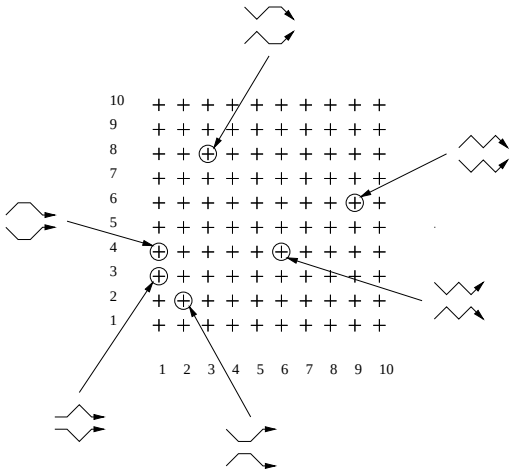

Figure 11: *The conformation of the third level map at the end of the training phase: for each input pattern, the most resonant neuron is corresponding labeled.*

In the following simulation, six different input patterns (shown in figure 11) have been taken into consideration. The map consists of a 10×10 grid of elements each of which is composed of four taps. Thus each element can represent a *four-stroke* component coordination. For example, two moving points following a *zig-zag* pattern can be characterized by a set of taps that respond maximally if the angular mean remains constant and the sign of the angular difference alternates.

During the training of the map, a series of synthesized movements are fed to the first level and then propagated up to the third level. Figure 11 shows the result of the learning process where, for each input example, the most resonant neuron in the third level is highlighted.

The network is tested using a natural movement acquired by recording the motion of a mouse in a graphics window. The movement represents a *zig-zag* coordination that should be recognized by the two right extreme elements of the map (refer to Figure 11).

Figures 12 to 14 display six different snapshots of the model behavior. Each snapshot shows the activation of the first level at the end of each input sampling pair (the current sampling positions in the input trajectory are shown at the top of each snapshot) and the corresponding activation of the third level. The maps are represented by a grid of $x's$ each of which identifies an element of the map. The membrane potential is represented by a grey bowl around the x (the wider the bowl, the higher the activity of corresponding element). A black bowl represents the firing instant. The filter associated with each firing element of the first level is shown on the right. The sampling is delayed of a time interval $t = 2.0$. Figure 15 shows the full sequence of impulses and for each impulse at the second level, the activation state of the third level. The pair of impulses outlined by segmented ovals underline the snapshot displayed in Figures 12 to 14.

At the third level observe that, the two elements corresponding to the *zig-zag* patterns fire at two different time instants: $t = 182.0$ and $t = 240.0$.

6 Phase filtering by temporal neuron locking

One of the main characteristics of the proposed model is its ability to transform a sequence of impulses encoding some basis patterns into a sequence of impulses encoding a more complex pattern and at the same time, maintaining a bi-univocal correspondence between the inducing and induced impulses.

The temporal properties of the map elements enable the system to maintain correct bindings between the input and output impulses: in fact, even though each element detects an input temporal pattern at the end of its occurrence, the firing of each tap in the element is synchronized with the occurrence of the appropriate sub-event that make up the input temporal pattern.

In some cases, it could be important to implement phase filtering on the input impulses. For example, in the simulation described in the previous section, as soon as one of the taps of an element in the third level matches one of the input impulse corresponding to the initial component of an input coordination pattern, the element should restrict its attention only to that pattern and events that are continuations of that pattern. Other events (for example a new moving point sampled with a new phase) should not interfere with its computation. In this way, the firing of the element highlights that the element matched a complex event

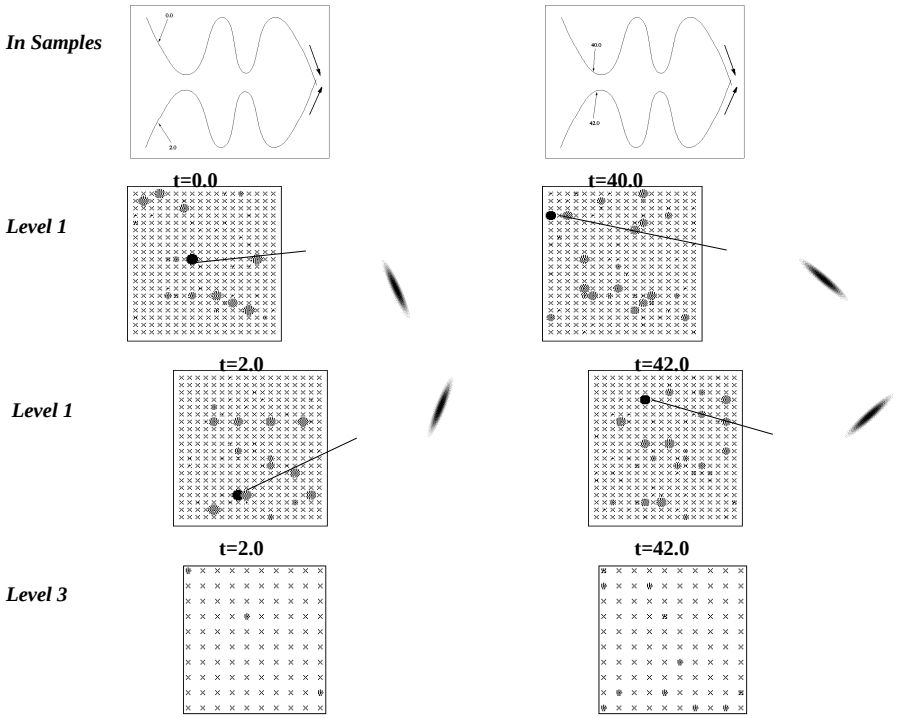

Figure 12:

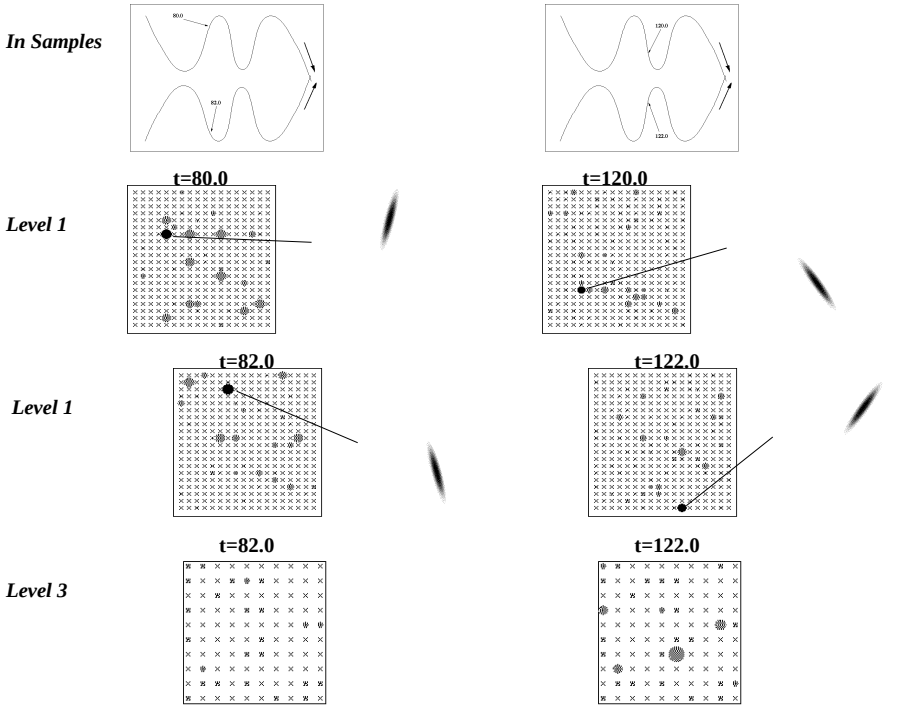

Figure 13:

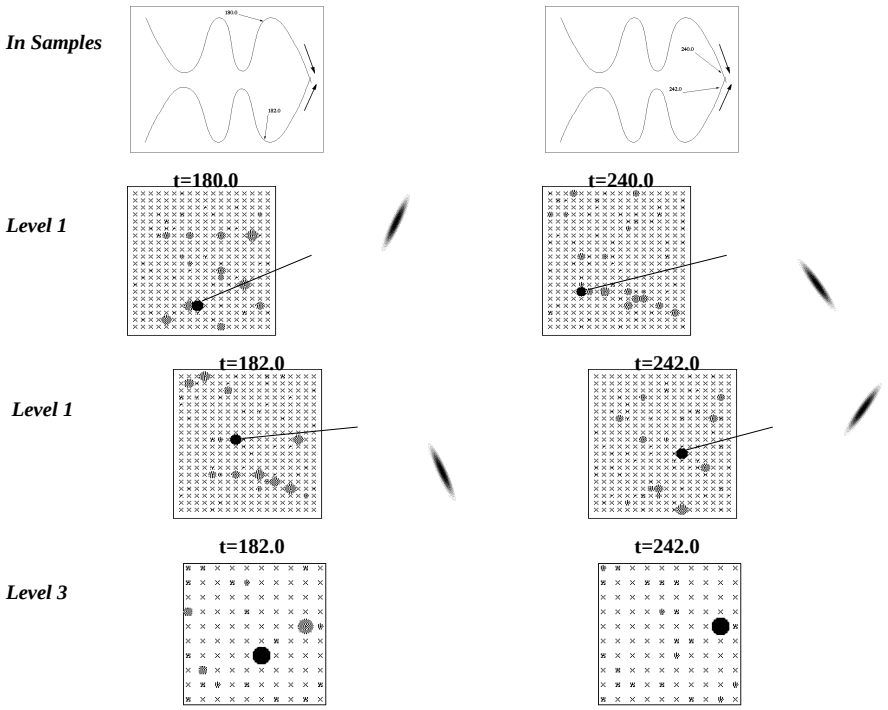

Figure 14:

whose initial component matched the first tap and subsequent components matched the remaining taps.

This *neuron locking* mechanism can be achieved by taking advantage of the nature of the input function. In fact, since each input event is identified by a specific phase (in the simulation the two phases were $\varphi_1 = 0.0$ and $\varphi_2 = 2.0$), all the induced impulses in the upper levels are regularly displaced in time. For example, the two input trajectories in the first level are combined into unique output impulse in the second level that always occurs at some multiple of the first time instant $t = 2.0$. The involved elements in the third level can consequently restrict their attention around the multiples of $t = 2.0$ and disregard all other incoming impulses.

This selective mechanism can be implemented simply by introducing a third general element in the map (see Figure 16) called *time unit*: it corresponds to a generic RC element which is connected to the input of the system and to all the elements of the third level. If the very first input impulse in the first level activates the time unit then, the consequent RC decaying activation represents an internal relative clock of the map which can be used to provide a reference time to select specific time instants. In this way, each element in the third level can *memorize* (for example using an internal unit memory S_i) the time-unit value when the first tap matched, subsequently, an input impulse will be taken into consideration only if the value of the internal memory S_i equals the time-unit activation.

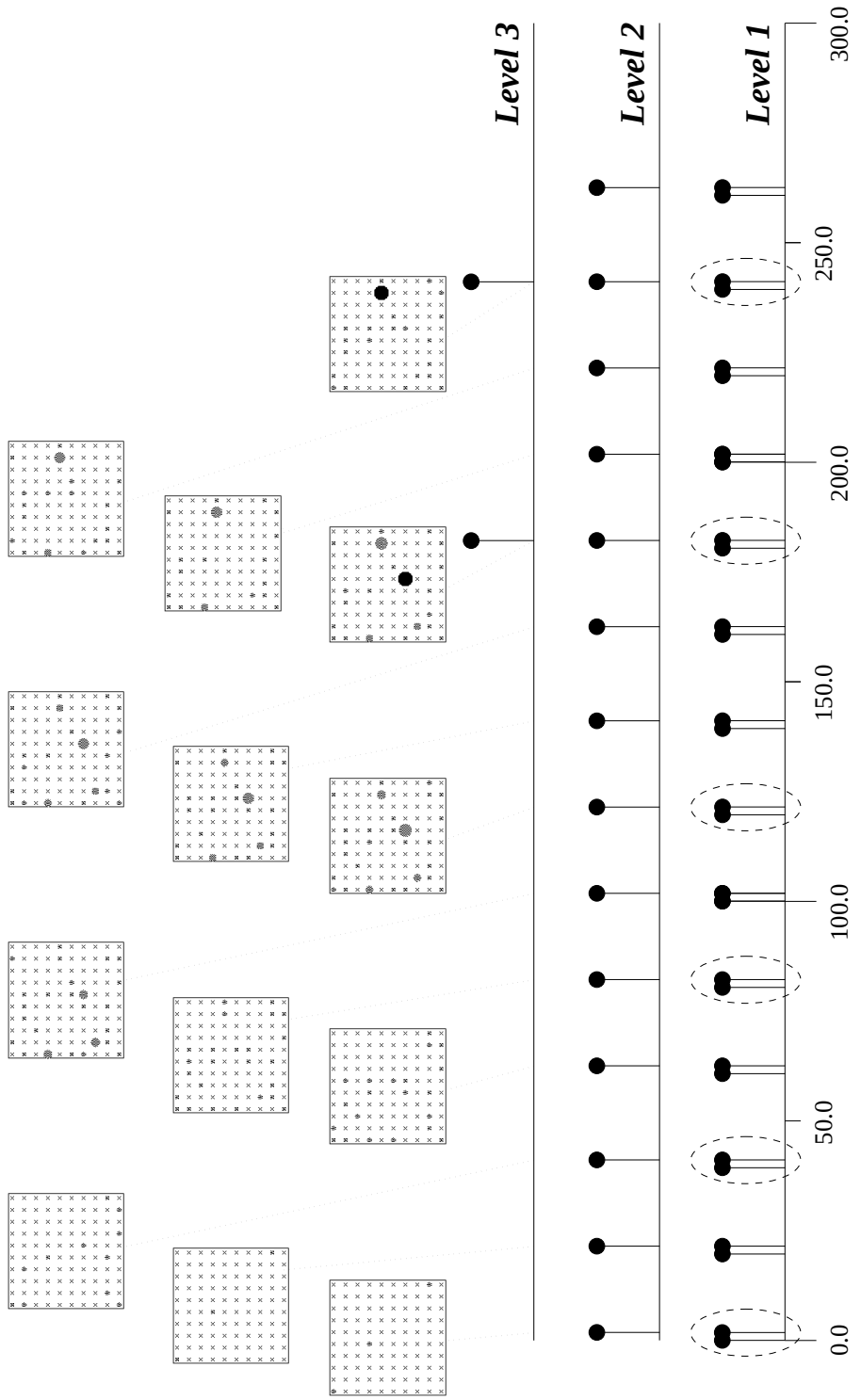

Figure 15: *Sequences of impulses in the three different levels: for each of the second level impulses, the corresponding activity in the third level is displayed. The dotted ovals in the first level outline the time instants already taken into consideration in the previous figures.*

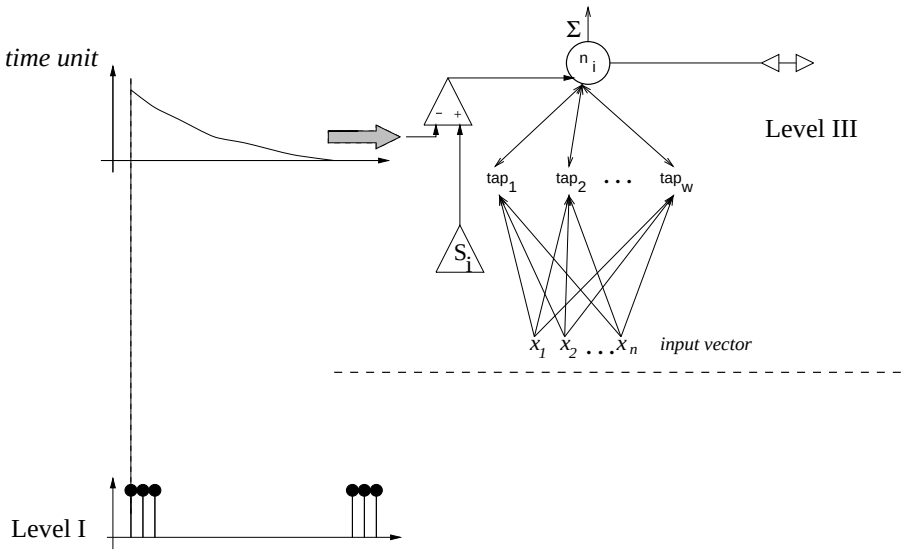

Figure 16: *Lock neuron mechanism.*

7 Conclusions

The ability of DSOM to process temporal patterns has been used to implement a compositional processing model where primitive descriptions of time varying situations can be processed in a hierarchical manner to produce progressively abstract and structured descriptions. The model represents information as a sequence of impulses. It can analyze an input consisting of time varying sequence of impulses and generate an output sequence of impulses, wherein each impulse represents the occurrence of certain patterns in the input sequence.

Enlarging the hierarchy simply involves the training of a new map with the output impulsive function of the corresponding lower map so that finally, knowledge is built up by a hierarchical bottom-up strategy.

The proposed paradigm has been tested in general situations where multiple moving dots in an input visual image generate different classes of events at each level in the hierarchy. In future work, we plan to investigate the use of DSOM structures to address additional problems involving compositionality that arise in language, handwriting recognition, and rapid reasoning.

References

[1] M. Benaim and L. Tomasini. Competitive and Self-Organizing algorithms based on the minimization of an information criterion. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, Amsterdam, 1991. North-Holland.

[2] N. Bernstein. *The Co-ordination and Regulation of Movements.* Pergamon, Oxford, 1967.

[3] E. Bienenstock. A model of neocortex. *Netwrok: Computation in Neural Systems*, 6:179–224, May 1995.

[4] E. Bienenstock and S. Geman. Compositionality in neural systems. In M.A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 223–226. The Mit Press, 1995.

[5] I. Bierderman. Recognition-by-components: a theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[6] S. L. Blesser, R. Coppola, and R. N. Nakamura. Episodic multiregional cortical coherence at multiple frequencies during visual task performance. *Nature*, 366:153–155, 1993.

[7] D.C. Burr and J. Ross. Visual analysis during motion. In M.A. Arbib and A.R. Hanson, editors, *Vision, Brain, and Cooperative Computation*, pages 187–208. MIT Press, 1987.

[8] D.D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18:65–96, 1985.

[9] S.W. Keele, A. Cohen, and R. Ivry. Motor programs: concepts and issues. In M. Jeannerod, editor, *Attention and performance XIII*. Lawrence Erlbaum Ass., Hillsdale, NJ,, 1990.

[10] T. Kohonen. Self organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[11] T. Kohonen. *Self organization and associative memory*. Springer-Verlag, Berlin, 1988.

[12] T. Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In S. Gielen and B. Kappen, editors, *Proc. of the International Conference on ANN'93*, pages 427–434, Amsterdam, 1993. Springer-Verlag.

[13] T. Martinetz and K. Schulten. A 'Neural-Gas' network learns topologies. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, Amsterdam, 1991. North-Holland.

[14] T. Pavlidis. Structural pattern recognition: primitives and juxtaposition. In S. Watanabe, editor, *Frontiers of Pattern Recognition*, pages 421–451. Academic Press, New York, 1972.

[15] R. Plamondon and C.M. Privitera. A neural model for learning and generating rapid movement sequence. *Biological Cybernetics*, 34(2):117–30, 1996.

[16] T. Poggio and F. Girosi. Networks for approximation and learning. *Proc. of the IEEE*, 78:1481–1495, 1990.

[17] T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

[18] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation*. Clarendon Press, Oxford, 1987.

[19] C.M. Privitera and P. Morasso. The analysis of continuous temporal sequences by a map of sequential leaky integrators. In *Proceedings of IEEE International Conference on Neural Networks*, volume 5, pages 3127–3130, Orlando, Florida, 1994.

[20] C.M. Privitera and P. Morasso. A neural model for the execution of symbolic motor programs. In M. Marinaro and P. Morasso, editors, *International Conference on Artficial Neural Networks*, pages 254–257, London, 1994. Springer-Verlag.

[21] C.M. Privitera and R. Plamondon. A self-organizing neural network for learning and generating sequences of target-directed movements in the context of a delta-lognormal theory. In *Proc. IEEE Int. Conf. on Neural Networks IEEE ICNN'95 Perth, Australia*, volume 4, pages 1999–2004, Perth, Australia, 1995.

[22] C.M. Privitera, V. Sanguineti, and P. Morasso. Temporal sequences generation and computational maps. In *Proceedings of NeuroNimes'93: Neural Networks and their Industrial and Cognitive Applications*, pages 55–64, Nimes, France, 1993.

[23] M. Reiss and J.G. Taylor. Storing temporal sequences. *Neural Networks*, 4:773–787, 1991.

[24] H. Ritter. Learning with the self-organizing map. In T. Kohonen, K. Makisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, Amsterdam, 1991. North-Holland.

[25] H. Ritter and K. Schulten. Extending kohonen's self-organizing mapping algorithm to learn ballistic movements. In R. Eckmiller and C. Malsburg, editors, *Neural Computers*, pages 393–406, Berlin, 1987. Springer-Verlag.

[26] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and brain sciences*, 16:417–494, 1993.

[27] L. Shastri and D. Grannes. A connectionist treatment of negation and inconsistency. In *Proc. of Cognitive Science Society Conference (to appear)*, San Diego, July 1996.

[28] W. Singer. Synchronization of cortical activity and its putative role in information processing and learning. *Annu. Rev. Physiol.*, 55:349–74, 1993.

[29] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[30] C. von der Malsburg. Am i thinking assemblies? *Brain Theory*, 1986.

[31] D.L. Wang and M.A. Arbib. Timing and chunking in processing temporal order. *IEEE Trans. on System, Man, and Cybernetics*, 23(4):993–1009, 1993.