A Topological and Temporal Correlator Network for Spatiotemporal Pattern Learning, Recognition, and Recall

Narayan Srinivasa, *Member, IEEE,* and Narendra Ahuja, *Fellow, IEEE*

***Abstract*—In this paper, we describe the design of an artificial neural network for spatiotemporal pattern recognition and recall. This network has a five-layered architecture and operates in two modes: *pattern learning and recognition* mode, and *pattern recall* mode. In pattern learning and recognition mode, the network extracts a set of topologically and temporally correlated features from each spatiotemporal input pattern based on a variation of Kohonen's self-organizing maps. These features are then used to classify the input into categories based on the fuzzy ART network. In the pattern recall mode, the network can reconstruct any of the learned categories when the appropriate category node is excited or probed. The network performance was evaluated via computer simulations of time-varying, two-dimensional and three-dimensional data. The results show that the network is capable of both recognition and recall of spatiotemporal data in an on-line and self-organized fashion. The network can also classify repeated events in the spatiotemporal input and is robust to noise in the input such as distortions in the spatial and temporal content.**

***Index Terms*— Neural networks, pattern recognition, pattern recall, self-organizing maps, spatiotemporal patterns.**

I. INTRODUCTION

RECOGNITION of spatiotemporal patterns is an important and emerging area in the field of pattern recognition. This is primarily because several application areas such as speech processing [19], [24], video image processing [5], sonar [6], [11], and radar [11], [27] signal processing require systems that are capable of processsing signals that are variant in *both* space and time. In the recent past, there has been a keen interest in using neural-network-based approaches to address this important problem. This is because neural networks in general incorporate learning algorithms that are capable of adapting to the presented data as opposed to algorithmic approaches that are usually based on preprogrammed routines. This capability makes learning more flexible than algorithmic approaches.

There are two primitive approaches to learning spatiotemporal patterns using neural networks. The first approach is to convert the spatiotemporal signal into a spatial signal by ignoring the temporal component and treating the entire signal as a spatial pattern. These spatial patterns can be learned by conventional neural networks such as *backpropagation* [22] and *radial basis functions* [16] or networks with preprocessed inputs using tapped delay lines called *time-delayed neural networks* [23], [2]. These networks belong to a class of neural networks called the *feedforward* network architectures. Here the inputs to a node/neuron at a layer is propogated forward to the nodes of the next layer. The corresponding outputs are typically *fixed points* in the output feature space and thus can be used for the classification of spatiotemporal input patterns. There are two disadvantages in using these neural networks. The first is that the training time is long when solving large scale problems (this is true in particular for backpropagation algorithms). The other disadvantage is that it is necessary to retrain the entire network when a new pattern is added. This problem was referred to by Grossberg as the *stability/plasticity dilemma* [9].

The second approach to learn spatiotemporal patterns is using *recurrent* neural networks. In these networks, forward, backward, and self-loop connections are permissible. A key consequence of these connections is that dynamical behaviors not possible with strictly feedforward networks, such as cyclic outputs (or *limit cycles*) and *chaos*, can be produced with recurrent networks. Recurrent networks can be divided into two main types based on the symmetry of the connections between neurons in the network. The first type is recurrent networks with symmetric connections (such as the Hopfield networks [13]) that always converge to a stable state. The other type of recurrent network have assymmetric weight connections. Assymetric recurrent networks [1], [7], [8], [12], [14], [17], [18], [21], [25], [26], in general, lead to spatiotemporal behavior, such as limit cycles rather than a single stable state. An advantage of recurrent neural networks is that smaller networks may provide the functionality of much larger feedforward networks. However, the stability/plasticity problem faced by feedforward networks also affects recurrent networks. In addition, learning a large set of spatiotemporal patterns prohibitively increases the training time. These networks also do not scale well when a large number of spatiotemporal patterns have to be learned.

We present a topological and temporal correlator network (TOTEM) that is capable of learning and recognizing an arbitrary number of spatiotemporal input patterns in a self-organized fashion without suffering from the stability/plasticity problem. TOTEM is capable of recall or reconstruction of each learned pattern with high fidelity. The learning can be

Manuscript received December 1, 1997; revised December 1, 1998.

N. Srinivasa is with HRL Laboratories, Malibu, CA 90265 USA.

N. Ahuja is with the Beckman Institute and the Department of Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801 USA.

Publisher Item Identifier S 1045-9227(99)02200-6.

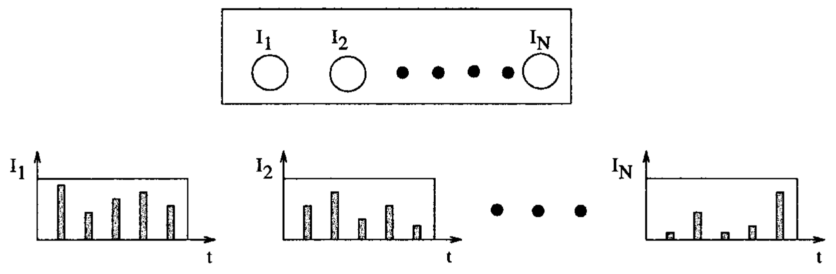


Fig. 1. Feature detectors at the input layer.

performed in an on-line fashion. The network is capable of handling repeated occurances of a fixed spatial location in a spatiotemporal pattern (also referred to as a repeated event) and is fairly robust to noise in both the spatial and temporal content of the pattern. This paper is organized as follows. In Section II, we define the nature and scope of the learning task. An overview of the TOTEM network is provided in Section III. In Section IV, the five-layered network architecture and the learning algorithm is provided. The performance of the TOTEM network is then evaluated through computer simulations in Section V. In Section VI, issues related to the network performance are discussed. Concluding remarks are provided in Section VII.

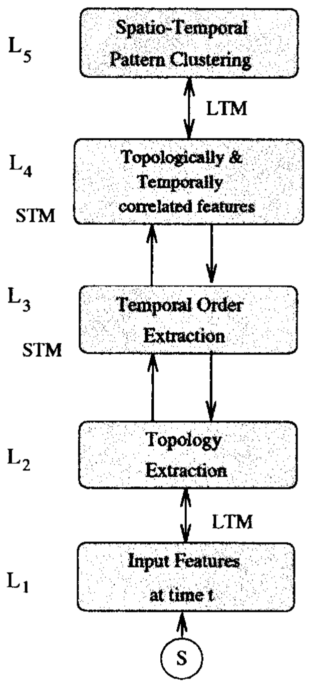
II. Problem Definition

Consider a set of N feature detectors $(I_1, \cdots, I_N)$ as an input layer of a network as shown in Fig. 1. The spatial content of the spatiotemporal pattern is registered as the amplitude $(|I_i|)$ of the signal while the temporal content is registered by the change in these amplitudes at various time instants. Let us assume that these detectors are activated by a spatiotemporal signal $(I_1(k), \cdots, I_N(k))$ at any time k. Further, let us assume that the activity at these detectors is not affected by its activity at previous time t $(t < k)$. The objective is to recognize the spatiotemporal activity pattern that impinges on these feature detectors in a self-organized manner. In other words, the goal is to design a learning architecture that can use the short-term spatiotemporal activity across all the feature detectors to extract and classify them into spatiotemporal categories without any external supervision. Further, the network must also be able to recall the learned categories with high fidelity.

In this paper, we will study this design problem with the following constraints imposed on the characteristics of the spatiotemporal signal.

- The spatiotemporal signals are presegmented. This implies that the beginning and end of a signal are clearly marked or provided.
- The spatiotemporal signals belonging to the same category are approximately of the same duration and speed.
- The spatiotemporal patterns belonging to the same category have approximately the same number of sample points.

Under these constraints, the goal is to design a network that is able to recognize and recall arbitrary number of spatiotemporal patterns even in the presence of noise. In addition, the network must also be capable of handling repeated events within the spatiotemporal pattern. In terms of feature values, a repeated event implies that the activity $(I_1(t), \cdots, I_N(t))$ across all the feature detectors at a given time t is the same as that at a previous time $t - \Delta t$.



L_5
Spatio-Temporal Pattern Clustering
LTM
L_4
STM
Topologically & Temporally correlated features
L_3
STM
Temporal Order Extraction
L_2
Topology Extraction
LTM
L_1
Input Features at time t
S

Fig. 2. Overview of the TOTEM architecture.

III. Overview of TOTEM

The TOTEM contains five layers (L_1 through L_5) as shown in Fig. 2. Let us assume that the spatiotemporal input in the form of a two-dimensional (2-D) trajectory $(x(t), y(t))$ for $t = 1, \cdots N$ is being processed by the TOTEM network. The coordinates (x, y) represent the features of the input pattern that span R^2 (i.e., set of real numbers in 2-D). Before any processing, an off-line preprocessing step is required to organize the input features into clusters based on Kohonen's self-organizing maps (SOM) [15]. During the clustering process, input features (x, y) (uniformly sampled in R^2 space) are presented to the L_1 layer. Each input sample selects a winner among a 2-D lattice of nodes in L_2 that best matches (i.e., minimum Euclidean distance) the state of the winner node. The state of each node in L_2 is defined by its synaptic strength/weight (also referred to as the long-term memory (LTM)). Initially, these weights are assigned randomly. After a winner is selected, the weights of the winner and its neighbors are updated such that the Euclidean distance between the input sample and the weights is decreased (also called *learning*). After many input samples have been presented to the L_2 layer, its nodes become organized such that neighboring nodes in the L_2 layer correspond to neighboring inputs in R^2 space. Thus, the nodes of L_2 define a topology-preserving map of the input. This completes the preprocessing step. The network then begins the processing of spatiotemporal inputs during which it can operate in two modes: the *pattern learning and recogniton* mode (or the PLR mode) and the *pattern recall* mode (or the PR mode). We will first outline the working principle during the PLR mode and then the PR mode.

A. The PLR Mode

The input $(x(t), y(t))$ is sequentially presented to L_1 for $t = 1, \cdots, N$. The input at each time step is classified into a node (winner node) in the 2-D lattice of nodes in L_2 based on the minimum Euclidean distance between the input features and its states defined by its weights. The winner node at each time step excites the nodes in the L_3 layer. The L_3 layer consists of a lattice of nodes with a one-to-one connection to nodes in L_2. Thus, an L_2 layer winner node at any time t excites a node in the same lattice position as L_2.

The magnitude of response (or the short-term memory (STM) activity) of the excited nodes (for $t = 1, \cdots, N$) in L_3 is inversely proportional to time t at which it was excited. Thus, if the activity of an L_3 node corresponding to an input $(x(t), y(t))$ is denoted by A_t, then $A_1 > A_2 > \cdots > A_{N-1} > A_N$ for $t = 1, \cdots, N$. This property thus enables the L_3 layer to store the temporal order in which the input features $(x(t), y(t))$ were presented to the L_1 layer. At the end of input presentation (i.e., $t = N$), the activity of L_3 nodes is propagated to L_4 layer. This process creates an input vector to L_4 that consists of the lattice positions of L_3 nodes juxtaposed in decreasing order of L_3 node activity (this process is referred to as the *sequential uploading* process in Section IV-B3). The L_4 input vector (or its STM activity) is then classified into categories in L_5 using the fuzzy-ART learning algorithm [4]. Each category in L_5 represents a class of similar spatiotemporal input patterns. The template for each learned class is stored in the weights (or the LTM) connecting the input vector at L_4 to the category node at L_5.

An interesting aspect of the TOTEM network is that the length of the input vector in L_4 is always twice the number of time steps (i.e., $2N$) at which the spatiotemporal input was sampled but not on the number of features in it. This is because the input features at any given time are stored in an L_3 node whose 2-D position is then stored in L_4 in a temporal order of occcurance. The input vector at L_4 is far more robust to noise such as distortions in input features and their temporal occurance than the original input. This is because the input undergoes two transformations before reaching the L_4 layer. The first transformation is at L_2 where the input is compressed into a 2-D lattice of neurons. This compression creates robustness to small distortions (due to noise) in features of the input because input features of two similar input patterns at a given time step will activate the same lattice node or a topologically close neighbor despite distortions. This results in a very minimal change in the lattice position of the excited L_3 node and thus translates into minimal activity change for the input vector at L_4 after sequential uploading. The second transformation occurs at L_3 where the temporal order is extracted. This transformation provides robustness to error in time. This is because features of two similar input patterns that occurred at approximately the same time would result in very similar magnitudes of response in L_3. Thus, the input vector at L_4 (extracted during the sequential uploading process) will not be affected.

B. PR Mode

The PR mode is activated only if the L_5 layer contains learned classes of spatiotemporal input patterns. In this mode, there is no input at the L_1 layer. Instead, a category node in the L_5 layer is excited. This excitation results in the activation of all the nodes in L_4. The magnitude of response (or activity) for each L_4 node is equal to the weight connecting it to the excited L_5 node. The L_4 node activities provides both the temporal order in which the lattice positions of L_3 should be excited. The resulting sequence of excitation of the nodes in the L_3 layer is then transferred directly to the L_2 layer by the one-to-one connections that exist between the L_3 and L_2 layers. Finally, the activity of the L_2 nodes is utilized to recall the spatiotemporal pattern by using the learned weights between L_2 and L_1 layers.

IV. Network Architecture and Dynamics

We will first provide the network dynamics for the preprocessor stage based on the SOM. This will be followed by the details of the network architecture and network dynamics during the PLR mode. Finally, the network dynamics during PR mode will be provided.

A. Preprocessor for Topology Extraction

The preprocesor is based on the Kohonen's SOM [15]. Initially, spatiotemporal patterns that have very little or no temporal order is presented to the L_1 layer. The feature detectors at the L_1 layer are fully connected to a 2-D lattice of neurons in L_2 layer via the weight vector $\mathbf{W}_{ij}$ (where (i, j) is the lattice location). These weights are initialized to random and small values. The input vector at each time instant $\mathbf{I}(t)$ is then used to select a winner latice neuron (i^*, j^*) in L_2 whose weight $\mathbf{W}_{i^*j^*}$ has the least Euclidean distance to $\mathbf{I}(t)$ as shown in Fig. 3. In this figure, the input is shown as a point in the input feature space for clarity purposes.

The weight vector for the winner and its neighbors are then updated in L_2 as

$$\Delta\mathbf{W}_{ij} = \alpha h_{i^*j^*}(t)[\mathbf{I}(t) - \mathbf{W}_{ij}(t)] \tag{1}$$

where $h_{i^*j^*}(t)$ is the Gaussian weighting function for the weight update. This function modifies the weights for the winner and its neighboring lattice neurons in proportion to their Euclidean distance from the winner (i^*, j^*). This function

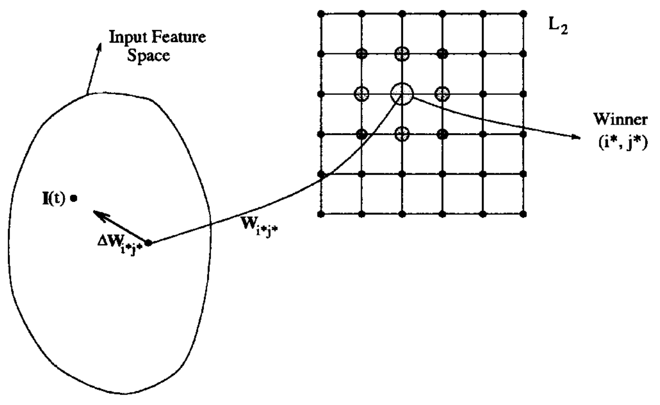


Fig. 3. Preprocessing using the Kohonen self-organizing map.

is defined as $h_{i^*j^*}(t) = e^{\frac{-([i-i^*]^2+[j-j^*]^2)}{2\sigma^2}}$ where σ defines the size of the neighborhood for the weight update. The Gaussian weighting update process is illustrated in Fig. 3. The amount of weight update for the winner and the neighboring lattice neurons is depicted by the size of neurons in Fig. 3. This update essentially shifts the weight vector $\mathbf{W}_{i^*j^*}(t)$ toward the input $\mathbf{I}(t)$. The parameter α controls the rate of weight update and is called the learning rate.

After each weight update, the parameters σ and α are decreased by small amounts before the spatiotemporal input at the next time step is processed. This is necessary for the network to asymptotically reach a stable state [15]. An appropriate choice for the rate of decrease of α and σ with each weight update is important for good results and rapid convergence (the choice for decreasing these parameters in all our simulations will be provided in Section V). This training process is continued until $\Delta\mathbf{W}_{ij}$ is smaller than a user defined threshold for all lattice neurons (i, j). If the exact topology of the input space is known *a priori*, then this threshold for termination of learning can be set in terms of faulty connections between neurons in the lattice as follows. For a 2-D lattice, it is straightforward to show that $4n(n-1)$ connections exist between its lattice neurons (as shown in Fig. 3). The term n corresponds to the total number of neurons per dimension. So, the learning can be terminated when the total number of faulty connnections (measured by the Euclidean distance between the weights of neighboring lattice neurons) is below a user defined threshold. At the end of the training process, the projection of the weight vector $\mathbf{W}_{ij}$ onto the input feature space maintains neighborhood relationships in that space. Therefore, the SOM is also called as *topology preserving* map [20]. After training is completed, the final weights $\mathbf{W}_{ij}$ represent the LTM between the L_1 and L_2 layers (refer to Fig. 2) and store the topology information of the input feature space.

B. Network Dynamics for PLR Mode

Once the topology of the input feature space is extracted using SOM, the TOTEM is ready for learning and recognition of spatiotemporal patterns. We now provide a detailed exposure to the network architecture and dynamics during the PLR mode.

1) $L_1 - L_2$ Dynamics: The spatiotemporal inputs at each time instant are presented to the feature detectors of the L_1 layer as an input $\mathbf{I}(t)$. This input activates a lattice neuron (i^*, j^*) in the L_2 layer that is closest to $\mathbf{I}(t)$ by Euclidean distance. The frequency of the winner $f_{i^*j^*}$ is then updated as

$$f_{i^*j^*} = f_{i^*j^*} + 1 \tag{2}$$

Initially, $f_{ij} = 0$ for all (i, j) in the 2-D lattice of the L_2 layer. The frequency information is useful for detecting if any lattice neuron has been excited repeatedly (i.e., repeated event) during the processing of a spatiotemporal trajectory. This information is exploited by the L_3 layer as described in Section IV-B2.

In Fig. 4, a sequence of lattice neuron activations is shown during the course of an example spatiotemporal input trajectory presentation. It can be seen that at any given time, only one neuron is active in the L_2 layer. This is exactly how the SOM [15] is designed to operate once the learning of topology has been completed. This means that the activity of all previous time steps is lost at the current time step. To prevent this, the activity of a neuron in the L_2 layer at any given time step is transferred immediately to the L_3 layer. It should be noted that the termination of a spatiotemporal trajectory is provided via a gate S (refer to Fig. 2) to the L_1 layer. This gate is set by the user such that $S = 1.0$ when the input has reached its end and is zero otherwise. If the input is terminated at L_1 (i.e., $S = 1.0$), then no neurons are excited in the L_2 layer.

2) $L_2 - L_3$ Dynamics: The neurons in the L_3 layer are also organized in the form of a 2-D lattice much like the L_2 layer. In fact, at the beginning of each spatiotemporal input presentation, the L_3 layer contains exactly the same number of lattice neurons as the L_2 layer. There exists a one-to-one correspondence in lattice position between the neurons of L_2 and L_3 layers and they are connected by one-to-one pathways as shown in Fig. 5 (all pathways are not shown).

The winner neuron (i^*, j^*) in the L_2 layer at each time step t sends an excitatory signal via the one-to-one pathways to activate the corresponding lattice neuron in L_3 layer. In addition to this excitatory input, the winner neuron (i^*, j^*) in L_3 also receives inhibitory inputs (not shown in Fig. 5) from all the nonwinners within L_3. An interesting aspect of the L_3 layer is that the number of neurons at a given lattice location can be increased in a dynamic manner. This feature is required to handle repeated events in a spatiotemporal input. If a spatiotemporal input at the current time t occurred at a previous time step $t - \Delta t$, then the winner lattice neuron corresponding to $t - \Delta t$ would once again win at the current time. In this case, the winner neuron would have a frequency $f_{i^*j^*} = 2$. When the frequency of any lattice neuron (i, j) in L_2 increases by one, a new lattice neuron is recruited at the corresponding lattice location (i, j) in L_3. This recruitment is accomodated in our notation by using a third index k for each lattice position in L_3 where $k = f_{i^*j^*}$.

The activity of the winner neuron, $x_{i^*j^*k}(t)$, can be modeled using the shunting equations of Grossberg [10] as

$$\frac{dx_{i^*j^*k}(t)}{dt} = -B_1 x_{i^*j^*k}(t) + (A_1 - x_{i^*j^*k}(t))J_{i^*j^*k} - (C + x_{i^*j^*k}(t))X \tag{3}$$

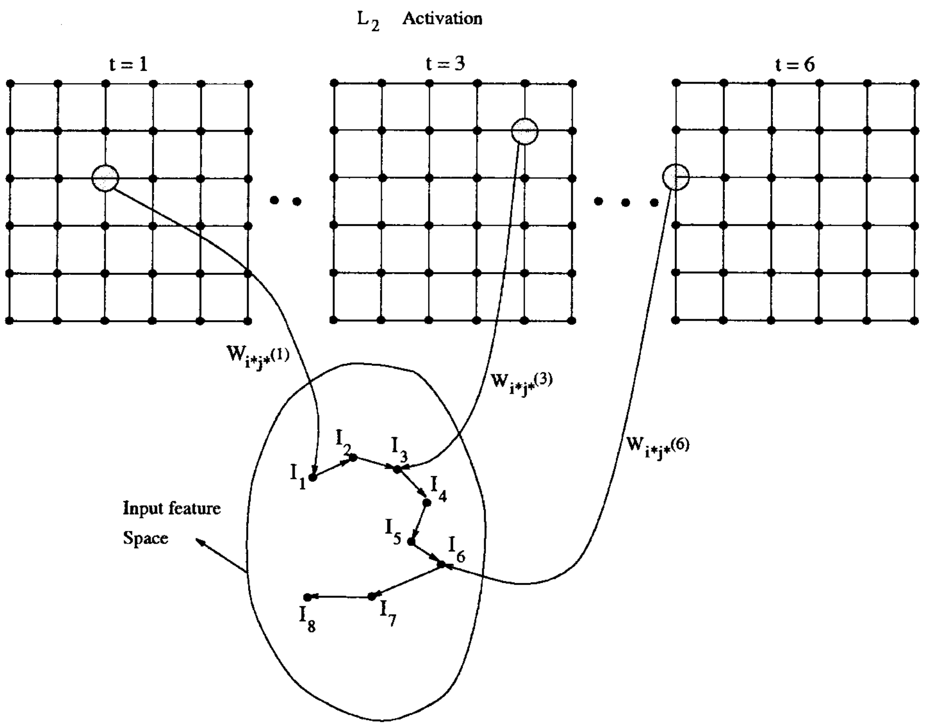


Fig. 4. Activation of lattice neurons in L_2 layer during spatiotemporal PLR mode.

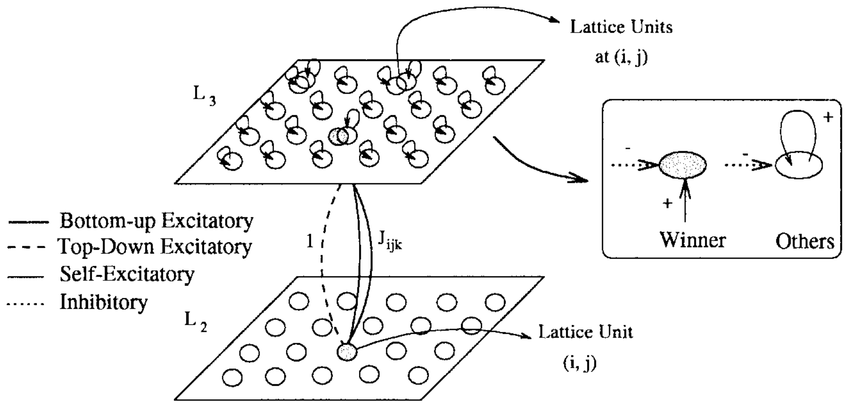


Fig. 5. The network architecture at the L_2 and L_3 layers.

where $J_{i^*j^*k} = \gamma$ (γ is an attenuation constant and is <1) is the strength of the one-to-one excitatory pathways between the winner neurons in the L_2 and L_3 layers at time t. The term X denotes the net inhibition at the winner received from all the nonwinners at time t and is given by $X = \sum_{(i,j)\neq(i^*,j^*)} x_{ijk}(t)$. In (3), the excitatory inputs drive $x_{i^*j^*k}(t)$ toward a finite maximum while the inhibitory inputs drives it toward a finite minimum. In particular, the value of $x_{i^*j^*k}(t)$ in (3) remains bounded within the range $(-C, A_1)$ and decays to the resting level zero in the absence of any inputs.

The nonwinning neurons in the L_3 layer do not receive any excitatory input from the L_2 layer. Instead, they self-excite using their activity, $x_{ijk}(t-1)$, at the previous time step. They also receive inhibitory inputs from all the remaining nonwinners within L_3. The winning neuron in L_3 cannot self-excite since it has no activity at previous time steps. The activity of all the nonwinning neurons $x_{ijk}(t)$ can also be modeled using the shunting equations as

$$\begin{aligned}\frac{dx_{ijk}(t)}{dt} &= -B_2 x_{ijk}(t) + (A_2 - x_{ijk}(t))x_{ijk}(t-1)\\ &\quad - (C + x_{ijk}(t))X. \qquad (4)\end{aligned}$$

The activity $x_{ijk}(t-1)$ corresponds to a self-loop connection (refer to Fig. 5) and represents the nonwinning neuron's activity. It should be noted that no neuron in L_3 layer can ever win at two different time steps because of the dynamic recruitment process in L_3.

In this paper, we assume that the integration rate for these two differential equations is extremely fast and thus we can directly obtain the activity of the winning and nonwinning neurons by the setting $\frac{dx_{i^*j^*k}(t)}{dt} = \frac{dx_{ijk}(t)}{dt} = 0$ in (3) and

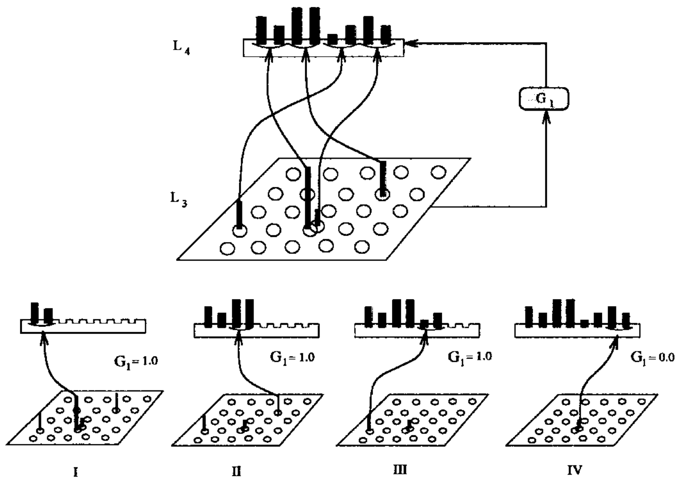


Fig. 6. Sequential uploading of L_3 activity using gate G_1.

(4). Thus, the activity of the winning neuron at L_3 can be algebraically computed by

$$x_{i^*j^*k}(t) = \frac{A_1 J_{i^*j^*k} - CX}{B_1 + X + J_{i^*j^*k}}. \tag{5}$$

Similarly, the activity of the nonwinners in L_3 can be obtained algebraically as

$$x_{ijk}(t) = \frac{A_2 x_{ijk}(t-1) - CX}{B_2 + X + x_{ijk}(t-1)}. \tag{6}$$

By selecting $A_1 \ll 1.0$, $A_2 \ll 1.0$, $B_1 = B_2 \cong 1.0$ and $C = 0$, the activity of winning neurons becomes a monotonically decreasing function of time. The temporal order is then extracted by using this decreasing pattern of activity in the winning neurons. Depending on the choice of constants $[A_1, A_2, B_1, B_2, C]$, the winning neurons can also achieve a monotonically increasing or parabolic pattern of activity as a function of time during the processing of spatiotemporal inputs in the L_3 layer. Among these patterns of activity, only the monotonically decreasing or increasing cases are desirable. This is because for a parabolic case, there can exist two time instants (one in the past and one in the present or future) for which the activities of the winning neuron can be identical and this will destroy the temporal order information in the L_3 layer. In this paper, we selected the constants so as to simulate a decreasing pattern of activity for the winning neurons.

3) $L_3 - L_4$ Dynamics: Unlike the activation of the neurons in L_3 at each time step, the neurons at L_4 are not activated at every time step. The activation of the neurons in L_4 is controlled by gate G_1 (shown in Fig. 6) between L_3 and L_4 layers. Gate G_1 is closed $(G_1 = 0.0)$ if the net activity, $\sum x_{ijk}$ at the L_3 layer has either not reached a steady state or is zero. This can be expressed as

$$\begin{aligned}&\Delta \sum x_{ijk} = \sum x_{ijk}(t) - \sum x_{ijk}(t-1) \neq 0.0 \quad \text{OR}\\ &\sum x_{ijk} = 0.0 => G_1 = 0.0.\end{aligned} \tag{7}$$

This ensures that there is no transfer of activity from L_3 to L_4 during the processing of a spatiotemporal input.

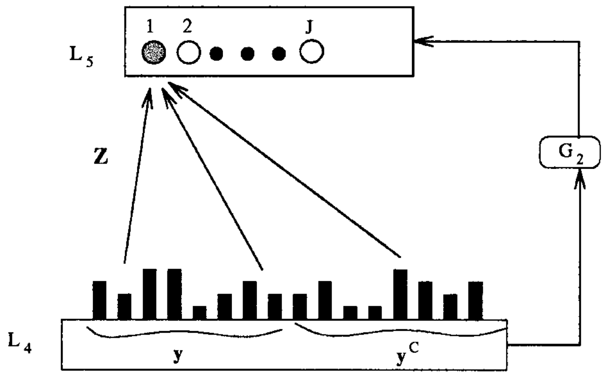


Fig. 7. Coding the features y in K_5 using the fuzzy ART algorithm.

When the input is terminated at L_1 (as signified by $S = 1.0$), the net activity, $\sum x_{ijk}$ at the L_3 layer, reaches a steady state. This causes the gate G_1 to open (i.e., $G_1 = 1.0$). The opening of gate G_1 can also be expressed as

$$\sum x_{ijk} = \text{constant} > 0 \quad \Delta \sum x_{ijk} = 0.0 => G_1 = 1.0. \tag{8}$$

The opening of G_1 initiates a sequential uploading process from L_3 to L_4 as shown in Fig. 6. During this process, the lattice location (i, j) of the neurons are normalized and downloaded to the L_4 layer in an order of decreasing activity x_{ijk} (i.e., the temporal order). The normalized activity of neurons at L_4 can be realized as

$$y_{i1} = \frac{i}{n} \quad y_{i2} = \frac{j}{n} \quad \text{for } i = 1, \cdots M \tag{9}$$

where n is the size of each dimension of the 2-D lattice at L_3 and M is the total number of active neurons in the L_3 layer when G_1 opened for the first time. It should be noted that $\mathbf{y}$ is a $2M$ dimensional vector since for each lattice location there are two components (i, j). After each upload, the activity of the uploaded lattice location in L_3 is set to zero as shown in Fig. 6. When $\sum x_{ijk} = 0.0$, the gate G_1 is closed and the sequential uploading process is complete. The activity of the L_3 neurons is then reset to zero.

4) $L_4 - L_5$ Dynamics: The L_4 layer contains a set of M active nodes whose activity $\mathbf{y}$ is governed by (9). This activity contains the temporal order of excitation of the lattice locations in L_3. Thus, $\mathbf{y}$ represents a set of topologically and temporally correlated features. The $\mathbf{y}$ is then coded into the LTM ($\mathbf{Z}$ in Fig. 7) between the L_4 and L_5 layers using the fuzzy ART algorithm [4]. This coding is initiated by gate G_2 (shown in Fig. 7). The conditions for closing and opening of this gate is governed by the following equations:

$$\begin{aligned}&\Delta \sum \mathbf{y} = \sum \mathbf{y}(t) - \sum \mathbf{y}(t-1) \neq 0.0 \quad \text{OR}\\ &\sum \mathbf{y} = 0.0 => G_1 = 0.0\end{aligned} \tag{10}$$

and

$$\sum \mathbf{y} = \text{constant} > 0 \quad \Delta \sum \mathbf{y} = 0.0 => G_2 = 1.0. \tag{11}$$

The condition in (10) implies that the fuzzy ART coding will not be initiated (i.e., $G_2 = 0.0$) either when the sequential downloading process is taking place (i.e., $\Delta \sum \mathbf{y} \neq 0.0$) or there is no activity in the neurons at L_4. Similarly, the condition in (11) implies that the fuzzy ART coding process is initiated (i.e., $G_2 = 1.0$) when the sequential downnloading process is completed (i.e., $\Delta \sum \mathbf{y} = 0.0$) and the activity of the L_4 neurons reaches a steady state (i.e., $\sum \mathbf{y} = \text{constant} > 0$).

Once the gate G_2 is open, the input $\mathbf{y}$ is coded into the L_5 layer using the fuzzy ART algorithm. The fuzzy ART algorithm [4] is capable of unsupervised classification of both binary and analog inputs in real time as opposed to the original ART-1 algorithm [3] that can handle only binary inputs. The main advantage of this network is that it is very stable to previously stored inputs but at the same time plastic to new inputs. This implies that new classes can be allocated dynamically and the number of classes that can be formed is only limited by the total memory available. Thus, this network provides TOTEM with the capability of handling the stability/plasticity dilemma that was addressed in Section I.

The fuzzy ART network consists of two processing layers (L_4 and L_5 in Fig. 7). The L_4 layer corresponds to the input layer while the L_5 layer corresponds to the category layer. The two layers are fully connected by adaptive weights $\mathbf{Z}$ (fully connected architecture is not shown in Fig. 7 for clarity). The learning algorithm for the fuzzy ART is outlined below. Before any learning takes place, all the components of the weight vector $\mathbf{Z}$ are initialized to 1.0. Then, the input vector $\mathbf{y}$ (of dimension $2M$) and its complement $\mathbf{y}^c$ are stored in L_4 as a $4M$ dimensional input vector $\mathbf{Y} = (\mathbf{y}, \mathbf{y}^c)$ where $\mathbf{y}^c = 1.0 - \mathbf{y}$. This process of storing both the input and its complement is called *complement coding* and helps in preventing a category proliferation problem [4].

The input at L_4 activates the nodes j in L_5 or the category layer as

$$T_j = \frac{|\mathbf{Y} \wedge \mathbf{Z}_j|}{\delta + |\mathbf{Z}_j|} = \frac{\sum_i \min(Y_i, Z_{ji})}{\delta + \sum_i z_{ji}} \tag{12}$$

where δ is a parameter that minimizes recoding [4] and the operator $\wedge$ represents the fuzzy AND operator [28] as defined in the above equation. The definition of the term $|x| = \sum_{i=1}^{4M} x_i$ where M is the total number of features in x. The network then makes a hypothesis by selecting the node J that has the maximum T_j to be the category that stores the presented input. In case of a tie, the node with the least index is chosen.

Then, this hypothesis is tested using a similarity test called the *vigilance criterion* as follows:

$$\frac{|\mathbf{Y} \wedge \mathbf{Z}_J|}{|\mathbf{Y}_J|} \geq \rho \tag{13}$$

where ρ is called the vigilance parameter. The ratio on the left of the above inequality represents the degree of match between $\mathbf{Y}$ and the category J in L_5. If node J satisfies the vigilance criterion, then the weights $\mathbf{Z}_J$ are updated by using the *fast learning rule* [4] as

$$\mathbf{Z}_J = \beta|\mathbf{Z}_J \wedge \mathbf{Y}| + (1-\beta)\mathbf{Z}_J \tag{14}$$

where β (<1.0) is called the forgetting factor. This equation tells us that the weight $\mathbf{Z}$ is modified such that the previous value of the weight is retained by a factor of $(1-\beta)$ while the current input $\mathbf{Y}$ affects the weight by a factor of β.

If node J does not satisfy the vigilance criterion, it is shut down. Then, a search process is initiated in L_5 to see if any other node satisfies the vigilance criterion. If there exists one such node, then its weights are updated using (14). If there exists no nodes in L_5 that satisfy the vigilance criterion, then a new node is recruited to store the spatiotemporal input. Thus, the vector $\mathbf{Y}$ is classified into a category node in the L_5 layer. It should be noted that the LTM weights remain

TABLE I
THE EIGHT GEOMETRIC CURVES USED TO GENERATE THE 2-D SPATIOTEMPORAL TRAJECTORIES

Curve Number	x(t)	y(t)
1	$0.5cos^3\theta(t)$	$0.5sin^3\theta(t)$
2	$0.5(\theta(t) - \sin\theta(t))$	$0.5(1 - \cos\theta(t))$
3	$0.5\cos\theta(t)(1 - \cos\theta(t))$	$0.5\sin\theta(t)(1 - \sin\theta(t))$
4	$0.5\cos\theta(t)(1 + \cos\theta(t))$	$0.5\sin\theta(t)(1 + \sin\theta(t))$
5	$0.5\sin3\theta(t)\cos\theta(t)$	$0.5\sin3\theta(t)\sin\theta(t)$
6	$0.5\sin5\theta(t)\cos\theta(t)$	$0.5\sin5\theta(t)\sin\theta(t)$
7	$0.75\cos\theta(t) - 0.5\cos1.5\theta(t)$	$0.75\sin\theta(t) - 0.5\sin2.5\theta(t)$
8	$1.25\cos\theta(t) - 0.5\cos2.5\theta(t)$	$1.25\sin\theta(t) - 0.5\sin2.5\theta(t)$

TABLE II
THE SIX GEOMETRIC CURVES USED TO GENERATE THE 3-D SPATIOTEMPORAL TRAJECTORIES

Curve Number	x(t)	y(t)	z(t)
1	$0.5cos^3\theta(t)$	$0.5sin^3\theta(t)$	$0.5\cos2\theta(t)$
2	$0.5(\theta(t) - \sin\theta(t))$	$0.5(1 - \cos\theta(t))$	$0.5\cos2\theta(t)$
3	$0.5\cos\theta(t)(1 - \cos\theta(t))$	$0.5\sin\theta(t)(1 - \sin\theta(t))$	$0.5\cos2\theta(t)$
4	$0.5\cos\theta(t)(1 + \cos\theta(t))$	$0.5\sin\theta(t)(1 + \sin\theta(t))$	$0.5\cos2\theta(t)$
5	$0.5\sin5\theta(t)\cos\theta(t)$	$0.5\sin3\theta(t)\sin\theta(t)$	$0.5\cos2\theta(t)$
6	$0.5\sin5\theta(t)\cos\theta(t)$	$0.5\sin5\theta(t)\sin\theta(t)$	$0.5\cos2\theta(t)$

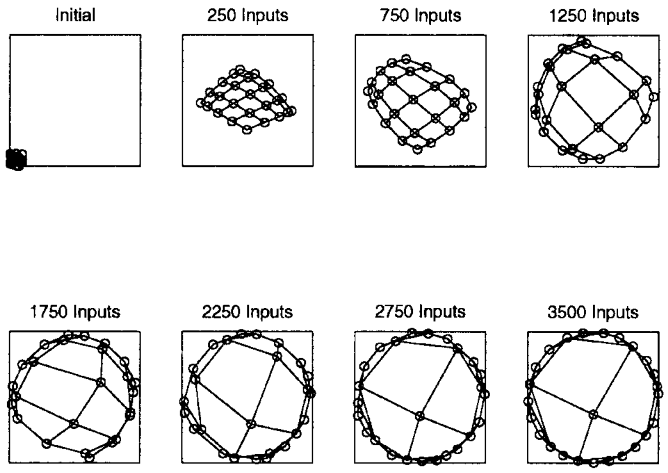
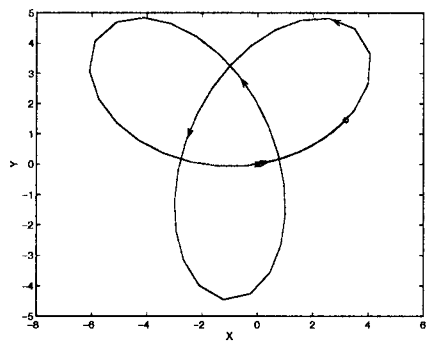
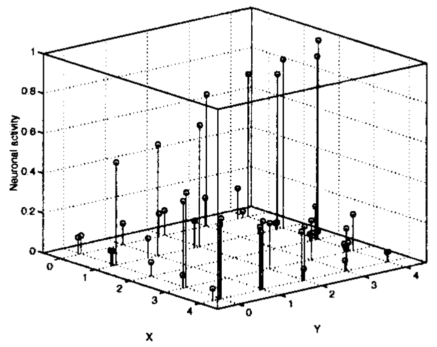


Fig. 8. The projection of W_{ij} onto input space during topology abstraction of the 2-D spatiotemporal data by SOM.


(a)


(b)

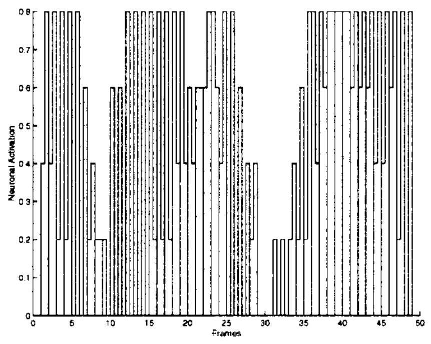

(c)

Fig. 9. Illustration of network dynamics. (a) Noise-free 2-D spatiotemporal trajectory. (b) L_3 level neuron activation at the end of the trajectory presentation. (c) L_4 level neron activation after sequential uppload process.

permanently modified when the next spatiotemporal input is presented. However, the activity $\mathbf{Y}$ at L_4 is also stored in the STM. Also, all the gates are closed before processing the next input.

C. Network Dynamics for PR Mode

In addition to learning and recognition of a given spatiotemporal input, the TOTEM is also capable of pattern recall. We now provide a detailed exposure to the network dynamics during the PR mode of operation.

1) $L_5 - L_4$ Recall Dynamics: In the pattern recall mode, the network does not receive any inputs in the L_1 layer. Instead, the category nodes at the L_5 layer can be probed to recall or reproduce the spatiotemporal patterns that they represent. When a category node in L_5 is probed, then this results in a top-down excitation of the nodes in the L_4 layer via the LTM $\mathbf{Z}$. The activity $\mathbf{Y}$ can be recovered as: $\mathbf{Y} = \mathbf{Z}$.

2) $L_4 - L_3$ Recall Dynamics: Once the activity $\mathbf{Y}$ is recovered, then a sequential download process is initiated between the L_4 and L_3 layers. This process is the inverse of the sequential uploading process described in Section IV-B3. For this process, only the first $2M$ dimensional input $\mathbf{y}$ is used. This vector is then sequentially downloaded in an increasing order of its indices. The downloading process is controlled by a gate G_3 between L_3 and L_4 layers. Gate G_3 is closed $(G_3 = 0.0)$ if the net activity, $\mathbf{y}$ at the L_3 layer, has either not reached a steady state or is zero. This can be expressed as:

$$\Delta \sum \mathbf{y} \neq 0.0 \quad \text{OR} \quad \sum \mathbf{y} = 0.0 => G_3 = 0.0. \tag{15}$$

This ensures that there is no transfer of activity from L_4 to L_3 during the recovery of the $\mathbf{y}$ vector from L_5. Similarly, the condition for the G_3 to be open is given by

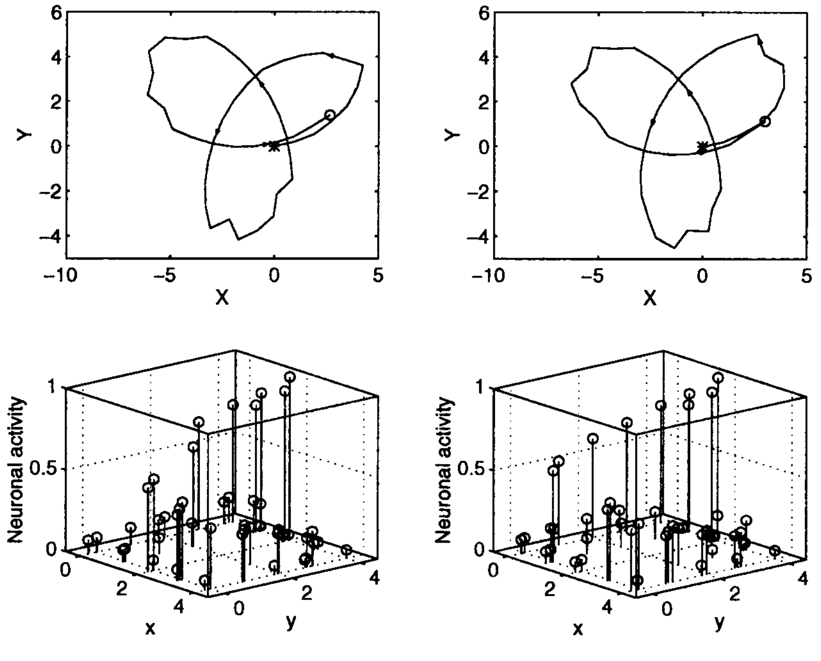
$$\sum \mathbf{y} = \text{constant} > 0, \quad \Delta \sum \mathbf{y} = 0.0 => G_3 = 1.0. \tag{16}$$

The sequence of lattice nodes that are activated during the sequential download process activates the corresponding lattice nodes in L_2.

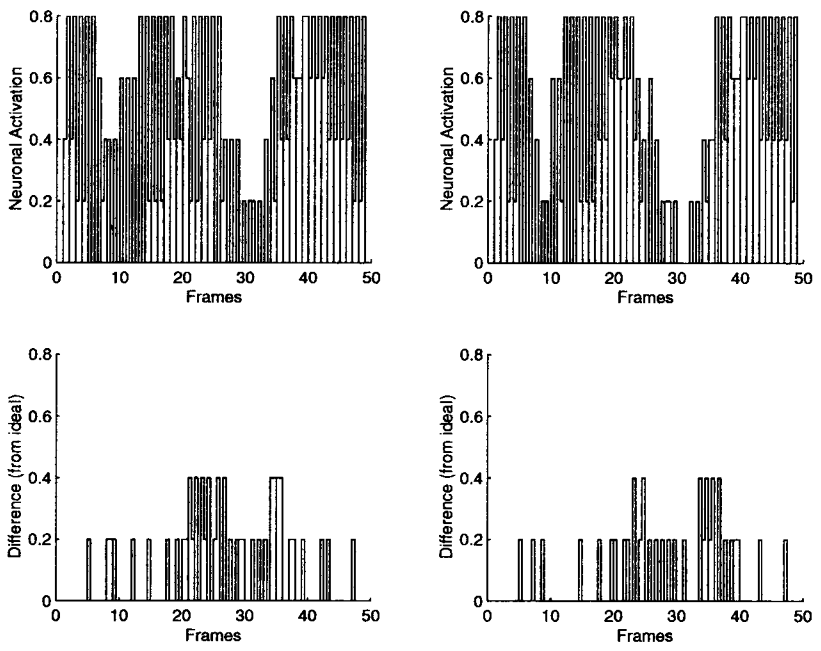
3) $L_2 - L_1$ Recall Dynamics: For each activated lattice neuron (i, j) in L_2, the corresponding chunk of the spatiotemporal pattern is recovered using the LTM $\mathbf{W}_{ij}$ between the L_2 and L_1 layers. The entire spatio-temporal pattern is recovered when the pattern corresponding to the entire sequence of lattice nodes activated in L_2 (due to the sequential download process) has been recovered.

V. Computer Simulations

In order to test the performance of TOTEM, we performed computer simulations using two different data sets. The first data set consisted of 2-D spatiotemporal data that was generated using a set of eight 2-D geometric curves (listed in Table I) as primitives and selecting different combinations of these curves for different segments of the spatiotemporal trajectory. Originally, this data set consisted of $(x(t), y(t))$ samples for a duration that was randomly selected between 45 and 55 time units. The sampling rate was randomly selected between one and three samples per time unit. This data was then converted to spatial direction data as a function of time, $(n_x(t), n_y(t))$, that essentially consists of the unit directional



(a)



(b)

Fig. 10. TOTEM dynamics for two similar spatiotemporal trajectories. (a) The two spatiotemporal trajectorires and their corresponding L_3 neuronal activity. (b) The L_4 activities of the two trajectories and the error from the ideal trajectory.

vector components of the $(x(t), y(t))$ data. These components were computed as

$$n_x(t) = \frac{(x(t) - x(t-1))}{[(x(t) - x(t-1))^2 + (y(t) - y(t-1))^2]^{0.5}}$$
$$n_y(t) = \frac{(y(t) - y(t-1))}{[(x(t) - x(t-1))^2 + (y(t) - y(t-1))^2]^{0.5}}. \tag{17}$$

The advantage of using unit directional data samples is that it provides robustness to scale and translation in the spatiotemporal data.

The second data set consisted of three-dimensional (3-D) spatiotemporal data that was generated using a set of several 3-D geometric curves derived from a combination of six 3-D geometric curves listed in Table II as primitives. This data was also converted into a set of 3-D spatial direction data as a function of time. We will first analyze the performance during the PLR mode for the 2-D case. Then, we shall analyze the performance of TOTEM during the PR mode for the 2-D case. This will be followed by a similar analysis for the 3-D spatiotemporal trajectory data. While the 2-D

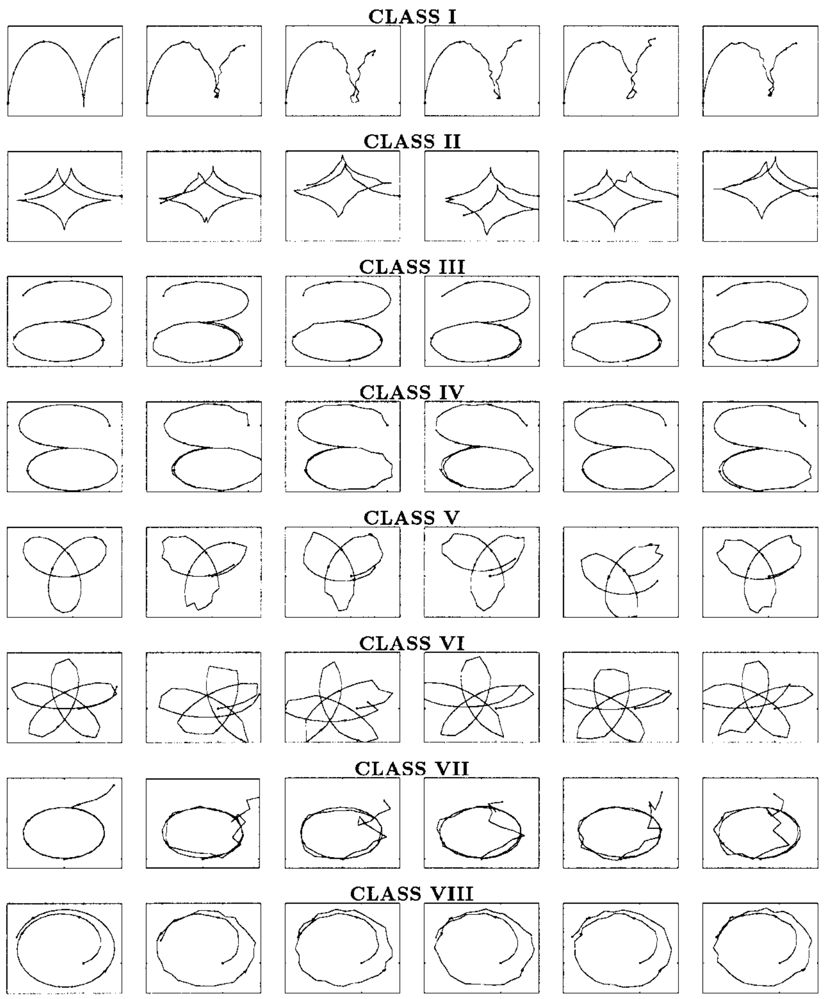


Fig. 11. The classification of eight 2-D spatiotemporal trajectories by TOTEM during the PLR mode.

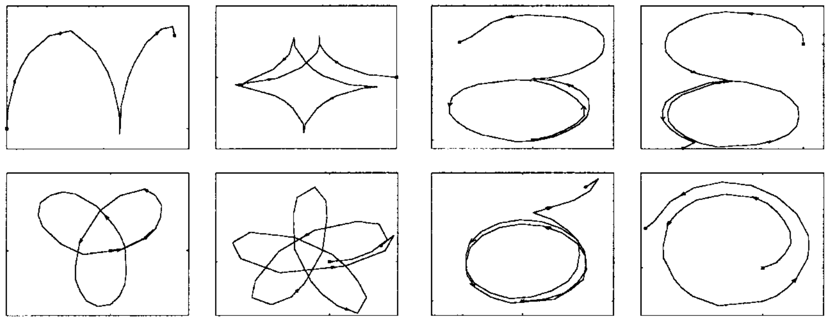


Fig. 12. Recall of the eight gemoetric curves by TOTEM for $\beta = 0.025$.

and 3-D data in these computer simulations were chosen for ease of visualization and understanding, the network is not restricted by the dimensionality of the spatiotemporal input data.

A. 2-D Topology Extraction Using SOM

Before the PLR mode was initiated, the preprocessor based on SOM was trained in order to learn the topology of the 2-D spatial direction data. The training was performed using 2-D spatial direction data with very little or no temporal order. The L_2 layer that consists of 25 (five along each dimension) lattice neurons (the issue of selecting this number is discussed in Section VI). It is possible to verify if the preprocessor learns the topology because the 2-D spatial directional data must lie on a unit circle. The various stages of learning is presented in the form of projection of the weight vectors $\mathbf{W}_{ij}$ (25 of them) onto the 2-D unit directional space as shown in Fig. 8. The location of the lattice neurons are depicted by an "o" mark. The connections between the lattice neurons are only shown to visualize how the 2-D lattice projects onto the input space. By the first 1250 input presentations, the network has abstracted a rough estimate of the input space topology. The learning is terminated after 3500 iterations. At this stage, all the lattice neurons except one lie on the unit circle. Thus, there are only four faulty connections between the lattice neurons (the four lines inside the unit circle of the rightmost image of second row in Fig. 8) among the 80 lattice connections in L_2 layer (i.e., an error of 5% in topology). In all our simulations, the parameters ϵ and σ were decreased as: $\epsilon = 0.99995 * \epsilon$ and $sigma = 0.99995 * \sigma$. This choice provided us with a rapid and accurate convergence as illustrated by Fig. 8.

B. 2-D Spatiotemporal Pattern Learning and Recognition

After the input topology is extracted, the processing of the 2-D spatiotemporal inputs is initiated. A total of 200 different spatiotemporal trajectories were presented to TOTEM (these were generated by various combinations of curves in Table I). Each pattern was also corrupted by gaussian noise with zero mean and a variance ranging from 5% to 20% of the spatiotemporal input at each frame. This created an additional 800 distorted spatiotemporal input trajectories. Some of these trajectories were also subjected to translations and scaling. At each frame, the spatiotemporal trajectory excites a lattice neuron (i.e., one among the 25 neurons) based on minimum Euclidean distance. This in turn is fed forward to L_3 were the temporal order of excitation is stored in the form of decreasing amounts of excitation of the L_3 lattice neurons.

In order to visualize the network dynamics, consider a spatiotemporal trajectory as shown in Fig. 9(a). This trajectory starts at the point marked "*" and terminates (i.e., $S = 1.0$) at the point marked "o." The total duration of the trajectory is 50 time units and the sampling rate is one point per time step. The L_3 level neuronal activity at the onset of sequential uploading (i.e., $G_1 = 1.0$) is plotted in Fig. 9(b). From this activity, it can be inferred that the first (lattice neuron with most activity) and last (lattice neuron with least activity) frames were captured at the lattice positions $(2, 4)$ and $(0, 4)$, respectively. Some of the lattice positions have multiple activities. These correspond to the occurance of a repeated event in the spatiotemporal trajectory. After the completion of the sequential uploading process (i.e., $G_1 = 0.0$), the resulting activity $\mathbf{y}$ of the neurons in the L_4 layer is shown in Fig. 9(c). At each frame, there are two bars to indicate the normalized lattice position (i, j) [refer to (9)]. It can be seen from Fig. 9(b) that the lattice positions range from zero to four along each dimension. Thus, if a lattice position $(2, 4)$ was uploaded from L_3, then its normalized activity would be represented by $(0.4, 0.8)$ in Fig. 9(c).

The activity at L_4 is used by the fuzzy ART algorithm to classify the spatiotemporal input at L_5. The vigilance parameter setting for classification of these L_4 activities was determined as follows. The vigilance parameter to classify the first 200 noise free spatiotemporal trajectories was determined. For the simulations reported here, this was found to be 0.9. Then, for processing the noisy versions, the vigilance was lowered until a winner that satisfied the vigilance criterion was selected in L_5 among the 200 created categories. For the example shown in Fig. 10(b), the vigilance criterion was satisfied at 0.83 for both the noisy versions. Thus, at this vigilance, the noisy versions are accepted into the same category as the ideal trajectory.

In order to illustrate the robustness of TOTEM to noise, let us consider the two spatiotemporal trajectories shown in Fig. 10(a). These are two noisy versions of the noise-free (or ideal) spatiotemporal trajectory shown in Fig. 9(a). The first trajectory (top left in Fig. 10(a)) is corrupted by a 8% Gaussian noise and is scaled and translated with respect to the ideal trajectory. The second trajectory (top right in Fig. 10(a)) is corrupted by a 6% Gaussian noise and is also scaled and translated with respect to the ideal trajectory. The sampling durations for the first and second trajectories are 47 and 48 time units and the sampling rate was fixed at one point per frame. The corresponding L_3 and L_4 activities are plotted in the second row of Fig. 10(a) and the first row of Fig. 10(b), respectively. The last row of Fig. 10(b) corresponds to the error in L_4 activity between the ideal and noisy spatiotemporal trajectories.

The classification results for the eight primitive geometric curves (listed in Table I) and their noisy versions are shown in Fig. 11. The first column corresponds to ideal trajectories (i.e., one among the first 200 trajectories) and the remaining columns correspond to noisy versions. Each input is directly coded into the L_5 layer by using the TOTEM network dynamics (described above) after a single presentation. If

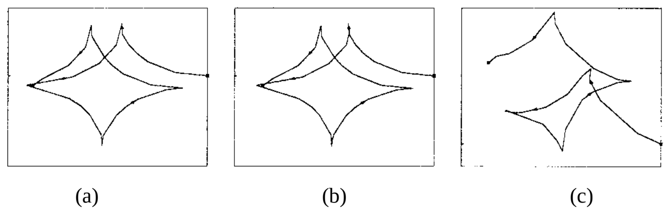


Fig. 13. The effect of recalled pattern due to β. (a) $\beta = 0.025$. (b) $\beta = 0.05$. (c) $\beta = 0.2$.

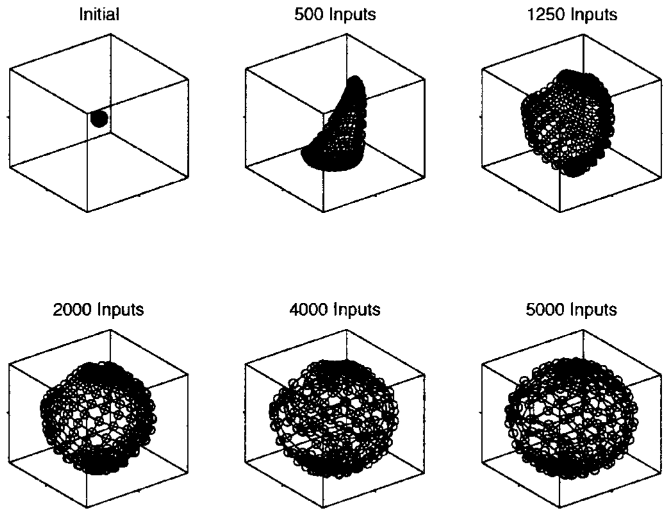
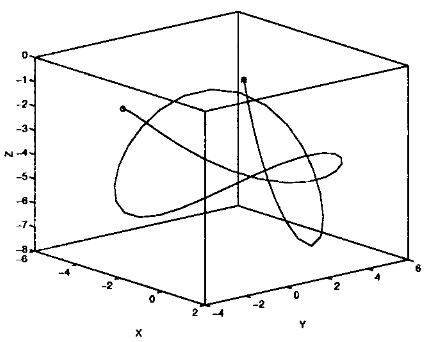
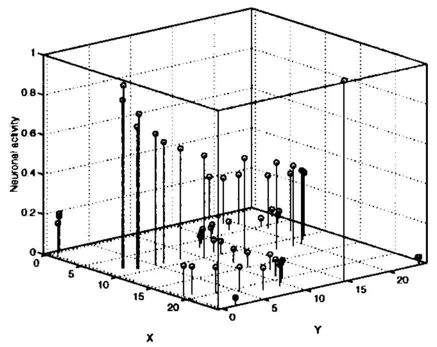


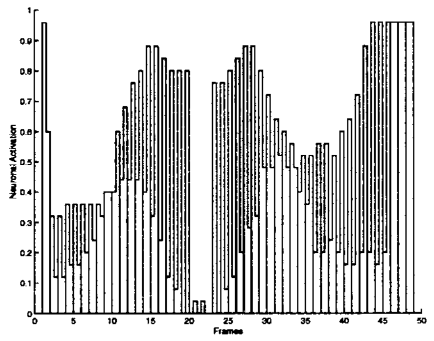
Fig. 14. The projection of W_{ij} onto input space during topology abstraction of the 3-D spatiotemporal data by SOM.



(a)



(b)



(c)

Fig. 15. Illustration of network dymanics. (a) Noise-free 3-D spatiotemporal trajectory. (b) L_3 level neuron activation at the end of the trajectory presentation (c) L_4 level neuron activation after sequential upload process.

any of the existing categories in L_5 does not match the spatiotemporal trajectory, then a new category is created on-line to accomodate the new spatiotemporal input (i.e., plasticity property). These results illustrate that TOTEM is capable of learning spatiotemporal inputs in an on-line and self-organized manner and is also robust/stable to noise in both spatial and temporal content.

C. 2-D Spatiotemporal Pattern Recall

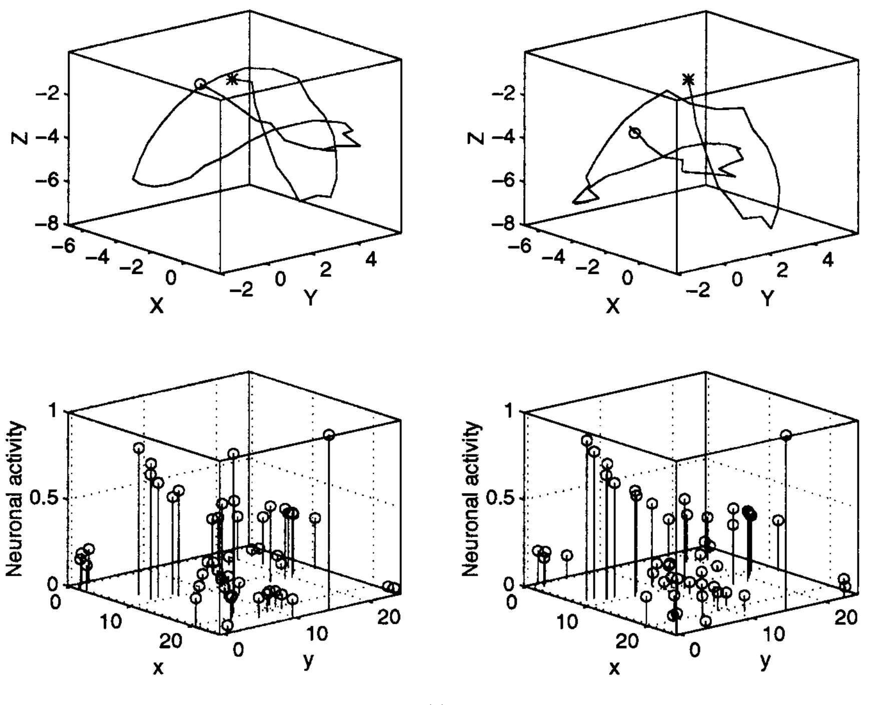
During the pattern recall mode, the category nodes in L_5 are probed and then the network tracks its way down to the LTM between the L_2 and L_1 layers to recall the spatiotemporal pattern (refer to Section IV-C). The recalled spatiotemporal patterns after the presentation of the 1000 spatiotemporal inputs (i.e., 200 noise-free and 800 noisy versions) for the eight primitive geometric curves are shown in Fig. 12. The recalled patterns match the noise-free version of their trajectories with high fidelity. The quality of this match depends upon the forgetting factor β [refer to (14)] that is used to adapt the LTM weights $\mathbf{Z}$ during PLR mode. A high value for β (i.e., $\beta > 0.2$) implies that the network will modify the stored template (i.e., $\mathbf{Z}$) for each category by more than 20% during the pattern recogntion mode. The modifed weights can affect the recall dynamics considerably as highlighted by Fig. 13. The pattern recalled in Figs. 13(a) and (b) correspond to β values of 0.025 and 0.05, respectively. It can be seen that these two recalled patterns have a very good match with its noise-free version (the plot in the second row, first column of Fig. 11). On the other hand, if $\beta = 0.2$, the recalled pattern is poor as illustrated in Fig. 13(c).

D. 3-D Topology Extraction Using SOM

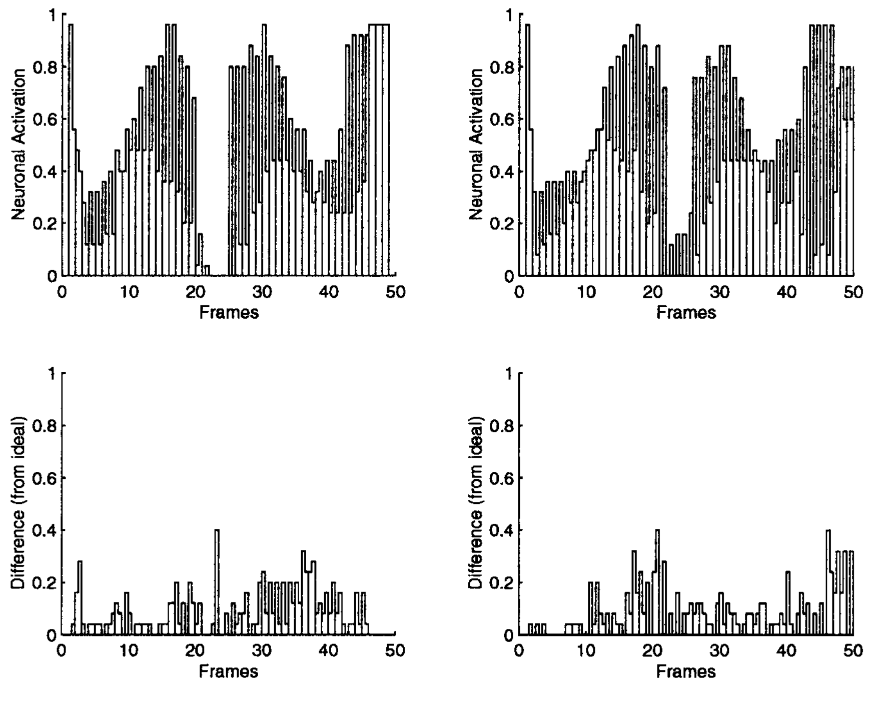
The SOM-based preprocessor was trained on 3-D spatiotemporal data in order to learn the topology of the 3-D spatial direction data. The training was performed similar to the 2-D case using 3-D spatial direction data with very little or no temporal order. The L_2 layer consists of 625 (25 along each dimension) lattice neurons. The topology of lattice neurons must assume the shape of a unit sphere because of the 3-D spatial directional data. The various stages of learning is presented in the form of projection of the weight vectors $\mathbf{W}_{ij}$ (625 of them) onto the 3-D unit directional space as shown in Fig. 14. The connections between the lattice neurons are only shown to visualize how the 2-D lattice projects onto the input space. By the first 2000 input presentations, the network has abstracted a rough estimate of the unit sphere. The learning is terminated after 5000 iterations. At this stage, all the lattice neurons except 12 lie on the unit sphere. Thus, there are only 47 faulty connections between the lattice neurons among the 1200 lattice connections in L_2 layer (i.e., an error of less than 4% in topology).

E. 3-D Spatiotemporal Pattern Learning and Recognition

A total of 150 different 3-D spatiotemporal trajectories were presented to TOTEM (these were generated by various combinations of the six geometric curves in Table II). Each pattern was also corrupted (similar to the 2-D case) by Gaussian noise ranging from 5% to 20% of the spatiotemporal input at each frame. This created an additional 750 distorted 3-D spatiotemporal input trajectories. Some of these trajectories



(a)



(b)

Fig. 16. TOTEM dynamics for two similar 3-D spatiotemoral trajectories. (a) The two trajectories and their corresponding L_3 neuronal activity. (b) The L_4 activities of the two trajectories and the error from the ideal trajectory.

were also subjected to translations and scaling. At each frame, the spatiotemporal trajectory excites a lattice neuron (i.e., one among the 625 neurons) based on minimum Euclidean distance. In order to illustrate the network performance, consider a spatiotemporal trajectory as shown in Fig. 15(a). The total duration of the trajectory is 50 time units and the sampling rate is one point per time step. The L_3 level neuronal activity at the onset of sequential uploading is plotted in Fig. 15(b). After the completion of the sequential uploading process, the activity $\mathbf{y}$ of the neurons in the L_4 layer is shown in Fig. 15(c). The activity at L_4 is used by the fuzzy ART algorithm to classify the spatiotemporal input at L_5.

In order to illustrate the robustness of TOTEM to noise, consider the two 3-D spatiotemporal trajectories shown in Fig. 16(a). These are two noisy versions of the noise-free (or ideal) spatiotemporal trajectory shown in Fig. 15(a). The

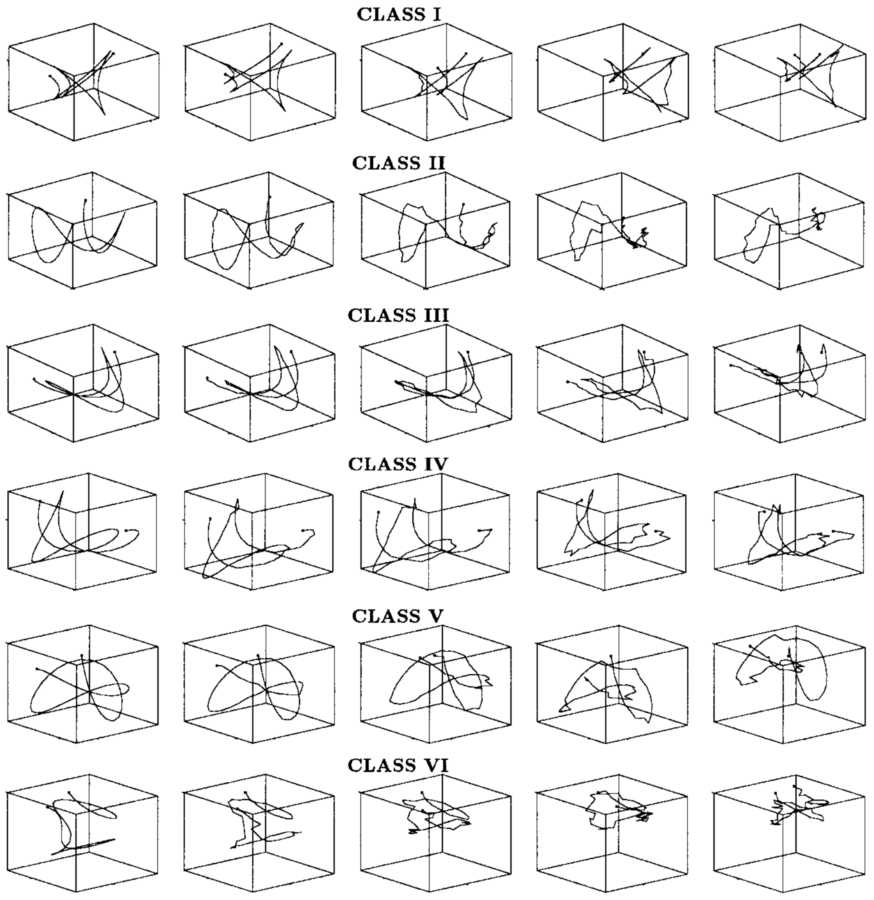


Fig. 17. The classification of the six 3-D spatiotemporal trajectories by TOTEM during the PLR mode.

first trajectory [top left in Fig. 16(a)] is corrupted by a 5% Gaussian noise and is scaled and translated with respect to the ideal trajectory. The second trajectory [top right in Fig. 16(a)] is corrupted by a 9% Gaussian noise and is also scaled and translated with respect to the ideal trajectory. The sampling durations for the first and second are both 46 time units and the sampling rate was fixed at one point per frame. The corresponding L_3 and L_4 activities are plotted in the second row of Fig. 16(a) and the first row of Fig. 16(b) respectively. The last row of Fig. 16(b) corresponds to the error in L_4 activity between the ideal and noisy spatiotemporal trajectories. The noisy versions are accepted into the same category as the ideal trajectory for a vigilance of 0.83 for the first and 0.81 for the second trajectory, respectively. The classification results for the six primitive 3-D geometric curves (listed in Table II) and their noisy versions are shown in Fig. 17.

F. 3-D Spatiotemporal Pattern Recall

The recalled 3-D spatiotemporal patterns after the presentation of the 900 spatiotemporal inputs (i.e., 150 noise-free and 750 noisy versions) for the six primitive geometric curves are shown in Fig. 18 for $\beta = 0.025$. These recalled patterns match well with the noise-free version of their trajectories (compare with the first column of Fig. 17). The effect of β for the 3-D spatiotemporal inputs is similar to the 2-D case.

VI. Discussion

In all our simulations, we have used spatial direction data instead of directly using the absolute position data. This was done to achieve high robustness to translations and scaling of the spatiotemporal data. However, if absolute position is important, we propose two approaches to handle the situation. The obvious approach is to present spatiotemporal data without extracting the spatial directions. If this is done, the network will not be as robust to translations and scaling as shown in this paper. The other approach could be to combine the absolute position information in L_4 along with the features already extracted by TOTEM (using the direction information) before learning the spatiotemporal input.

The inclusion of gate S in the TOTEM network might indicate that the network is actually operating in a supervised and not in a self-organized manner. This is because the opening of gate S indicates the termination of the input signal. However, unlike supervised learning networks, the network does not receive external feedback as to which category the input belongs to. The TOTEM network extracts the topological and temporal information and then classifies the input without external supervision. The network only uses the gate S for segmenting the spatiotemporal input. Thus, the network indeed operates in an unsupervised or self-organized manner. Another issue of importance is the selection of the number of neurons in the L_2 layer. If there are too many neurons, the benefit

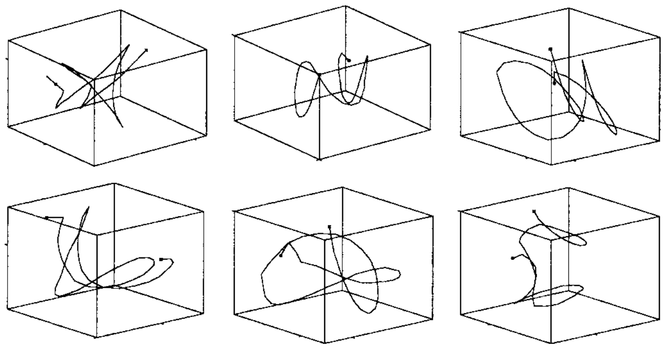


Fig. 18. Recall of the six gemoretric curves by TOTEM for $\beta = 0.025$.

is that the mapping is more acurate but at the cost of lack of adequate compression of the input information and vice versa. Data compression is important because it provides the network with robustness to noise in the spatial content of the spatiotemporal input. Thus, there is a tradeoff between these two factors. The number of neurons in L_2 in all our computer simulations were determined on a trial and error basis.

The vigilance parameter of the fuzzy ART network plays the role of a threshold on the similarity function and determines if a spatiotemporal input belongs to a given category or not. The value of the vigilance parameter in our network was selected based on the setting that creates a seperate class for each noise-free spatiotemporal pattern. This may not be possible in real situations were there may not be an explicit knowledge of whether an input is noise-free or not. In that case, a possible approach may be to first classify the inputs at multiple vigilance values. When a new input is presented, it is recognized based on the network that matched the input with the highest vigilance. This approach is however computationally expensive.

VII. Conclusion

We presented a novel five-layered self-organizing neural network for spatiotemporal pattern recognition and recall. This network can operate in either a pattern learning and recognition mode or a pattern recall mode. During the pattern recogntion and learning mode, the first layer is the input layer. It consists of feature detectors that receive the spatiotemporal input. The second layer extracts the topology of the features at each time step based on Kohonen's self-organizing maps. The sequence of excitation of the neurons in the second layer is then utilized by the third layer to extract the temporal order in the spatiotemporal pattern. The topology information extracted at the second layer is combined with the temporal order extracted in the third layer to derive a set of topologically and temporally correlated features in the fourth layer. These features are finally classified in the fifth layer using the fuzzy ART network. During pattern recall, the categories at the fifth layer are probed and the network is then able to reconstruct the spatiotemporal pattern that belongs to the probed category by using the LTM. The quality of this reconstruction depends on the value of the forgetting factor β. The network was evaluated via computer simulations using both 2-D and 3-D spatiotemporal inputs. The results show that the network is capable of learning, recognition and recall of spatiotemporal inputs in a self-organized and on-line manner. The results also demonstrate that the network is capable of handling repeated events as well as noise in the spatiotemporal data.

References

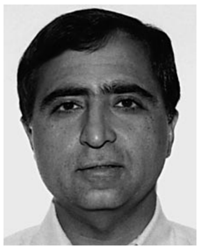
[1] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Proc. Int. Conf. Neural Networks*, 1987, pp. 609–618.
[2] U. Bodenhausen, "Learning internal representations of pattern sequences in a neural network with adaptive time-delays," in *Proc. Int. Conf. Neural Networks*, 1990, pp. 113–118.
[3] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vision, Graphics, Image Processing*, vol. 37, pp. 54–115, 1987.
[4] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 759–771, 1991.
[5] T. Darrell and A. Pentland, "Space-time gestures," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 1993, pp. 335–340.
[6] J. Daugmann, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *J. Opt. Soc. Amer.*, vol. 2, p. 1160, 1985.
[7] J. L. Elman, "Finding structure in time," *Cognitive Sci.*, vol. 14, pp. 179–211, 1990.
[8] C. L. Giles, C. B. Miller, D. Chen, G. Z. Sun, and Y. C. Lee, "Learning and extracting finite-state automata with second order recurrent neural networks," *Neural Comput.*, vol. 4, pp. 395–405, 1992.
[9] S. Grossberg, "How does a brain build a cognitive code?," *Psych. Rev.*, vol. 87, pp. 1–51, 1980.
[10] S. Grossberg, *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*. Boston, MA: Reidel, 1982.
[11] R. Hecht-Nielsen, "Nearest matched filter classification of spatiotemporal patterns," *Appl. Opt.*, vol. 26, pp. 1892–1899, 1987.
[12] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley, 1991.
[13] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Academy Sci., USA*, 1982, pp. 2554–2558.
[14] M. I. Jordan, "Attractor dynamics and parallelism in a conectionist sequential machine," in *Proc. 8th Annu. Conf. Cognitive Sci. Soc.*, 1986, pp. 531–546.
[15] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, pp. 59–69, 1982.
[16] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, pp. 281–294, 1989.
[17] B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Comput.*, vol. 1, pp. 263–269, 1989.
[18] J. B. Pollack, "Recursive distributed representation," *Artificial Intell.*, vol. 46, pp. 77–105, 1990.
[19] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[20] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps*. Reading, MA: Addison-Wesley, 1991.
[21] R. Rohwer and B. Forrest, "Training time-dependence in neural networks," in *Proc. Int. Conf. Neural Networks*, 1987, pp. 701–708.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” in *Parallel Distributed Processing*. Cambridge, MA: MIT Press, vol. 1, 1986.
[23] A. Waibel, “Modular construction of time-delay neural networks for speech recognition,” *Neural Comput.*, vol. 1, pp. 39–46, 1989.
[24] R. L. Watrous, “Speech recognition using connectionist networks,” Ph.D. dissertation, Univ. Pennsylvania, Philadelphia, PA, 1988.
[25] P. J. Werbos, “Generalization of backpropagation with application to a recurrent gas market model,” *Neural Networks*, vol. 1, pp. 338–356, 1988.
[26] R. J. Wiiliams and D. Zipser, “A learning algorithm for continually running fully recurrent neural networks,” *Neural Comput.*, vol. 1, pp. 270–280, 1989.
[27] P. M. Woodward, *Probability and Information Theory with Applications to Radar*. London, U.K.: Pergamon, 1953.
[28] L. Zadeh, “Fuzzy sets,” *Inform. Contr.*, vol. 8, pp. 338–353, 1965.

Narayan Srinivasa (S’90–M’95) received the B.Tech. degree from the Institute of Technology, Banaras Hindu University, India, in 1988, the M.S. degree from the University of Rhode Island, Providence, in 1991, and the Ph.D. degree from the University of Florida, Gainesville, in 1994, all in mechanical engineering.

From 1994–1997, he was a Beckman Fellow and a member of the Human Computer Intelligent Interaction Group at the University of Illinois at Urbana-Champaign. Currently, he is a Research Scientist in the Information Sciences Department at Hughes Research Laboratories, Malibu, CA. He has published more than 40 articles in peer-reviewed journals and conferences. His research interests include intelligent control, signal processing, computer vision, and neural networks.

Dr. Srinivasa is a member of ASME and INNS.



Narendra Ahuja (F’92) received the B.E. degree with honors in electronics engineering from the Birla Institute of Technology and Science, Pilani, India, in 1972; the M.E. degree with distinction in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1974; and the Ph.D. degree in computer science from the University of Maryland, College Park in 1979.

From 1974 to 1975 he was Scientific officer in the Department of Electronics, Government of India, New Delhi. From 1975 to 1979 he was at the Computer Vision Laboratory, University of Maryland, College Park. Since 1979 he has been with the University of Illinois at Urbana-Champaign, where he was named a professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute in 1988. He has coauthored the books *Pattern Models* (New York: Wiley, 1983) and *Motion and Structure from Image Sequences* (New York: Springer-Verlag, 1992). His interests include computer vision, robotics, image processing, image synthesis sensors, and parallel algorithms. His current research involves integrated use of multiple image sources of scene information to construct 3-D descriptions of scenes; the use of integrated image analysis for realistic image synthesis; parallel architectures and algorithms and special sensors for computer vision; and use of the results of image analysis for a variety of applications, including visual communication, image manipulation, video retrieval, robotics, and scene navigation. He was a Beckman Institute Associate in the University of Illinois Center for Advanced Study during 1990 and 1991.

Dr. Ahuja received the University Scholar Award in 1985, the Presidential Young Investigator Award in 1984, the National Scholarship from 1967 to 1972, and the President’s Merit Award in 1966. He is a fellow of the American Association for Artificial Intelligence, the International Association for Pattern Recognition, ACM, American Association for Advancement of Science, and International Society for Optical Engineering. He is a member of the Optical Society of America. He is on the editorial boards of IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE; *Computer Vision, Graphics, and Image Processing*; *Journal of Mathematical Imaging*; and *Journal of Information Science and Technology* and is a guest coeditor of the *Artificial Intelligence Journal’s* special issue on vision.