Learning Activity Patterns Using Fuzzy Self-Organizing Neural Network

Weiming Hu, Dan Xie, Tieniu Tan, and Steve Maybank

***Abstract*—Activity understanding in visual surveillance has attracted much attention in recent years. In this paper, we present a new method for learning patterns of object activities in image sequences for anomaly detection and activity prediction. The activity patterns are constructed using unsupervised learning of motion trajectories and object features. Based on the learned activity patterns, anomaly detection and activity prediction can be achieved. Unlike existing neural network based methods, our method uses a whole trajectory as an input to the network. This makes the network structure much simpler. Furthermore, the fuzzy set theory based method and the batch learning method are introduced into the network learning process, and make the learning process much more efficient. Two sets of data acquired, respectively, from a model scene and a campus scene are both used to test the proposed algorithms. Experimental results show that the fuzzy self-organizing neural network (fuzzy SOM) is much more efficient than the Kohonen self-organizing feature map (SOFM) and vector quantization in both speed and accuracy, and the anomaly detection and activity prediction algorithms have encouraging performances.**

***Index Terms*—Activity prediction, anomaly detection, fuzzy SOM, learning activity patterns.**

I. Introduction

Visual surveillance has attracted much attention in the computer vision community due to its potential applications. The main problem in visual surveillance systems include motion detection, object classification, tracking, activity understanding, and semantic description. Motion segmentation, moving object classification, and tracking have been widely studied for many years [30], [31], while activity understanding and semantic description have attracted much attention in recent years [1], [2], [37]. Activity understanding involves analyzing and recognizing motion patterns of objects, and producing high-level descriptions of object activities, and multiobject interactions. This paper focuses on activity understanding, and in particular, the learning of activity patterns, anomaly detection, and activity prediction. Other important issues, such as object tracking, are discussed elsewhere [4]–[6].

Most current visual surveillance and activity recognition systems depend on known scenes [19], where the objects move in predefined ways. These methods are not adaptable to changing environments because for each scene, one set of object activities has to be defined, and the definition of object activities must be updated as object activities are changed. Furthermore, it is hard to pre-define all object activities in some situations. Therefore, it is highly desirable to construct a general approach for activity recognition based on the activity patterns automatically generated by the self-organizing method or other effective methods [3]. Some previous efforts have been made to handle this problem. Fernyhough *et al.* [22] establish spatio-temporal regions by learning the results of tracking the objects in a video sequence, and constructing a qualitative activity model by qualitative reasoning and statistical analysis. Johnson *et al.* [23] describe a statistical model for object trajectories generated from image sequences. The movement of objects is described using the positions and velocities of the objects in the image plane. The statistical model of object trajectories is formed with two competitive learning networks which are connected with leaky neurons. Johnson *et al.* [24] generalize this method to the learning of interactions among humans, for example shaking hands. Stauffer *et al.* [25] present a method very similar to [23] where the learning activity patterns are used in real-time tracking. In this method joint co-occurrence statistics are accumulated over a codebook by treating the set of representations in each sequence as an equivalency multiset. Sumpter *et al.* [26] present a novel approach for learning long-term spatio-temporal patterns of objects in image sequences, using a neural network paradigm to predict future activities. Owens *et al.* [27] determine whether a point on a trajectory is normal using the distributions of flow vectors.

Given the desirability of automatically constructing activity patterns by self-organizing learning without prior knowledge of the pattern classes (namely without predefining motion patterns of objects), we propose a new self-organizing method for learning activity patterns for anomaly detection and activity prediction. The learning process is to classify the activity patterns represented by trajectories. By learning the trajectories and features of moving objects, the activity patterns are built up. Based on the learned activity patterns, our algorithm can detect anomalies and predict object activities. The main contributions of this paper are as follows.

1) A novel network mapping method is proposed to use a whole trajectory as an input to the network. This makes the network structure much simpler and the learning process much more efficient.
2) A fuzzy self-organizing neural network (fuzzy SOM) is based upon a learning algorithm that uses the batch manner [21] to introduce improved learning speeds and accuracy.
3) Mathematical methods are presented to detect anomalies and predict activities using the learned activity patterns.

This paper is organized as follows. Section II briefly reviews the related work. Section III introduces the method for acquiring training data. Section IV presents the fuzzy SOM method for learning activity patterns. Section V covers anomaly detection and activity prediction. Section VI describes experimental results. The last section summarizes the paper and discusses future work.

II. Related Work

In Section I, we have reviewed the references which concern construction of activity patterns by unsupervised learning in order to strengthen the motivation of this paper. These references are closely related to this paper. For completeness, we review in this section activity understanding methods by supervised learning. The major existing methods are outlined as follows.

1) *Dynamic time warping* (DTW): DTW is a template-based dynamic programming matching technique. It has been widely used for speech recognition in the early days and has been used recently in the matching of human movement patterns [7], [8]. For instance, Bobick *et al.* [8] use DTW to match an input signal to a deterministic sequence of states. Even if the time scale between a test pattern and a reference pattern is inconsistent, DTW can still successfully establish matching as long as the time ordering constraints hold.

2) *Finite state machine* (FSM): The most important feature of a FSM is its state-transition function. The states are used to decide which reference sequence matches with the test sequence. Wilson *et al.* [9] analyze

Manuscript received November 23, 2003; revised January 19, 2004. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 60105002, 60373046, and 40335010, the Natural Science Foundation of Beijing under Grant 4031004, the National 863 High-Tech Research and Development Program of China under Grant 2002AA117010-11 and 2002AA142100, the International Cooperation Project of Beijing, and the LIAMA Project. This paper was recommended by Associate Editor H. Ishibuchi.

W. Hu, D. Xie, and T. Tan are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China (e-mail: wmhu@nlpr.ia.ac.cn; dxie@nlpr.ia.ac.cn; tnt@nlpr.ia.ac.cn).

S. Maybank is with the School of Computer Science and Information Systems, Birkbeck College, London WC1E 7HX, U.K. (e-mail: sjmaybank@dcs.bbk.ac.uk).

Digital Object Identifier 10.1109/TSMCB.2004.826829

the explicit structure of natural gestures where the structure is implemented by an equivalent of a finite state machine but with no learning involved. Bremond *et al.* [10] use hand crafted deterministic automata to recognize airborne surveillance scenarios.

3) *Hidden Markov models* (HMMs): An HMM is a kind of stochastic state machine. HMMs generally outperform DTW in the processing of undivided successive data, and are therefore extensively applied to activity understanding. For instance, Starner *et al.* [11] propose an HMM-based approach for the recognition of sign language. Oliver *et al.* [12] propose and compare two different state-based learning architectures, namely, HMMs and CHMMs (coupled hidden Markov models) for modeling people activities and interactions. The CHMMs are shown to work much more efficiently and accurately than HMMs. Brand *et al.* [13] show that by minimizing the entropy of the joint distribution, a HMMs internal state machine can be made to organize observed activities into meaningful states.

4) *Time delay neural network* (TDNN): TDNN is also an interesting approach for analyzing time-varying data. As larger data sets become available, more emphasis is being placed on neural networks for representing temporal information. TDNN has been successfully applied to hand gesture recognition [14] and lip-reading [15]

5) *Syntactic techniques* [16]: The syntactic approach in machine vision has been studied mostly in the context of pattern recognition in static images. Recently the grammatical approach has been used for visual activity recognition. Brand [17] uses a simple nonprobabilistic grammar to recognize sequences of discrete activities. Ivanov *et al.* [16] describe a probabilistic syntactic approach to the detection and recognition of temporally extended activities and interactions between multiple objects.

6) *Nondeterministic finite automaton* (NFA): Wada *et al.* [18] employ NFA as a sequence analyzer. They present an approach for multi-object activity recognition based on activity driven selective attention.

III. Acquisition of Training Data

To learn object activity patterns, the training data representing object activities should be acquired. Our training data are composed of the features of trajectories and the features of moving objects.

Trajectories are acquired by tracking moving objects [4]–[6]. The centroids of an object at different times are connected to form a trajectory. Trajectories are sampled at fixed time intervals (once every Δt frames). Given an object o, the two–dimensional (2-D) image coordinates of its centroid at the ith sampling are (x_i, y_i). After sampling n times, we obtain a point sequence T_o which is composed of n pairs of 2-D image coordinates: $T_o = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)\}$. We use $(\delta x_i, \delta y_i)$ ($\delta x_i = x_{i+1} - x_i$, $\delta y_i = y_{i+1} - y_i$) to represent the apparent velocity of the object at time i. The velocity is very important for measuring the similarity between trajectories. At the ith sampling, the position of the object and its instantaneous velocity are represented by a flow vector $f_i = (x_i, y_i, \delta x_i, \delta y_i)$. Thus, the movement of object o is represented by set Q_o composed of n flow vectors: $Q_o = \{f_1, f_2, \ldots, f_i, \ldots, f_{n-1}, f_n\}$, where $\delta x_n = \delta x_{n-1}$, $\delta y_n = \delta y_{n-1}$. Similar trajectory coding schemes are used in [23] and [27].

The features of an object o, such as size and shape, are represented with F_o, so the input data become $X_o = \{F_o, Q_o\}$. In this paper, we only consider the object size. The apparent area of an object is considered to be its size. The area is estimated using the pixels corresponding to the moving object. The method can be of course easily extended to include more features such as shape, color, texture, etc. Thus, the training data sampled for object o are $X_o = \{size, f_1, f_2, \ldots, f_i, \ldots, f_{n-1}, f_n\}$. The size is used to distinguish different kinds of objects. For example, the sizes of a pedestrian and a vehicle are different, and even if they move along the same trajectory, their corresponding activities are treated as different.

In [25], the size of the object is added into the flow vector: $f_i = (x_i, y_i, \delta x_i, \delta y_i, size)$. When the size of an object can be treated as constant (if the camera is high enough above the scene or the object sizes are based on three–dimensional (3-D) models, the size of the object can be treated as constant), our approach is more appropriate. When the object size varies dramatically, we use the approach used in [25] to represent the object's features.

IV. Learning Activity Patterns

After sample trajectories are obtained, one is ready to learn activity patterns from the sample data. In this section, we describe the fuzzy SOM based learning algorithm. The algorithm includes a neural network, a mapping method and a learning algorithm.

A. Neural Network Model

1) Kohonen self-organizing feature map: The Kohonen self-organizing feature map (SOFM) [28], [29], [32] usually consists of a 2-D flat grid of simple nodes. Each node j (called an output neuron) has a weight vector W_j where the ith component of W_j is represented with W_{ij} which is the weight between the ith component of the input vector and the jth output neuron. The input feature vectors are presented sequentially to all of the neurons. For each input vector X, the best matching neuron c, compared with other neurons, holds the minimal Euclidean distance to X. the "neighborhood" is used to reflect the short range and side-feedback actions between neurons in the grid. The neurons in neighborhood NE_c of the best matching neuron c are all excited, while neurons outside neighborhood NE_c are inhibited.

There are two prominent problems in the SOFM.

a) Once a sample vector is fed into the network, it is brought to the neuron that is nearest to it in distance. This "winner takes all" rule goes against the network to grip all samples' features and thus affects learning speed and accuracy.
b) It is very difficult to select the proper network parameters (such as the learning rate function $\alpha(t)$, the neighborhood function $\beta(j, c)$ [28], [29], the neighborhood NE_c) in order to guarantee the success of training. $\beta(j, c)$ represents the distance relation between c and the neuron j within the neighborhood of c. It is a decreasing function of this distance.

In order to solve the first problem, batch manners which apply all available sample vectors as a batch in the regression is introduced. The typical one is the batch self-organizing feature map [21], [28] (Batch SOM). It is noted that an online SOM needs much less memory than a Batch SOM, and is also more realistic since the input vectors enter serially into the network. However, the learning process in our paper is offline. For offline learning, the batch manner is superior to the serial manner both in terms of computational speed and estimation accuracy. For the second problem, the time adaptive self-organizing map [35], [36] (TASOM) is introduced. The learning parameters of the TASOM are adaptive to the environment and the input vectors.

2) Fuzzy SOM: The fuzzy SOM [33], [34] introduces the concept of membership function in the theory of fuzzy sets to the learning process in the batch manner. The membership R_{lj} of each sample l to each neuron j is calculated, and then the weight vector of each neuron is adjusted according to all the memberships of all samples to the neuron. The learning algorithm will be detailed in Section IV-D. In the fuzzy SOM, some network parameters, such as $\beta(j, c)$, NE_c, related to the neighborhood in the SOFM are replaced with the membership function. Furthermore, the parameter $\alpha(t)$ is omitted. So the burden of choosing network parameters is eased. The fuzzy SOM considers all input data at each iteration step, and is thus more effective at decreasing oscillations and avoiding "dead units." The above fuzzy SOM is a combination of the SOFM and the fuzzy C clustering algorithm. Integration of the SOFM and other fuzzy set based algorithms can produce other variants of fuzzy SOM. For example, Osowski *et al.* [38] use the fuzzy SOM trained with the Gustafson–Kessel (GK) algorithm as a classifier.

B. New Neural Network Structure for Learning Activity Patterns

1) Review of existing neural network structures: Most neural networks currently used in trajectory analysis are competitive learning neural networks [23], and [26], [27]. The inputs to the neural networks are features of the point (corresponding to a flow vector) where a motion object is detected in the scene. Fig. 1(a) shows the neural network structure used in [23]. In [23], the statistical model of object trajectories is formed using two competitive learning networks that are connected with leaky neurons [20]. The first network is used to model the distributions of flow vectors. The number of its output neurons equals the number (G) of flow vectors. (G is chosen manually in [23], [26], [27]). The output of the first neural network is also the input to the second network. The leaky neurons act as memories of activations of the first neural network. The second network builds the distribution of trajectories. The number of its input vector components equals the number of output neurons in the first neural network (G). Its output neurons correspond to trajectories. Let the number of the output neurons in the second network be denoted by H. It is obvious that $G \gg H$. The learning speed of the second network is much slower than that of the first one. Fig. 1(b) shows the neural network structure used in [26]. Sumpter *et al.* [26] keep the first network and the input vector components in the second network the same as Fig. 1(a), and introduce feedback to the second network shown in Fig. 1(a) giving a more efficient prediction of object activities. This idea is very original, but the number of input vector components and the number (G) of output neurons in the second network remains equal to the number (G) of flow vectors, so the learning efficiency decreases inevitably as the size of the network increases. Furthermore, this neural network structure cannot be used to detect abnormal activities.

2) New neural network structure: We propose a new mapping method that uses the whole trajectory curve as an input to the network, as shown in Fig. 1(c). Each input vector represents a complete trajectory whereas the weight vectors of the neurons correspond to the classes of trajectories. If there are n sampling points on a trajectory, the input vector corresponding to this trajectory includes the components $(size, x_1, y_1, \delta x_1, \delta y_1, x_2, y_2, \delta x_2, \delta y_2, \ldots x_n, y_n, \delta x_n, \delta y_n)$. It can be seen that each input vector contains full information on a trajectory, including size, position, and velocity at each position, and in particular the linking relationships between successive positions. In our network model, the input vector components are composed of g sampling points corresponding to the trajectory that has maximal sampling points, and thus the number (N) of input components equals $4g + 1$. Flow vectors, the number of which is G, can form many trajectories containing the particular trajectory that has maximal sampling points. So, it is clear that the number (N) of input vector components in our network is much smaller than that (G) in the second network shown in Fig. 1(a) ($N \ll G$). The number of output neurons in our network structure is the same as that in the second network shown in Fig. 1(a). It is thus clear that our network is much simpler than those shown in Fig. 1(a) and (b). However, our network can realize the same functions of learning activity patterns, anomaly detection and activity prediction as the others.

C. Normalization

In order to use a whole trajectory as an input, input samples should be normalized to the same length. Suppose that one input sample has n sampling points, where (x_n, y_n) and (x_{n-1}, y_{n-1}) represent, respectively, the coordinates of the last and the second to last points in the trajectory. As mentioned above, the components of the input vector are composed of g sampling points corresponding to the trajectory that has maximal sampling points, so $g - n$ flow vectors are required to be padded to the input sample to obtain a vector with g sampling points. The $g - n$ flow vectors are represented by the formula, as shown in (1) at the bottom of the page.The physical meaning of the normalization is that the object is allowed to move c points with the same velocity as that when the object was detected at the last sampled point, and then it stops. "c" is a small figure (In our experiments, c is chosen to be 3). It is assumed that the object will go somewhat beyond the scene. After the learning introduced in Section IV-D is completed, the trajectory length processing introduced in Section IV-E will truncate the padded points and thus model different trajectory lengths. Because of the trajectory length processing, the affection of this padding can be ignored.

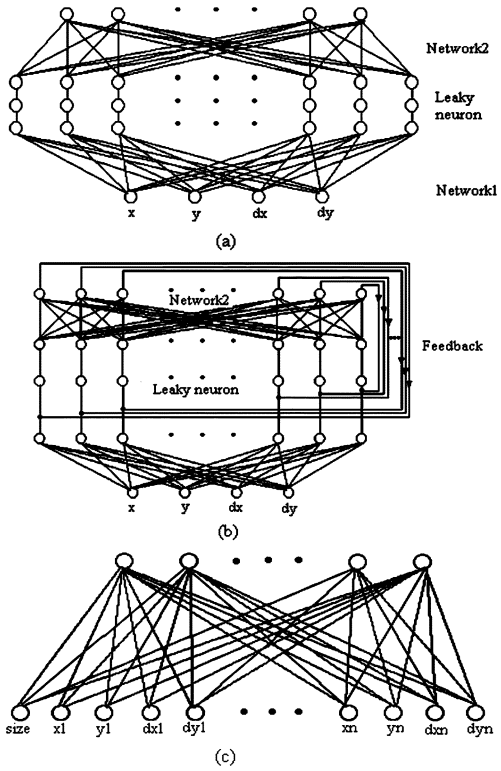


Fig. 1. Comparison between neural network structures. (a) Neural network structure used in [23]. (b) Neural network structure used in [26]. (c) Neural network structure used in this paper.

D. Network Learning Algorithm

In our neural network structure, each output neuron directly corresponds to a class of trajectories. The number of output neurons used to describe the activity patterns is essentially arbitrary. The more neurons used the greater the accuracy of the model. The number of output neurons needed for good accuracy depends on the complexity of a scene. The more complex the scene is, the more output neurons are required. The number of output neurons is manually selected. The weights (W) connect the input vector components and the output neurons. The weight vectors are of the same dimensions as the sample vectors. The weight components are initialized randomly and adjusted gradually using a self-organizing learning algorithm, and ultimately a

$$\begin{cases} (x_n + (x_n - x_{n-1}) * i, y_n + (y_n - y_{n-1}) * i, x_n - x_{n-1}, y_n - y_{n-1}), & i = 1, \ldots, c \\ (x_n + (x_n - x_{n-1}) * c, y_n + (y_n - y_{n-1}) * c, 0, 0), & i = c+1, \ldots, g-n, \end{cases} \tag{1}$$

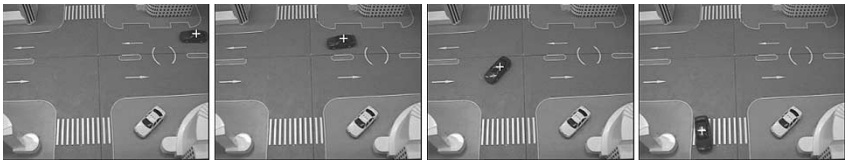


Fig. 2. Tracking of toy vehicles in a model scene.

Fig. 3. Tracking of pedestrians in a campus scene.

mapping, from input to output, that keeps the distribution features of trajectories is formed.

Let M denote the number of input samples, N the number of input vector components, and K the number of output neurons. The learning algorithm consists of the following steps.

Step 1) Randomize the initial values of the components of the weight vectors.

Step 2) Input all samples $X_l = [X_{l,1}, X_{l,2}, \ldots, X_{l,N}]$, $l = 1, 2, \ldots, M$.

Step 3) Calculate the Euclidean distances from each sample X_l to all output neurons

$$d_{lj}(t) = \sqrt{\sum_{i=1}^{N} (X_{li} - W_{ij}(t))^2} \quad l = 1, 2, \ldots, M, \quad j = 1, 2, \ldots, K. \tag{2}$$

Step 4) Compute the memberships of each sample to all neurons

$$R_{lj}(t) = \frac{\frac{1}{d_{lj}^2(t)}}{\sum_{m=1}^{K} \left(\frac{1}{d_{lm}^2(t)} \right)} \quad l = 1, 2, \ldots, M, \quad j = 1, 2, \ldots, K. \tag{3}$$

Step 5) Adjust the weights of each neuron according to the computed memberships

$$W_{ij}(t+1) = W_{ij}(t) + \frac{\sum_{l=1}^{M} R_{lj}(t) \cdot (X_{li} - W_{ij}(t))}{\sum_{l=1}^{M} R_{lj}(t)}. \tag{4}$$

Step 6) Determine the stability condition of the network

$$\max_{\substack{1 \le i \le N \\ 1 \le j \le K}} \{|W_{ij}(t+1) - W_{ij}(t)|\} < \varepsilon. \tag{5}$$

If the stability condition is satisfied or the predefined number of iterations is achieved, then the learning process terminates; otherwise go to Step 2 for another loop of learning.

From the above offline learning procedure, we can see that the fuzzy SOM eases the difficulty of selecting network parameters. In the above learning procedure, the weights are adjusted only once in each learning loop and the features of all input samples are taken into consideration once the weights are adjusted, so the learning speed and estimation accuracy are both greatly improved. In fact, different kinds of fuzzy membership functions can be used in the above learning algorithm. In above fuzzy membership functions, all features of samples have been considered. This favors validity of the learning algorithm.

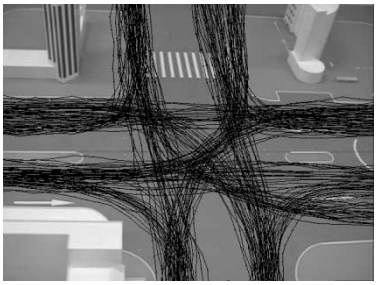


Fig. 4. Samples from model scene.

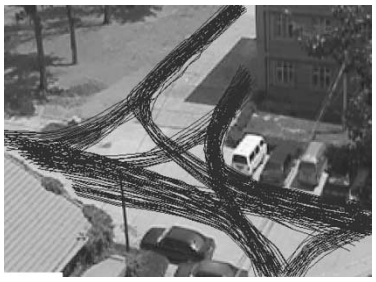


Fig. 5. Samples from campus scene.

E. Trajectory Length Processing

After the learning is completed, trajectory length processing is used to adjust the weight vector of each output node to the original length. The adjustment method is: we input the trajectory samples into the learned network and then truncate the points, in the tail of each weight vector, which correspond to padded points. For each output neuron j, we find all trajectory samples which best match it. These trajectory samples are represented with set $S = \{S_1, S_2, \ldots, S_A\}$ (suppose there are A such samples). The trajectory samples in set S may be not padded with the same number of points in the length normalization process. It is easy to find what number of points is mostly padded. If, in sample set S, most samples were padded with m points, then we truncate m points at the latter part of the weight vector of output node j. Thus, trajectories of different lengths are modeled.

V. Anomaly Detection and Activity Prediction

The representative activity patterns are obtained after learning normal trajectories. Based on the learned activity patterns, we can judge whether or not an observed activity is abnormal, and predict the

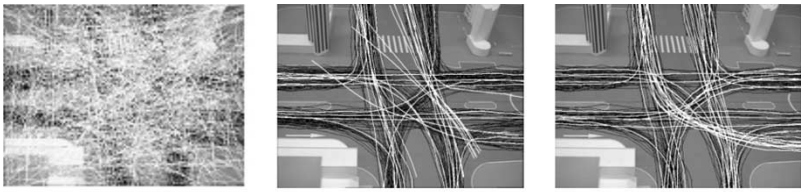


Fig. 6. Learning results of indoor scene with SOFM. (a) Random initial weights. (b) After 100 iterations (5.17 s). (c) After 598 iterations (30.44 s, over). P3(Mean) = 91.45, and P4(Variance) = 3677.26.

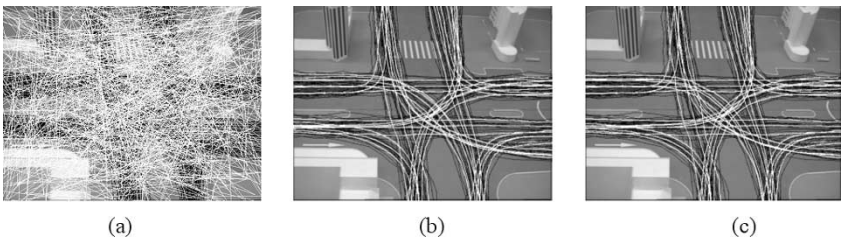


Fig. 7. Learning results of indoor scene with fuzzy SOM. (a) Random initial weights. (b) After 50 iterations (2.76 s). (c) After 132 iterations (7.51 s, over). $P3 = 71.58$ and $P4 = 1342.35$.

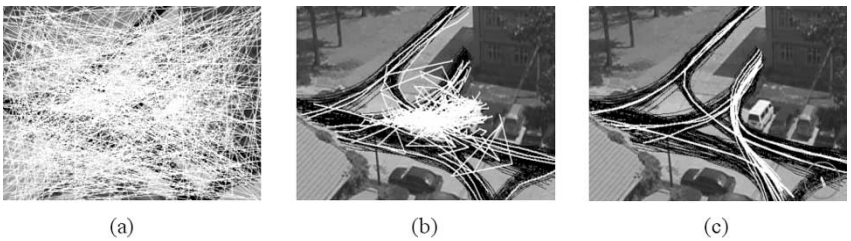


Fig. 8. Learning results of outdoor scene with SOFM. (a) Random initial weights. (b) After 50 iterations (5.48 s). (c) After 675 iterations (52.08 s, over). $P3 = 90.42$ and $P4 = 4175.57$.

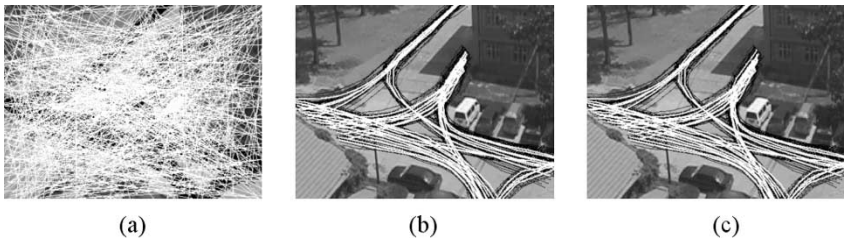


Fig. 9. Learning results of outdoor scene with fuzzy SOM. (a) Random initial weights. (b) After 50 iterations (5.06 s). (c) After 143 iterations (14.76 s, over). $P3 = 63.41, P4 = 1409.46$.

future trajectory along which the object will move according to the current partial trajectory.

A. Anomaly Detection

1) Detection of Abnormal Trajectories: Given a trajectory $T_o = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$, by using the input vector $X_o = (size, x_1, y_1, \delta x_1, \delta y_1, x_2, y_2, \delta x_2, \delta y_2, \ldots x_n, y_n, \delta x_n, \delta y_n)$, we look for the neuron that best matches the input vector. Then, the Euclidean distance (represented with D_o) between the input vector and the best matching neuron (j) is calculated. If D_o/n is greater than a threshold q_j, the trajectory is considered as abnormal and the activity represented by the trajectory is treated as abnormal. We use the Euclidean distances between samples which best match the neuron j and weight vector of neuron j to calculate the threshold value q_j. This means if the difference between the input vector and its best matching neuron is greater than that between the sample vectors and the neuron, the input vector is treated as abnormal. The threshold value q_j depends on the maximal Euclidean distance between a sample vector that best matches the neuron j and the weight

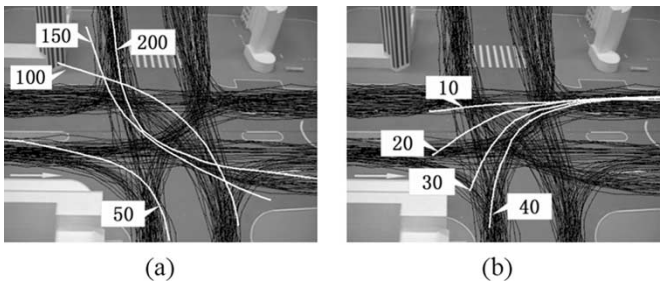
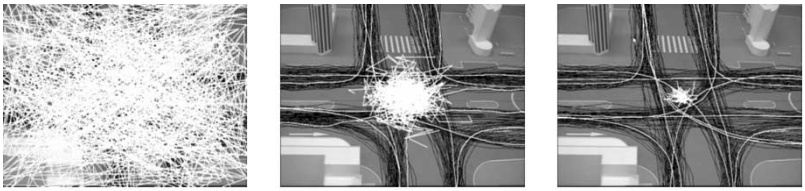


Fig. 10. Example of oscillation. (a) SOFM (oscillation occurs). (b) Fuzzy SOM (no oscillation).



(a) (b) (c)

Fig. 11. Activity pattern learning with vector quantization (traffic model scene). (a) Random initial weights. (b) After 100 iterations (5.19 s). (c) After 797 iterations (40.88 s). $P3 = 122.47$ and $P4 = 4025.65$.

TABLE I
COMPARISON OF THE PERFORMANCES

The stability condition ε	Vector quantization				SOFM				Fuzzy SOM			
	P1	P2	P3	P4	P1	P2	P3	P4	P1	P2	P3	P4
0.0001	924	47.39s	112.39	3546.30	739	37.94s	81.26	2865.77	216	12.78s	68.24	976.92
	817	41.91s	117.25	4008.44	617	31.63s	89.10	3280.92	227	13.34s	69.83	1262.47
	883	45.31s	112.70	3718.67	662	34.03s	88.61	3297.59	206	12.17s	69.63	1036.62
0.001	797	39.88s	116.62	3917.31	593	30.44s	89.92	3481.59	138	8.01s	72.38	1431.42
	822	42.17s	116.81	3583.40	614	31.50s	86.38	3322.51	141	8.14s	72.44	1559.92
	910	46.69s	121.95	4103.61	595	30.52s	91.51	3408.61	132	7.62s	71.04	1480.57
0.01	779	38.94s	133.47	4820.32	507	26.02s	95.98	4190.26	116	6.68s	74.96	2186.08
	667	34.20s	123.98	4578.75	565	28.99s	93.76	3707.05	120	6.90s	75.06	1813.24
	784	39.19s	124.08	3969.05	578	29.70s	93.04	3718.79	114	6.54s	72.03	1477.51

vector of neuron j. We find all samples which best match neuron j. For each of these samples X_l on which, we suppose, there are m sample points, we calculate the Euclidean distance (D_l) between X_l and the weight vector of neuron j. $q_l = D_l/m$ (D_l is divided by m). Then we take one half of the maximum of all q_l as the threshold q_j

$$q_j = \frac{1}{2}\max_l q_l. \tag{6}$$

The following points should be noted.

a) As trajectories of different lengths are modeled, longer trajectories gain more Euclidean distance. In order to balance different lengths of trajectories and treat each point equally in a trajectory, the Euclidean distance should be divided by the number of points on a trajectory, when we calculate the threshold q_j and judge if the input vector is abnormal.

b) Here we take 1/2 maximum of all q_j as q_j. In this way, some normal trajectories may be treated as abnormal. If we take the maximum of all q_i as the threshold q_j, no sample trajectory, which best matches neuron j, has D_o/n greater than q_j, but some abnormal trajectories may not be detected. Avoidance of missing detection of some anomalies is more important than avoidance of mistaking a normal activity as an anomaly. In real applications, detected anomalies will be checked by humans. So we select the threshold q in this way.

2) Detection of wrong sections of an abnormal trajectory: For a detected abnormal trajectory, we find its closest matching neuron and calculate the distance from the ith point on the trajectory to the corresponding point in the neuron. If the distance is greater than a threshold q_i^*, the point and the section where it is located are treated as abnormal. To do this, the threshold q_i^* must be computed. We find all the trajectory samples which best match with the neuron, compute the distance d_j from the ith point on each of the sample trajectories (j) to the corresponding point in the output neuron, and take half of the maximum of d_j as q_i^*

$$q_i^* = \frac{1}{2}\max_j d_j, \quad i = 1, 2, \ldots, n \tag{7}$$

where n is the number of standard sampling points on the trajectory.

B. Activity Prediction

Given part of a motion trajectory T_o, we can sample it to get a subsample. Suppose there are k sample points in T_o. The subsample vector is represented as follows: $(size, x_1, y_1, \delta x_1, \delta y_1, x_2, y_2, \delta x_2, \delta y_2, \ldots x_k, y_k, \delta x_k, \delta y_k)$. Each point in T_o doesn't contribute equally to the matching between T_o and the learned activity patterns. For activity prediction, the older the information the less useful it is. So, the contribution of current point (x_k, y_k) is the most, that of point (x_{k-1}, y_{k-1}) is the second, ..., and that of point (x_1, y_1) is the least. We introduce a weight $T(i)$ for each point (x_i, y_i)

$$T(i) = e^{-[(i-k)/k]^2} \quad i = 1, 2, \ldots, k. \tag{8}$$

$T(i)$ decreases when i decreases from k to 1.

The matching score between the sub-sample and each output neuron j is the weighted Euclidean distance between the subsample vector and the vector made up of the first $4k+1$ components of the output neuron

$$\begin{aligned} r_j = &(size - w_{1j})^2 + \sum_{i=1}^{k}((x_i - w_{4i-2,j}(t))^2 \\ &+ (y_i - w_{4i-1,j})^2 + (\delta x_i - w_{4i,j})^2 \\ &+ (\delta y_i - w_{4i+1,j})^2)T(i) \end{aligned} \tag{9}$$

where $j = 1, 2, \ldots, K$ (K is the number of the output neurons). Once the matching score r_j of the subsample to output neuron j is obtained, we can calculate the corresponding probability between T_o and each output neuron, which is the probability of the future motion trajectory of the object. The trajectory represented by the neuron with the highest probability is chosen as the most probable one along which the object will move in the future. This probability is computed according to

$$P_j = \frac{\frac{1}{r_j}}{\sum_{i=1}^{K}\frac{1}{r_i}}, \quad j = 1, 2, \ldots, K. \tag{10}$$

This conversion from a weighted Euclidean distance to a probability is approximate rather than strict, favoring the neurons with small distance

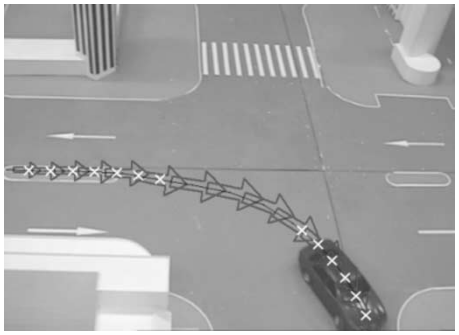


Fig. 12. Anomaly detection in model scene.

Fig. 13. Anomaly detection in campus scene.

to partial trajectory X. If the probability in which the sub-sample corresponds to an output neuron is lower than a threshold (e.g., 20%), the object will not be thought to move along the trajectory represented by the output node.

VI. Experimental Results

All the above algorithms are implemented using Visual C++6.0 on the Windows2000 platform. In the following, training samples are first introduced. The different methods for learning activity patterns are then compared using experimental results. Finally, the results of anomaly detection and activity prediction are demonstrated.

A. Training Samples

In this paper the tracking of moving objects is done automatically, using the method derived from our previous papers [4]–[6]. Only trajectories, in which perfectly tracking results, are manually selected for samples. The following two sets of training data are used.

1) The first set of training data is acquired by tracking the moving toy vehicles controlled by radios in a traffic model scene, as shown in Fig. 2. The tracked vehicle is labeled with a white cross, whose center is at the centroid of the object.
2) The second set of training data is acquired by tracking moving objects, such as pedestrians, bikes and vehicles in an outdoor campus. An example of pedestrian tracking is shown in Fig. 3.

With continuous tracking, we acquired two sets of trajectories, as shown in Figs. 4 and 5 which, respectively, correspond to 201 and 268 trajectories. The sample trajectories are smoothed

B. Learning Activity Patterns

In the following, the fuzzy SOM method is first compared with the SOFM method and then with the vector quantization method used in [23], [26]. We use the following four numbers to assess the performance of an algorithm for learning activity patterns.

1) The number of iterations (P1) and computational cost (P2) needed to reach the predefined stability condition. These two numbers are used to evaluate the speed of the algorithm. In order to make a fair comparison between the batch manner and the serial manner the number of iterations for the latter two algorithms, which use the serial manner, is divided by the number of training samples. All the following running times are calculated on a Pentium3-933 computer with 256 MB RAM.
2) The mean (P3) and variance (P4) of all Euclidean distances between each sample and its best matching neuron. These two numbers are used to evaluate the learning results of the algorithm.

1) Comparison With SOFM: By using the same network mapping method as the presented fuzzy SOM method, it is easy to construct the SOFM method for learning activity patterns. The values for parameters of the SOFM are chosen as follows:

$$\alpha(t) = Ae^{-t/\tau} \tag{11}$$

where τ is a positive constant and $0 < A \leq 1$. Neighborhood NE_c shrinks linearly with the increase in t until it contains only one neuron.

$$\beta(j,c) = \frac{1}{\sqrt{d+1}} \tag{12}$$

where d is the distance between j and c in the grid. The following diagrams (Figs. 6–9) illustrate the convergence properties of the two algorithms at different iteration steps. In the diagrams, the black lines represent the trajectory samples used for training, and the white ones denote the output neurons that represent the learned activity patterns.

Figs. 6 and 7 show the comparison between the convergence results of the SOFM and the fuzzy SOM in the indoor model scene. The number of the output nodes is 40, and the network stability condition is $\epsilon = 0.001$.

Figs. 8 and 9 show the comparison between the convergence results of the SOFM and the fuzzy SOM in the outdoor campus scene. The number of output neurons is 50 and the network stability condition is $\epsilon = 0.001$.

From the above experimental results, we can see that when the number of the input samples and the number of the neurons are selected to be the same, both the required number of iterations and the running time of the fuzzy SOM are much less than those of the SOFM to meet the same network stability condition. This means that the learning speed of the fuzzy SOM is faster than that of the SOFM. The learning processes show that oscillations for some neurons often occur in the SOFM. Fig. 10(a) shows an example of local oscillation. It illustrates that one and the same neuron undergoes severe oscillation among several classes of input samples in four different iteration phrases (after 50, 100, 150, and 200 iterations). Fig. 10(b) shows that in four different iteration phrases (after 10, 20, 30, and 40 iterations) the same neuron almost corresponds to the same type of input samples. It is also shown that the learned results with the SOFM have some dead units [as circled in Fig. 8(c)] and the distributions of the activity patterns are affected, whereas the results with the fuzzy SOM have no dead unit. The P3 and P4 of the fuzzy SOM are both less than those of the SOFM. This means the learned activity patterns with the fuzzy SOM are more consistent with the samples. The consistency also can be evaluated by one's visual judgement.

2) Comparison with vector quantization: In [23] and [26], vector quantization is used to train the network in order to learn the distributions of trajectories. In the following we compare our fuzzy SOM method with the vector quantization method in order to illustrate that our method outperforms the existing ones. Vector quantization does not take the neighborhood into consideration. When a neuron c best matches the input vector, only neuron c is excited, and all others are inhibited. The weight sensitivity is used to ensure that each neuron can be excited at some stage. In experiments we find that when vector quantization is used to learn activity patterns, most neurons are not excited at the early stage of training and shift toward the center of samples just according to the weight sensitivity determined in the learning process. This slows down the speed of the network convergence and

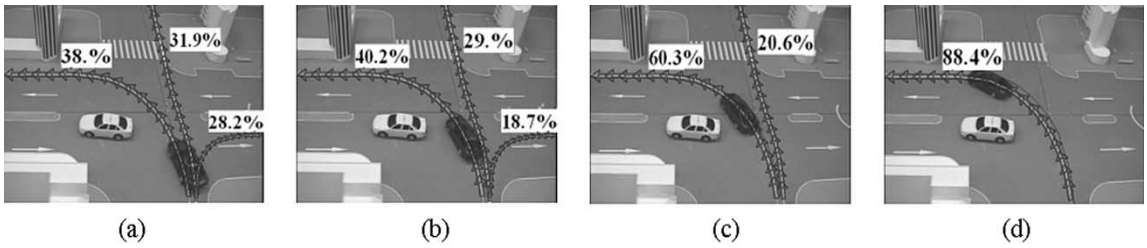


Fig. 14. Prediction in indoor model scene.

Fig. 15. Prediction in outdoor campus scene.

greatly affects the learning accuracy. In addition, local oscillation may occur easily. Fig. 11 shows the learning results of vector quantization when the number of output nodes and the network stability condition are the same as Figs. 6 and 7. Both the learning speed and the learning effectiveness of vector quantization are worse than those of the SOFM which are in turn worse than those of the fuzzy SOM.

Table I demonstrates the performance measures P1, P2, P3, and P4 of the three training algorithms for the model scene with different initial weights and different stability conditions. The number of output neurons is fixed at 40. We see, from Table I, that the fuzzy SOM needs much fewer iterations and less running time to reach the same stability condition than the SOFM, and the SOFM needs a little fewer iterations and a little less running time than vector quantization. The P3 and P4 of the fuzzy SOM are less than those of SOFM, which are less than those of vector quantization. Extensive experiments show that the fuzzy SOM is clearly the best learning scheme. It has much faster learning speed and much more effective learning results than the SOFM and vector quantization.

Now we compare our method for learning activity patterns with that used in [23]. From Fig. 1, we can see that the learning time (T) of [23] includes learning time of neural network $1(T_1)$, running time of leaky neurons (t) and learning time of neural network $2(T_2)$, i.e., $T = T_1 + t + T_2$. As mentioned in Section IV-B, even the scale of the network 2 used in [23] is much bigger than ours. For the same scale of network, our learning method is faster and more effective than that used in [23]. So we can conclude that our method for learning activity patterns is much more efficient than that used in [23].

C. Anomaly Detection

With the learned activity patterns, we can use the method introduced in Section V-A1 to detect abnormal trajectories and further use the method introduced in Section V-A.II to find the wrong sections of an abnormal trajectory. The correctness of the following anomaly detection results is evaluated by operator's visual judgement. Fig. 12 is an example of anomaly detection in the indoor model scene. The trajectory of the car is shown as a series of arrows, with the size of the arrowhead representing the speed of the object. The abnormal points are marked with white crosses at the center of the arrowheads. The car entered the scene from the left and then turned right. At the beginning the points are marked as abnormal as they are too close to the left side of the lane. Later the car turned right and moved within the proper region. However, when the car began to move down, it began to enter the wrong lane and this is correctly marked as abnormal. Fig. 13 illustrates another example of anomaly detection in the outdoor campus scene.

We used 40 known abnormal trajectories and 40 known normal trajectories to test the performance of the anomaly detection method. The results are as follows: 37 abnormal trajectories are correctly recognized; the recognition rate is 92.5%(37/40); the misdetection rate is 7.5%(3/40). 34 normal trajectories are correctly recognized; the recognition rate is 85%(34/40); the false alarm rate is 15%. The integrated recognition rate is 88.75%(71/80); the integrated midsection rate is 11.25%(9/80).

D. Activity Prediction

Fig. 14 shows an example of prediction in the indoor model scene. The percentage beside a trajectory represents the probability (calculated using Formula (10) in Section V-B with which the car will move along the trajectory. The car entered the scene from the bottom and then turned left. Fig. 14(a) shows the three most probable trajectories which the car might follow; in (b), the probability with which the car will move along these three trajectories is changed; in (c), the right-turn trajectory is eliminated because the probability of the car making a right turn is very small. In (d) the forward trajectory is also removed for the same reason. Another similar example of activity prediction in the campus scene is demonstrated in Fig. 15. Both examples in Figs. 14 and 15 show that the prediction is consistent with one's visual judgement, demonstrating the good accuracy of the algorithm in predicting object activities.

VII. Conclusion

In recent years, activity understanding has attracted much attention. Most current visual activity understanding methods depend on known scenes, where objects move in predefined ways. It is highly desirable to automatically construct activity patterns by self-organizing learning rather than predefine them manually. In this paper, we have presented a new self-organizing method for learning activity patterns for anomaly detection and activity prediction. Unlike existing methods that use individual flow vectors as inputs, our method takes a whole trajectory as an input. This makes the neural network structure much simpler. Furthermore, we have introduced the fuzzy SOM to improve the learning speed and accuracy. Based on the learned activity patterns, anomaly detection and activity prediction are realized. Experimental results using two different sets of data have demonstrated the effectiveness of the proposed algorithms.

Our future work will focus on the following three aspects.

1) We will use the detection probability theory to identify abnormal movements to increase the flexibility of the abnormal detection method.
2) We will apply the methods, for automatically extracting the rules explaining the phenomena hidden into the input data, for activity analysis.
3) We will try to introduce the interactions between objects to the activity patterns.

ACKNOWLEDGMENT

The authors wish to thank the reviewers for their constructive comments.

REFERENCES

[1] T. Collins, A. J. Lipton, and T. Kanade, "Introduction to the special section on video surveillance," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 745–746, Aug. 2000.

[2] R. J. Howarth and H. Buxton, "Conceptual descriptions from monitoring and watching image sequences," *Image Vis. Comput.*, vol. 18, no. 9, pp. 105–135, 2000.

[3] W. M. Hu, D. Xie, and T. N. Tan, "A hierarchical self-organizing approach for learning the patterns of motion trajectories," *IEEE Trans. Neural Networks*, vol. 15, pp. 135–144, Jan. 2004.

[4] J. G. Lou, H. Yang, W. M. Hu, and T. N. Tan, "Change detection for visual surveillance," in *Proc. Asian Conf. Computer Vision*, 2002, pp. 13–18.

[5] H. Yang, J. G. Lou, H. Z. Sun, W. M. Hu, and T. N. Tan, "Efficient and robust vehicle localization," in *Proc. IEEE Int. Conf. Image Processing*, 2001, pp. 355–358.

[6] J. G. Lou, H. Yang, W. M. Hu, and T. N. Tan, "Visual vehicle tracking using an improved EKF," in *Proc. Asian Conf. Computer Vision*, 2002, pp. 296–301.

[7] K. Takahashi, S. Seki, H. Kojima, and R. Oka, "Recognition of dexterous manipulations from time varying images," in *Proc. IEEE Workshop Motion Non-Rigid Articulated Objects*, Austin, TX, 1994, pp. 23–28.

[8] A. F. Bobick and A. D. Wilson, "A state-based technique to the representation and recognition of gesture," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 1325–1337, Dec. 1997.

[9] A. D. Wilson, A. F. Bobick, and J. Cassell, "Temporal classification of natural gesture and application to video coding," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 1997, pp. 948–954.

[10] F. Bremond and G. Medioni, "Scenario recogntion in airborne video imagery," in *Proc. Workshop Interpretation Visual Motion*, 1988, pp. 57–64.

[11] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer-based video," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, pp. 1371–1375, Dec. 1998.

[12] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 831–843, Aug. 2000.

[13] M. Brand and V. Kettnaker, "Discovery and segmentation of activities in video," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 844–851, Aug. 2000.

[14] M. Yang and N. Ahuja, "Extraction and classification of visual motion pattern recognition," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 1998, pp. 892–897.

[15] U. Meier, R. Stiefelhagen, J. Yang, and A. Waibel, "Toward unrestricted lip-reading," in *Proc. Int. Conf. Multi-Modal Interfaces*, 1999.

[16] Y. A. Ivanov and A. F. Boblic, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 852–872, Aug. 2000.

[17] M. Brand, "Understanding manipulation in video," in *Proc. Int. Conf. Automatic Face Gesture Recognition*, 1996, pp. 94–99.

[18] T. Wada and T. Matsuyama, "Multi-object behavior recognition by event driven selective attention method," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 873–887, Aug. 2000.

[19] R. J. Howarth and B. Hilary, "A analogical representation of space and time," *Image Vis. Comput.*, vol. 10, no. 7, pp. 467–478, 1992.

[20] M. Reiss and J. G. Taylor, "Storing temporal sequences," *Neural Networks*, vol. 4, no. 6, pp. 773–787, 1991.

[21] F. Mulier and V. Cherkassky, "Statistical analysis of self-organization," *Neural Networks*, vol. 8, no. 5, pp. 717–727, 1995.

[22] J. Fernyhough, A. G. Cohn, and D. C. Hogg, "Constructing qualitative event models automatically from video input," *Image Vis. Comput.*, vol. 18, no. 9, pp. 81–103, 2000.

[23] N. Johnson and D. Hogg, "Learning the distribution of object trajectories for event recognition," *Image Vis. Comput.*, vol. 14, no. 8, pp. 609–615, 1996.

[24] N. Johnson, A. Galata, and D. Hogg, "The acquisition and use of interaction behavior models," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, Silver Spring, MD, 1998, pp. 866–871.

[25] C. Stauffer, W. Eric, and L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 747–757, Aug. 2000.

[26] N. Sumpter and A. Bulpitt, "Learning spatio-temporal patterns for predicting object behavior," *Image Vis. Comput.*, vol. 18, no. 9, pp. 697–704, 2000.

[27] J. Owens and A. Hunter, "Application of the self-organizing map to trajectory classification," in *Proc. IEEE Int. Workshop Visual Surveillance*, 2000, pp. 77–83.

[28] T. Kohonen, *Self-Organizing Maps*, 2nd ed. New York: Springer-Verlag, 1997, vol. 30.

[29] ——, *Self-Organization and Associative Memory*, 2nd ed. Berlin, Germany: Springer-Verlag, 1988.

[30] I. Haritaoglu, D. Harwood, and L. S. Davis, "W^4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, pp. 809–830, Aug. 2000.

[31] C. Regazzoni and V. Ramesh, "Special issue on video communications, processing, and understanding for third generation surveillance systems," *Proc. IEEE*, vol. 89, pp. 1355–1367, Oct. 2001.

[32] M. C. Su and H. T. Chang, "Fast self-organizing feature map algorithm," *IEEE Trans. Neural Networks*, vol. 11, pp. 721–733, 2000.

[33] T. Tao, J. R. Gan, and L. S. Yao, "Application of fuzzy neural computing in circuit partitioning," *Chin. J. Comput.*, vol. 15, no. 9, pp. 640–647, 1992.

[34] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[35] H. Shah-Hosseini and R. Safabakhsh, "TASOM: A new time adaptive self-organizing map," *IEEE Trans. Syst., Man, Cybern. B*, vol. 33, pp. 271–282, Apr. 2003.

[36] ——, "Automatic multilevel thresholding for image segmentation by the growing time adaptive self-organizing map," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, pp. 1388–1393, Oct. 2002.

[37] Z. Q. Liu, L. T. Bruton, J. C. Bezdek, J. M. Keller, S. Dance, N. R. Bartley, and C. Zhang, "Dynamic image sequence analysis using fuzzy measures," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 557–571, Aug. 2001.

[38] S. Osowski and T. H. Linh, "Fuzzy clustering neural network for classification of ECG beats," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks*, vol. 5, Como, Italy, July 24–27, 2000, pp. 5026–5030.