Interactive Mining and Semantic Retrieval of Videos

Xin Chen and Chengcui Zhang
115A Campbell Hall, 1300 University Boulevard,
Department of Computer and Information Sciences,
University of Alabama at Birmingham, Birmingham, AL 35294, USA
Phone: (1) 205-934-8669
Email: {chenxin, zhang}@cis.uab.edu

ABSTRACT

With the wide use of monitoring systems, there emerges an ever increasing amount of surveillance videos. Sequential browsing of such videos from the database is time consuming and tedious for users, and thus cannot take full advantage of the rich information contained in video data. In this paper, a general framework for semantic video mining and retrieval is proposed. The framework detects and retrieves semantic events from surveillance videos. It starts by tracking and modeling the trajectories of semantic objects in videos. After that, some general user-interested semantic events are modeled. The goal is to retrieve these semantic events by analyzing the spatiotemporal trajectory sequences. However, since individual users may have their own subjective query targets, these event models may be too general to capture the subjectivity of each individual user. Therefore, in this paper, the mining and retrieval phase is designed to dynamically learn the user's interest by interacting with the user. This technique is called the Relevance Feedback (RF) which is commonly used for Content-based Image Retrieval, but seldom applied to the field of semantic video mining. Due to the spatiotemporal nature of video events, substantial extensions to RF, especially its associated learning mechanisms, are needed to apply it to semantic video mining. The learning framework proposed in this paper bases its structure on the neural network for time series data, which is usually adopted for prediction purposes, and we tailor it to suit the specific needs of spatiotemporal video event mining. In this paper, transportation surveillance videos are used to demonstrate the design details.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications – *data mining.* H.3.3 [**Information Systems**]: Information Search and Retrieval – *relevance feedback, search process.*

General Terms

Algorithms, Design, Experimentation, Human Factors.

Keywords

Multimedia Data Mining, Spatial and Temporal Data Mining, Visual Data Mining, Content-based Video Retrieval, Relevance Feedback, Neural Network.

1. INTRODUCTION

With the development of modern monitoring systems, there is an urgent need for techniques to automatically analyze and retrieve useful information from surveillance videos. Users are no longer satisfied with the video retrieval system that merely provides VCR functionality. They want to query the semantic content instead of sequentially browsing. Content-based retrieval of surveillance videos becomes a challenging and important task. This requires the machine to "learn" and "understand" the semantics of surveillance videos. In light of the status quo of existing techniques, there are two challenges we are yet to meet.

Firstly, current techniques on video content analysis are mostly shot-based. These techniques target at such video types as movies, news broadcasts or sports games. By shot segmentation and key frame extraction [23, 24], the content of the video can be analyzed. However, surveillance videos are different from the above in that they are composed of monotonously running frames with no clear shot boundaries. Therefore, the traditional shot based video content analysis techniques cannot be directly applied to the surveillance videos. The proposed semantic video retrieval framework extracts semantic scenes by analyzing the spatiotemporal relations among moving and still objects in the video.

Secondly, surveillance videos are a special kind of multimedia data. Traditional data mining methods cannot meet the special requirements in understanding the semantic meaning of and extracting knowledge from the raw multimedia data. Specifically, videos are composed of running images, which are dynamic and spatiotemporal in nature. Many researchers have focused on interpreting the video contents based on such low level features as color histograms and mid-level features as object trajectories. Some machine learning techniques are also employed to analyze these low level features [5, 15, 24]. However, it is inherently hard for the machine to understand the video content by simply reading pixels, frames or signals. There is a "semantic gap" between the low level features and the high level human concepts. The proposed framework strives to reduce this "gap" by the "Relevance Feedback" technique.

Relevance Feedback (RF) is a well-known technique in the field of image retrieval. It is used to incorporate the user's subjective perceptions with the learning process [18, 20] for Content-Based Image Retrieval (CBIR). The basic idea of Relevance Feedback is to ask the user's opinion on the retrieval result for a user-specified query target. Based on these opinions, the learning mechanism tries to refine the retrieval result in the next iteration. As a supervised learning technique, Relevance Feedback has been shown to significantly increase the retrieval accuracy.

Besides bridging between low level features and high level human concepts, RF can also progressively gather training samples and customize the retrieval process. It is different from the traditional classification process in machine learning, where prior knowledge is required to compose the "training set" for each class. In the scenario of information retrieval, especially for large multimedia databases, multiple "relevant" and "irrelevant" classes exist according to the different preferences of different users [16]. The data in each "relevant" class may only constitute a very small portion of the entire database. Thus, in a large-scale multimedia database, it is difficult to pre-define a complete set of training sets for all "relevant" classes before query, due to the scarcity of "relevant" samples and the uncertainty of users' interest. With RF, the initial query results are returned based on some heuristics i.e. the models of some general events. The training set for the user's specific query is built up gradually with the help of the user's feedback. Therefore, RF provides more flexibility in information retrieval as it customizes the search engine for the need of individual users.

The core learning algorithm used in this paper is the recurrent neural network for time series data. Video data is a special kind of time series data as it consists of sequences of values or events changing with time. There are a large amount of literatures [3, 9, 17] on applying neural networks in forecasting the behavior of real world time series data, which is popular in such applications as studying the fluctuations of stock market. However, relatively few works [11, 21] have addressed the issue of event detection in time series data with neural network. In this paper, we explore the spatiotemporal models of a neuron for semantic event mining and retrieval from video data.

To summarize, we propose an interactive framework for semantic surveillance video mining and retrieval. This is a general framework in that its components are designed to suit the general needs of surveillance video mining and retrieval. The framework first performs the object tracking and segmentation, which extracts content features and the moving trajectories of objects in the video. Trajectories are then segmented into short sequences. Event models are constructed to model different semantic events and incorporate human knowledge. In the learning and retrieval phase, the technique of Relevance Feedback is incorporated, with which the user provides feedback and the learning algorithm learns from it by depressing the "irrelevant" scenes and promoting the "relevant" scenes. Instead of the pre-defined "expert" knowledge, the individual user's subjective view serves as the guideline for learning. In this framework, recurrent neural network serves as the key learning mechanism to solve an event detection problem. It learns the spatiotemporal characteristics of user-interested video events, which is dynamic rather than static. The proposed framework is especially useful in mining and retrieving data from large video databases, where only raw data is stored. By using users' feedbacks, human knowledge is incorporated into such a database.

While the framework is designed to be of general use and can be tailored to many fields, transportation surveillance videos are used in this study to illustrate the design details. The semantic events in a transportation video database are incidents captured by the surveillance cameras on the road, such as car crash, bumping, U-turn and speeding. Experimental results show the effectiveness of the proposed framework for traffic accident and U-turn detection.

In our previous work [8], a framework for traffic accident retrieval is constructed. The proposed framework in this paper significantly extends the previous one in the following aspects:

- In [8], only one type of incident can be retrieved (i.e., accidents). For this purpose, two event models are constructed, one for single-vehicle accident and another for two-vehicle accident. In the proposed framework in this paper, only one event model for traffic accidents is designed, which is more general compared with the previous two models.
- Besides accidents, the proposed work in this paper is also able to retrieve such events as U-turns. An event model for this type of incidents is constructed to characterize its uniqueness.
- In [8], the .key learning algorithm (i.e. a static feedforword neural network) accepts time series data by expressing them in a spatial manner in the input nodes. In order to further explore the timeliness of the data, a dynamic recurrent neural network is applied in this framework for learning and retrieval.

In the remaining of this paper, Section 2 briefly introduces a semantic object extraction and tracking algorithm for traffic surveillance videos. This section also illustrates the trajectory segmentation technique used in this framework. Section 3 exemplifies the semantic event modeling. Section 4 presents the design details of the learning and retrieval process. Section 5 provides the experimental results. Section 6 concludes the paper.

2. SEMANTIC OBJECT TRACKING AND TRAJECTORY SEGMENTATION

2.1 Automatic Vehicle Tracking and Segmentation

As traffic surveillance videos are the target of our study in this paper, in this section, we provide some background information on the pre-processing of transportation surveillance videos. In our previous work [7], an unsupervised segmentation method called the Simultaneous Partition and Class Parameter Estimation (SPCPE) algorithm, coupled with a background learning and subtraction method, is used to identify the vehicle objects in a traffic video sequence. The technique of background learning and subtraction is to enhance the basic SPCPE algorithm in order to better identify vehicle objects in traffic surveillance videos.

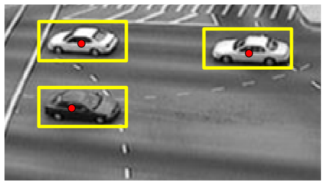

Figure 1. Tracked vehicle segments and their centroids

The framework in [7] also has the ability to track moving vehicle objects (segments) within successive video frames. By distinguishing the static objects from mobile objects in the frame, tracking information can be used to determine the trails of vehicle objects. Figure 1 shows an example of the tracking result of three vehicles. The yellow rectangular area is the Minimal Bounding Rectangle (MBR) of a vehicle segment. ($x_{centroid}$, $y_{centroid}$) are the coordinates of a vehicle segment's centroid represented by a red dot in the figure. It is used for tracking the positions of vehicles across video frames. The last phase of the framework is to classify vehicle objects into different classes such as SUVs, pick-up trucks, and cars, etc. The classification algorithm is based on

Principal Component Analysis.

With the pre-processing, lots of spatiotemporal data is generated. This provides a basis for video mining and retrieval. In this paper, suitable spatiotemporal models for traffic video data are built to further organize, index and retrieve these information.

2.2 Trajectory Modeling and Segmentation

By tracking each moving vehicle in the video, a series of object centroids on successive frames are recorded. We can approximate the trajectory of the vehicle by using the least-square curve fitting. A k^{th} degree polynomial for the curve is:

$$y = a_0 + a_1 x + \dots + a_k x^k \tag{1}$$

Given n centroids on a trajectory, the $k+1$ unknowns $[a_0, a_1, \dots, a_k]$ can be resolved by n equations through minimizing the squared sum of the deviations of the data from the model. The n equations can be represented as:

$$\begin{bmatrix} 1 & x_1 & \dots & x_1^k \\ 1 & x_2 & \dots & x_2^k \\ \dots & \dots & \dots & \dots \\ 1 & x_n & \dots & x_n^k \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \tag{2}$$

The fitted curve represents a rough shape of the moving trajectory. It can be described by only a few polynomial coefficients. The first derivative of a polynomial curve is a tangent vector, which represents the velocities of that vehicle at different time.

Trajectory segmentation is an important task in many applications such as gesture detection in the area of computer vision. Mann et al. [13] proposed a method using dynamic programming. The basic idea of Mann et al. is to divide the trajectory into piece-wise polynomial curves. Meanwhile, the trade off of curve fitting error and the cost of introducing new segments are optimized. It is shown that the global optimization for a single trajectory is reached through dynamic programming. This proposed method is applied to segmenting the motion trajectory of a basketball. Wang and Li [22] employed a spectral analysis method in motion segmentation. The centroid coordinates of moving objects are clustered. In this way, the breakpoints for segmentations are obtained. Anagnostopoulos et al. [1] proposed a global distance based method for trajectory segmentation. This method considers all the trajectories in the database instead of each individual trajectory separately. The optimization is based on the distances of each pair of trajectories.

All the above mentioned work, whether it is distance based or spectral based, whether it views the problem from a set of centroids or from each individual trajectory or even from the perspective of the whole trajectory set, are targeting at an optimization problem. With these methods, the segmented trajectories are suited for tasks such as trajectory retrieval. However, semantic video event retrieval is not as simple as to retrieve similar trajectories from a database. Sophisticated as these trajectory segmentation methods are, they are too general to meet the special needs of such application as incident detection in transportation surveillance videos.

The moving trajectories of vehicles in the surveillance videos are mostly smooth curves, unless there are some accidents with sudden turns or stops. Therefore, we can use some simple methods to detect the curvature of the trajectory and segment the trajectory into pieces such that each piece contains an integral semantic meaning. Bashir et al. [2] proposed to use curvature as a determinant of the concavity or convexity in the curve. The trajectories are divided into segments of smooth motions. Take Figure 2 as an example, each segment represents such a motion.

We follow a similar idea. However, smooth motions are not what we want since these motions signify the normal driving of vehicles and are usually not of the user's interest. Instead, in order to detect interesting events, we perform trajectory segmentation according to the rotation of tangent vectors on the curve.

As shown in Figure 3, the tangent vectors rotate continuously along the curve either clockwise or counterclockwise. The point where the rotation direction changes from clockwise to counterclockwise can be set as a breakpoint for a new segment and vice versa. With such a segmentation scheme, incidents such as U-turns, corner turns or sudden change of driving directions which may imply accidents can be singled out. Further analysis based on trajectory segments instead of the whole trajectory can eliminate the noise in the learning process. Instead of representing the trajectory segments by some scalar parameters [15] [2], such as Fourier Descriptor, parametric curve coefficients or PCA coefficients, we want to keep the spatial-temporal characteristics of the data set. Therefore, in the following sections, each segmented trajectory is still represented in a discrete manner, i.e., by a continuous set of centroids of the moving object. Features are extracted from these centroid sequences and presented to the learning algorithm in a sequential form.

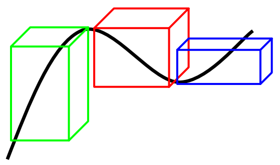

Figure 2. Trajectory segmentation of smooth motions

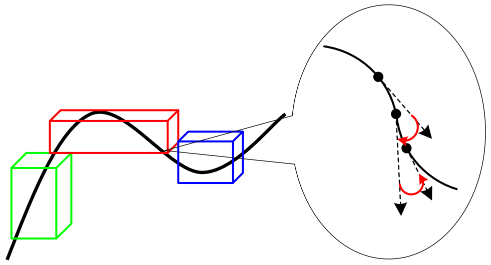

Figure 3. Segmentation of trajectories containing possible interesting events

3. EVENT MODELING

With different event types, different features of the semantic objects can be extracted to build models for specific event types. Take transportation surveillance videos as an example, the general events of interest may include car crashes, illegal U-turns, and overtaking. In this study, we tested our framework on car crashes and U-turns. In this section, two generic models for the above two types of events are built, one for the modeling of the behavior of vehicles in car crashes and another for the modeling of U-turns.

3.1 Traffic Accidents

Under some circumstances, a car accident may involve only one vehicle. Examples are sudden stops or crashes onto side walls in the tunnel. If a vehicle crashes into another vehicle or several vehicles bump into each other, the accident will involve more than one vehicle. In all cases, the focus shall be the sudden change of behavior pattern of individual vehicles. Within each vehicle trajectory, three properties of the vehicle are recorded: velocity, change of velocity, and change of motion vector. Once the sampling rate is known, the velocity at each sampling point can be directly calculated. The change of velocity *Vdiff* at each point can also be easily calculated by subtracting the velocity sampled at the previous sampling point from the current velocity. A motion vector is a vector with its starting point being the centroid of some vehicle at the previous sampling point and the ending point being the centroid of the same vehicle at the current sampling point. As illustrated in the figure below, the change of motion vector is denoted as the angle between the current motion vector and the previous motion vector, $\overrightarrow{M_1}$ and $\overrightarrow{M_2}$. θ is the difference angle between them. Since we only record the absolute angle difference, there is no need to normalize these vectors along the axis.

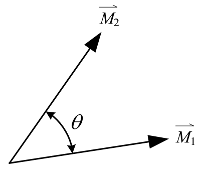

Figure 4. The change of motion vector

Another factor that needs to be taken into consideration is the distances among vehicles as it is a good indication for multi-vehicle accidents. For each vehicle, we record its minimum distance from its nearest vehicle – *mdist* at each sampling point.

As mentioned earlier, some heuristics need to be established in order to process the initial queries. This heuristic model is built based on the observation that the sudden change of velocity and driving direction may indicate an accident. Further, the closer the vehicle is to the other vehicles, the higher the chance of an accident. At the *i*th sampling point, the feature vector of a trajectory segment is α_i = [1/*mdist*$_i$, *vdiff*$_i$, θ_i]. A series of such vectors $\alpha = [\alpha_1, ..., \alpha_n]$ represent the whole trajectory segment.

3.2 U-turns

In events such as U-turns, the main focus is the change of driving direction of vehicles. As in the accident model, θ is the difference angle between two consecutive motion vectors. Besides its absolute value, we shall also look at its angle change direction. We use $\vec{\theta}$ to represent its vectorial property, which is also the feature vector of each sampling point in the trajectory segment for detecting U-turn events. According to our trajectory segmentation method, the angle change direction of each $\vec{\theta}_i$ is the same (either clockwise or counterclockwise. See Figure 4) within each trajectory segment. If we represent the clockwise rotation as positive and counterclockwise as negative, a set of such vectors, $\alpha = [\alpha_1, ..., \alpha_n]$ with $\alpha_i = \vec{\theta}_i$, can be represented by a set of real scalars within $[-\pi, \pi]$. α thus represents the entire trajectory segment. The heuristics for U-turns can therefore be represented by Equation (3).

$$h = \sum_{i=1}^{n} scalar(\vec{\theta}_i) \qquad (3)$$

h is the total angle change within the segment. *scalar* is a function that returns positive angle values for clockwise $\vec{\theta}$ s or negative angle values for counterclockwise $\vec{\theta}$ s. Ideally, a U-turn event occurs when *h* reaches π or $-\pi$. However, since video frames taken by a stationary camera are actually a series of artificial perspective projection images of the real world scenes, which come with the problem of perspective projection, a U-turn cannot always be tracked as a 180^o turn. Besides, some accidents, where a quick and substantial change of driving direction is involved, may also be mistakenly identified as U-turns. Therefore, we cannot rely on this heuristic alone to detect U-turns. Nevertheless, this heuristic can be used as the basis for the initial query, followed by a learning process which is guided by the user.

4. LEARNING AND RETRIEVAL

In this section, we shall present the core learning mechanism and the design details for mining and retrieving semantic events.

4.1 The Learning Mechanism

Neural network simulates human brain in constructing a complex, nonlinear and parallel computing environment. Knowledge is acquired by the network through a learning process. Interneuron connection strengths known as weights are used to store the knowledge. Each neuron is a processing unit. For much of the neural network analysis, the nonlinear model of a neuron is used as shown in Figure 5. $W_{1\times k}$ is a $1 \times k$ weight vector for each input vector x_i whose length is *k* and *b* is the bias. The limitation of this model is that it only accounts for the *spatial* behavior of a neuron.

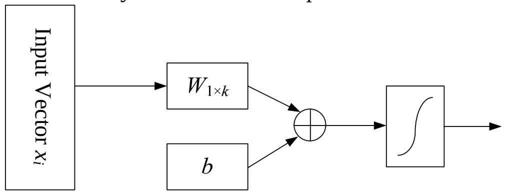

Figure 5. The standard model of a neuron

By extending this model for temporal processing, the neural network method for time series prediction induces the function *f* in a Multilayer Perceptrons (MLP) or Radial Basis Function (RBF) architecture. Time series analysis using neural networks requires the prediction or estimation of x_k based on the preceding *m* observed data points $x_{k-m}, \ldots, x_{k-2}, x_{k-1}$. In prediction, an exact value of x_k is required. Thus, prediction becomes a problem of function approximation which is to find an ***f*** such that:

$$x_k = \boldsymbol{f}(x_{k-m}, \ldots, x_{k-2}, x_{k-1}) \qquad (4)$$

However, for video event mining and retrieval, there is no need to predict an exact value for x_k. Instead, only an indication of whether x_k will be the event of interest is needed. In this case, the problem turns into a classification problem, mapping a temporal sequence onto the classes of “relevant” or “irrelevant”:

$$\boldsymbol{f}_c : (x_{k-m}, \ldots, x_{k-2}, x_{k-1}) \rightarrow c_i \in C \qquad (5)$$

where *C* is the set of all class labels. This can be treated as a special case of function approximation, in which the targets are binary values. Classical linear autoregressive models in modeling time series data are rather limited, since they assume linear relationship among consecutive data series. As illustrated in [10], MLP and RBFs offer an extension to the linear model by using a non-linear function, which can be estimated by such learning and optimization technique as back propagation. Figure 6 shows the basic architecture of a neuron for time series data. Unlike in Figure 5, the temporal relationships among the inputs ($x_{t-1}, \ldots, x_{t-m}$) are considered in Figure 6, i.e. the output signal of the current input will be feed back into the same layer and thus the processing of the next input is at least partially dependent upon earlier computations. This is different from the feedforward networks in that the output of a hidden unit not only can be connected to the units in the next layer but also the units in the same layer or even itself. Therefore, instead of feed *forward*, these units can also feed *back*. This type of network architecture is called Recurrent Neural Network (RNN). As a result, the network can perform more complex computations than static feedforward networks. For example, it is capable of learning temporal pattern sequences.

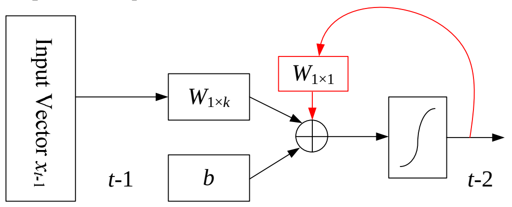

Figure 6. The time series model of a neuron

We can think the recurrent neural network (RNN) structure as a group of interconnected processing units, where any unit may be connected to any other unit or cyclically connected to itself. This network can be represented by a single weight matrix $W_{n\times(n+m)}$, where n is the total number of units with m being the number of external inputs. In this paper, our learning algorithm is based on a recurrent multilayer neural network that incorporates the technique of Relevance Feedback. The structure of the whole network is shown in Figure 7.

In the proposed learning mechanism, the user's feedback is added as a node in the input layer. Other input values include a sequence of vectors $[\alpha_1,...,\alpha_n]$, where α_i is the feature vector at each sampling point as illustrated in Section 3. The number of input nodes corresponds to the number of sampling points with a trajectory segment plus one node for user feedback. The processing units (neurons) in the hidden layer are grouped into n sets as illustrated by the dashed red rectangles (see Figure 7). Each set of hidden units processes an input vector α_i. The output signals of each set of hidden units are not only connected with the output unit of the network but also connected with the next set of hidden units i.e., the ones that process the subsequent input vectors. With this network architecture, the trajectory sequence are spatio-temporally connected and processed. The detailed design decisions as to the input nodes, the numbers of hidden units and hidden layers, weights, and learning algorithms will be discussed in the following section.

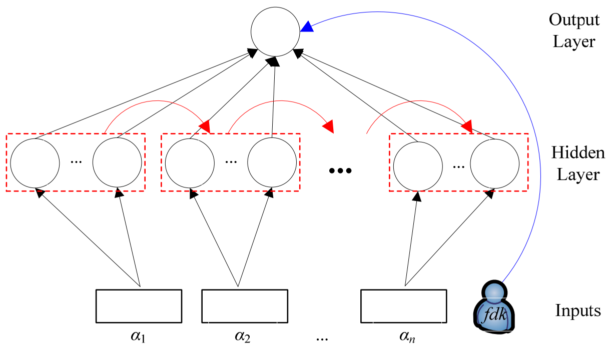

Figure 7. The proposed learning mechanism

4.2 Network Design

4.2.1 Input Nodes

As aforementioned, the length of the trajectory segment determines the number of input nodes. With a sampling rate of 5 frames per sampling point, the trajectory segments are represented by a series of such sampling points. As mentioned above, each input node, excluding the *fdk* node, is a feature vector extracted at its corresponding sampling point according to some event model. For example, in a car-crash model, the feature vector α_i = [1/*mdist*$_i$, *vdiff*$_i$, θ_i] is three dimensional. In a U-turn model, the feature vector $\alpha_i = \vec{\theta}_i$ is one dimensional. The trajectory segments are thus not uniform in length. The longest segment is then chosen to determine the number of input nodes. Some dummy variables (i.e., zeros) are used to pad the segments that are shorter than this length. More design details as to the *fdk* node is included in Section 4.3.

4.2.2 Hidden Layer and Output Layer

Most practical neural networks have used just two or three layers. Four or more layers are rarely used. In our framework, we adopt a two-layer neural network – one hidden layer having sigmoid transfer function and one output layer having linear transfer function. It has been shown that this network architecture can approximate virtually any functions of interest to any degree of accuracy, provided sufficiently many hidden units are available [12]. As demonstrated by our experiments, this architecture can be trained to approximate the function ***f*** well as discussed in Section 4.1. Particularly, the processing units in the hidden layer are interconnected in a sequential manner, which enables them to process a set of time series data.

In our case, there is one unit in the output layer indicating whether the sequence is the desired event or not. The optimal number of units in the hidden layer is a decision hard to make. A large number will reduce the convergence rate of learning and have a higher generalization error due to overfitting and high variance, while a small number cannot guarantee the approximation accuracy. An appropriate number of hidden units are necessary for adequate performance. One rule of thumb [4] is that it shall not exceed twice the size of the input nodes. Suppose the size of one input vector is l, in our platform, we tested on the hidden layer the sizes of nl, 1.5nl and 2nl and found that nl generates the minimum estimated generalization error. n is the number of input vectors i.e. the length of the longest trajectory segment.

4.2.3 Transfer Function

As mentioned above, the transfer function in the first layer (hidden layer) is sigmoid, which is used to introduce nonlinearity.

$$y = \tanh(\sum_i w_i x_i) \tag{6}$$

tanh is the tangent hyperbolic function, a conventional sigmoid function. w_i is a weight for each input or the weight connecting two hidden units.

4.2.4 Initial Weights

Most of the neural networks use random numbers as initial weights. However, since the back propagation is a hill-climbing technique, the randomness of initial weights may result in local optimum. In [6], a multiple linear regression weight initialization method is proposed. It is adopted in our learning algorithm. In this method, the initial weights in the first layer are still uniform random numbers. However, the weights in the second layer are obtained by multiple linear regression.

After the initial weights for the first layer are generated, the input nodes, their corresponding weights and the interconnecting weights of the hidden units are fed to the sigmoid function to get the output values for each neuron in the hidden layer. As mentioned above, in our design, there are *nl* processing units (neurons) in the hidden layer. Suppose these outputs are R_1, $R_{2,\ldots,}R_{nl}$. The second layer use a linear transfer function so that

$$y = \sum_i v_i R_i \qquad (7)$$

where, v_i is weight on the second layer. This is a typical multiple linear regression model. R_i's are regressors. v_i can be estimated using standard regression method. The least square optimization procedure is used in the proposed framework for such a purpose.

4.2.5 Learning Algorithm

Minsky and Papert [14] had shown that one can always derive an equivalent Multilayer Feedforward (MLFF) network from any recurrent neural network, in that the two networks demonstrate the same behavior. In doing so, the recurrent neural network can be unfolded with each time step *t* corresponding to an additional layer. During this unfolding process, the back propagation algorithm used in MLFF network can also be applied to the training of recurrent neural networks. A simple form of this is called “Back Propagation Through Time” (BTT), which is adopted in the proposed learning framework. This method requires that the errors due to feedback connections be accumulated in the weight adjustment process. Just as the regular back propagation, the weights are updated after a complete forward step and a complete backward step. Since the input sequence is finite, the weights are updated after importing the whole sequence, in our case the whole trajectory segment.

The basic form of the back propagation algorithm is computationally expensive and its training may take days or weeks. This has encouraged considerable research on methods to accelerate the convergence rate. As training neural networks to minimize the squared error is simply a numerical optimization problem, some existing numerical optimization techniques have been successfully applied to the training of multilayer perceptrons. Among them are steepest descent, conjugate gradient and Newton’s method. Steepest descent is the simplest algorithm, but is often slow in convergence. Newton’s method is much faster, but requires that the Hessian Matrix and its inverse be calculated. The conjugate gradient is a compromise in that it does not require the calculation of second derivatives, yet it still has the quadratic convergence property. In the proposed framework, we choose the conjugate gradient.

The learning algorithm is provided with a set of examples of proper network behavior: {*I*, *O*}, where *I* is a group of inputs and *O* is the set of the corresponding desired outputs. While each input is applied to the network, the network output is compared to the desired output. The algorithm then adjusts the weights to minimize the mean square error:

$$F(w) = \sum_i (O_i - \Phi(I_i))^2 = e^T e \qquad (8)$$

$\Phi(I_i)$ is the actual output of the network with the current weights. *e* is the error vector and *w* is the weight vector. *e* can be rewritten as $e = O - Gw$, where $Gw = \Phi(I_i)$ and *G* is a matrix. Then,

$$F(w) = O^T O - 2O^T Gw + w^T G^T Gw \qquad (9)$$

Compare this with the following quadratic form:

$$F(w) = \frac{1}{2} w^T Aw - b^T w + c \qquad (10)$$

we can see that these two are the same with $c = O^T O$, $b = 2G^T O$, and $A = 2G^T G$. It can be easily proved that *A* is positive-definite, in that for every nonzero vector *x*, $x^T Ax > 0$. The conjugate gradient algorithm starts from an initial point and searches along the conjugate directions to find the point where the network output error reaches its minimum i.e. to minimize *F*(*w*). Since *A* is positive-definite, the surface represented by Equation (10) is a concave surface that has one single lowest point. *F*(*w*) reaches its minimum at *F’*(*w*) = 0. Therefore, *F’*(*w*) = 0 at this single lowest point. In [19], this is explained in a more intuitive way. If *A* is symmetric, the gradient of the above equation can be reduced to:

$$F'(w) = Aw - b \qquad (11)$$

which makes $Aw = b$ the solution. In this way, the problem of finding the minimum of a quadratic form of Equation 9 equals to solving a linear system. If *A* is not symmetric, we can use the conjugate gradient to find a solution to the system $\frac{1}{2}(A^T + A)w = b$, where $\frac{1}{2}(A^T + A)$ is symmetric.

The conjugate gradient method searches the surface by stepping along the conjugate directions $\{d_{(i)}\}$. By updating the weights along the conjugate directions, we have:

$$w_{i+1} = w_i + s_i d_{(i)} \qquad (12)$$

where s_i is the step size:

$$s_i = r_i^T r_i \big/ d_{(i)}^T A d_{(i)} \qquad (13)$$

r_i is the residual.

$$r_{i+1} = -Ae_{i+1} = -A(e_i + s_i d_{(i)}) = r_i - s_i Ad_i \qquad (14)$$

Two vectors $d_{(i)}$ and $d_{(j)}$ are conjugate if

$$d_{(i)} A\, d_{(j)} = 0 \qquad (15)$$

The Conjugate Gram-Schmidt process provides a simple way to generate a set of conjugate search directions $\{d_{(i)}\}$.

$$d_{(i)} = r_i + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)} \qquad (16)$$

β_{ik} can be obtained by Equation (15) and Equation (14):

$$\beta_{ij} = -\left(r_i^T Ad_{(j)} \big/ d_{(j)}^T Ad_{(j)}\right) \quad (0 \le j < i) \qquad (17)$$

With an initial search point, Equations (11)(12)(13)(14)(15)(16) together form the conjugate gradient method in searching the minimum point in the surface represented by Equation (10).

4.3 Event Mining and Retrieval Process

In the initial query, the user specifies an event of interest as the query target. The ultimate goal is to retrieve those video sequences that contain similar semantic events. At this point, no relevance feedback information is provided by the user. Therefore, no training sample set is available that can be used in learning the pattern of user interested events. In order to provide an initial set of video sequences for the user to provide relevance feedback, for each trajectory segment in the database, we calculate its relevance (or similarity score) to the target query video event according to some event-specific search heuristics.

In the initial retrieval for car-crash events, each sampling point is represented by the square sum of all the three features in the feature vector α_i = [1/*mdist*$_i$, *vdiff*$_i$, θ_i]. The relevance score of a trajectory segment sequence is represented by the maximum value of sampling points within the segment. The retrieval results are returned in descending order of this value. It is assumed that a big velocity change, a sudden change of driving direction and a short distance between two vehicles indicate possible accidents.

For U-turn events, the initial retrieval takes into account the accumulative angle change as indicated in Equation (3), which is used as the heuristic for detecting possible U-turns. We calculate the absolute value of the total angle change *h* within each trajectory segment and sort them in a descending order.

After the initial query, a number of trajectory segment sequences are presented to the user. In our experiment, the top 20 video sequences are returned for feedback. The user identifies a returned sequence as "relevant" if it is of his/her interest; otherwise the user labels it "irrelevant". With this information, a set of training samples are gathered. Each training sample is in the form of [α_1, α_2, …, α_n, *fdk*, *opt*]. α_i's are the feature vectors at consecutive sampling points used to model a trajectory segment sequence. *fdk* is subtracted by a small positive number ε if the user marks it "irrelevant", otherwise it is incremented by ε. The value of ε is set to 0.2 in our case as we assume there are no more than 5 rounds of user-feedback and the normalized input value is within a range of [-1, 1]. *opt* is the desired output with the value of one for a "relevant" sequence or zero for an "irrelevant" sequence. These training samples are fed into the neural network based learning framework (Figure 7), which learns and models users' interest and refines the retrieval result in the subsequent runs of the retrieval-feedback process. After several iterations, the "relevant" sequences and their common features are encouraged by incrementing their *fdk* values, while the "irrelevant" sequences are panelized by decreasing their *fdk* values. It is shown in our experiment that, with this interactive learning technique, the retrieval results are improved through iterations.

5. EXPERIMENTS

5.1 System Overview

The overall flow of the whole system proposed in this paper is as follows - The raw video is analyzed by segmenting and tracking semantic objects (vehicles) in it. After tracking, the object trajectories are modeled with the curve fitting technique. Trajectories are then segmented into smaller pieces for the purpose of learning and retrieval. In our experiment, we test its performance on retrieving traffic accidents and U-turns from traffic surveillance videos. The corresponding event models are built and the feature vectors at each sampling point are extracted. When the user specifies a query type (i.e., accidents or U-turns), the system performs an initial query based on some heuristics as discussed in Section 4.3, and returns the initial retrieval results to the user. The user responds to each returned trajectory segment sequence by giving his/her feedbacks. The learning mechanism in the system will then learn from these feedbacks and refine the retrieval results in the next iteration. The whole process goes through several iterations until a satisfactory result is obtained.

Figure 8 shows the interface for the user to provide feedback information. In this example, the user chooses to query for U-turn events. The top 20 video sequences are returned to the user at each iteration. The user can play the retrieved sequence. If the user thinks a sequence is a U-turn event, that sequence will be selected. This is equal to labeling the video sequence "relevant". As shown in the interface, 5 sequences (in orange rectangles) are labeled "relevant" given a U-turn query.

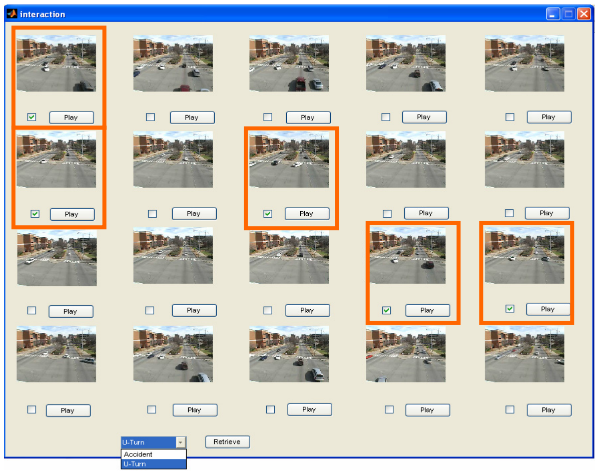

Figure 8. The user interaction interface

5.2 System Performance

5.2.1 Query Accidents

The proposed framework is tested on two video clips for accidents. The first one is taken in a tunnel and contains 2504 frames in total. The second one is taken by a real-life traffic surveillance video at a road intersection in Taiwan and contains 592 frames. The sampling rate is 5 frames/sampling points. After trajectory segmentation, 82 trajectory segments extracted from the first video and 69 trajectory segments from the second clip.

The proposed framework is compared with the traditional weighted relevance feedback method. In this method, each feature in the feature vector α_i has a weight. The initial round of retrieval is the same as that of the proposed framework. That is to say, the initial weights of the three features are all 1s and the square sum of the features is computed as the relevance score. With the user's relevance feedback, the feature vectors of all relevant trajectory sequences are gathered. The inverse of the standard deviation of each feature is computed and used as the new weight for that feature in the next round. In our experiment, we found that some large weights can introduce bias in computing relevance scores and hence affect the retrieval accuracy. Therefore, it is necessary to normalize these weights. We first tried to linearly normalize these weights to the range of [0, 1]. However, the problem with this method is that a weight that equals zero will always eliminate the corresponding feature. We then tried another method -- the percentage of each weight among the total weight is used as the normalized weight. In our experiment, the latter outperforms both the linear normalization and no normalization at all.

Besides weighted RF, the proposed framework is also compared with the feedforward neural network and One-class SVM. With feedforward neural network, there is no recurrent behavior in the network. Each trajectory segment is presented to the network as if there are no temporal relations among the sampling points in the trajectory segment. A trajectory segment sequence is expressed spatially in the network in that the feature vector of each sampling pint corresponds to an input node and there is no temporal relation among the hidden units processing these inputs. In other words, the temporal relation among inputs is not used explicitly in the learning process. Therefore the processing of one input node is not affected by the processing of any proceeding input node.

With One-class SVM, the temporal relation among sampling points in the trajectory segment sequence is also not used explicitly. The trajectory segment is simply represented by a vector whose length is the number of sampling points in the segment. With the user's feedback, the "relevant" sequences are gathered and fed into One-class SVM. Since "irrelevant" sequences are different in various ways, they are not considered to be in a single class. Instead, "irrelevant" sequences are treated as outliers of the "relevant" class. This justifies the usage of the One-class SVM instead of the binary SVM.

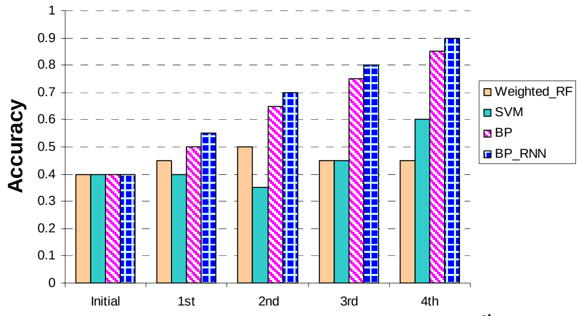

Figure 9. The retrieval accuracies for the 1st clip

Five rounds of relevance feedback are performed - Initial (no feedback), First, Second, Third, and Fourth. At each iteration, the top 20 sequences are returned to the user. In a large-scale information mining and retrieval system, since there is no prior knowledge as to the total number of "correct" results given a user's query, it is not applicable to use traditional data mining measurements such as precision and recall. Instead, we use the "accuracy" measure for such a purpose, which is defined as the percentage of all the "relevant" sequences within the top *n* (e.g. *n*=20) returned sequences. Figure 9 shows the retrieval accuracies within the top 20 sequences for the first video clip after Initial, First, Third, and Fourth round of iterations. "Weighted_RF" represents the weighted relevance feedback method, "SVM" is the One-class SVM, "BP" is the feedforward neural network and "BP_RNN" is the proposed framework.

It can be gleaned from Figure 9 that the initial accuracies of the four methods are the same since the same retrieval algorithm is used in the initial round. After that, the proposed framework performs much better than the other methods in that the accuracy values increase steadily from 40% to 90%. "SVM" and "BP" also have higher performance gains than that of "Weighted_RF".

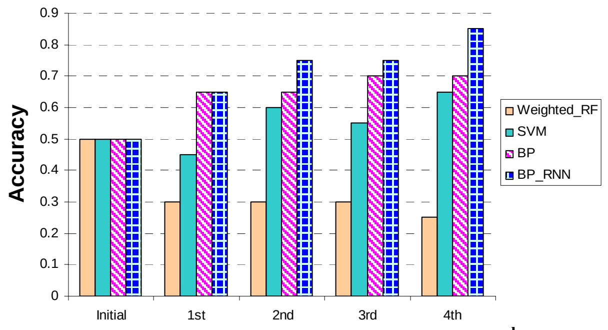

Figure 10. The retrieval accuracies for the 2nd clip

Most of the accidents in the first clip only involve one vehicle. The video was taken in a tunnel and features some accident scenes where speeding vehicles lost control and hit on the sidewalls of the tunnel. In the second clip, all the accidents occurred at a road intersection and often involve two or more vehicles. The retrieval results are compared with that of the weighted RF, One-class SVM and feedforward neural network in Figure 10. Although the 35% accuracy gains with the proposed framework is not as high as that for the first clip, it is far better than that of the other methods. Specifically, in "Weighted_RF", performance degradation occurs right after the initial iteration. "SVM" and "BP" achieved 15% and 20% accuracy increase, respectively.

5.2.2 *Query U-turns*

We test the U-turn events on the video taken at a major intersection at a frame rate of 30 frames per second. There are altogether 13261 frames. After trajectory segmentation, 377 trajectory segments are extracted. Since the feature vector in U-turn event model is one dimensional, it does not make sense to change the weight of this single feature. Therefore, we cannot directly compare the proposed framework with the weighed relevance feedback. In the experimental results shown in Figure 11, only the results of One-class SVM (SVM) and feedforward neural network (BP) are presented for comparison.

The proposed framework reaches 70% accuracy rate at the 4th iteration. "BP" has a total of 60% accuracy rate and "SVM" only has a 40% accuracy rate at the end of the query. The initial retrieval accuracy is only 25%, which further affects the first and the second iterations. This is due to the imperfection of vehicle segmentation and tracking. For example, in tracking a vehicle driving along the X axis, supposedly, the x-coordinate of the vehicle centroid shall increase in each subsequent frame. However, the current segmentation method cannot guarantee to 100% correctly locate the exact centroid of a vehicle in each frame. This inaccuracy may cause the vehicle centroid's next x-coordinate smaller than that in the current frame. In this case, the vehicle "seems" to be driving backward. As a result, an angle change of 180^o may be incorrectly computed due to this inaccuracy in vehicle segmentation and tracking. This significantly hinders the retrieval accuracy in the initial retrieval. However, after several rounds of learning and feedback, the accuracy rate of the proposed framework gradually picks up.

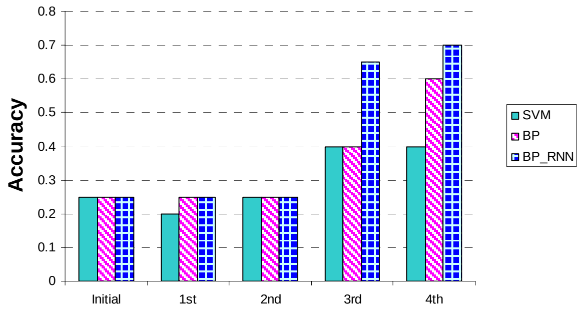

Figure 11. The retrieval accuracies for the U-turns

Ideally, all the video clips in a transportation surveillance video database shall be mined and retrieved as a whole. However, in order to do that, it requires that we normalize all the video clips taken at different locations with different camera parameters. Those parameters, such as camera angles and camera positions, are necessary for normalization. Unfortunately, these metadata are missing in our experimental videos. Therefore, at the current stage, the retrieval is performed independently for each group of videos taken by the same camera at the same location. Our next immediate step is to collect our own transportation surveillance videos and normalize them before storing them into the database.

6. CONCLUSION

In this paper, an interactive semantic video mining and retrieval framework for general surveillance videos is proposed. Given a set of raw video data, the semantic objects are tracked and the corresponding trajectories are modeled. After that, trajectories are segmented and stored into the database. Some general event models are then constructed. In the learning and retrieval phase, the user provides feedback to the relevance for each returned video sequence. The learning algorithm then refines the retrieval results with the user's feedbacks. This framework successfully incorporates the Relevance Feedback technique in mining video data, which is rarely studied in video mining and retrieval. For learning and retrieval, recurrent neural network is adapted to fit the specific needs of event detection for time series data. Our experimental results on live transportation surveillance videos demonstrate its effectiveness.

In future work, we will normalize videos before storing them into the database. To reduce the influence of imprecise vehicle segmentations, we will blur the difference between consecutive centroid coordinates. This may be achieved by sampling the fitted curve instead of using the original sampling points (real centroids). Currently, the framework only supports query by specified event types. This will be extended to include query by example, query by sketches and the customized query types.

7. ACKNOWLEDGEMENT

The work of Chengcui Zhang was supported in part by SBE-0245090 and the UAB ADVANCE program through the sponsorship of the National Science Foundation.

8. REFERENCES

[1] Anagnostopoulos, A., Vlachos, M., Hadjieleftheriou, M., Keogh, E., and Yu, P. S. Global distance-based segmentation of trajectories. In *Proceedings of KDD* (Philadelphia, Pennsylvania, USA, 2006).

[2] Bashir, F., Khokhar, A., and Schonfeld, D. A hybrid system for affine-invariant trajectory retrieval. In *Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval* (New York, NY, USA, 2004), 235-245.

[3] Bengio, S., Fessant, F., and Collobert, D. A connectionist system for medium-term horizon time series prediction. In *Proceedings of the International Workshop Application Neural Networks to Telecoms*, (1995), 308-315.

[4] Berry, M. J. A. and Linoff, G., *Data mining techniques*. John Wiley & Sons, New York, 1997.

[5] Bobick, A. F., Pentland, A. P., and Poggio, T. Vsam at the MIT media laboratory and CBCL: Learning and understanding action in video imagery PI report 1998. In *Proceedings of the DARPA Image Understanding Workshop*, (1998), 85-91.

[6] Chan, M.-C., Wong, C.-C., and Lam, C.-C. Financial time series forecasting by neural network using conjugate gradient learning algorithm and multiple linear regression weight initialization. *Computing in Economics and Finance*, 61 (2000).

[7] Chen, S.-C., Shyu, M.-L., Peeta, S., and Zhang, C. Learning-based spatio-temporal vehicle tracking and indexing for transportation multimedia database systems. *IEEE Trans. on Intelligent Transportation Systems*, 4, 3 (2003), 154-167.

[8] Chen, X. and Zhang, C. An interactive semantic video mining and retrieval platform--application in transportation surveillance video for incident detection. In *Proceedings of the IEEE International Conference on Data Mining* (Hong Kong, China, 2006).

[9] Davey, N. Time series prediction and neural networks. *Intelligent and Robotic System*, 31 (2000).

[10] Dorffner, G. Neural network for time series processing. *Neural Network World*, 6, 4 (1996), 447-468.

[11] Gao, D., Kinouchi, Y., Ito, K., and Zhao, X. Neural networks for event extraction from time series: A backpropagation algorithm approach. *Future Generation Computer Systems*, 21 (2005), 1096-1105.

[12] Hornik, K. M., Stinchcombe, M., and White, H. Multilayer feedforward networks are univeral approximators. *Neural Networks*, 2, 5 (1989), 359-366.

[13] Mann, R., Jepson, A. D., and El-Maraghi, T. Trajectory segmentation using dynamic programming. In *Proceedings of the 16th International Conference on Pattern Recognition* (Waterloo University, Ontario, Canada, 2002), 331-334.

[14] Minsky, M. and Papert, S., *Perceptrons*. MIT Press, Cambridge, MA, 1969.

[15] Naftel, A. and Khalid, S. Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. *Multimedia Systems*, 12, 1 (2006), 227-238.

[16] Nakazato, M., Dagli, C., and Huang, T. S. Evaluating group-based relevance feedback for content-based image retrieval. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'03)* (Spain, 2003), 599-602.

[17] Patterson, D. W., Chan, K. H., and Tan, C. M. Time series forecasting with neural nets: A comparative study. In *Proceedings of the International Conference on Neural Network Applications to Signal Processing (Singapore, 1993)*, 269-274.

[18] Rui, Y., Huang, T. S., and Mehrotra, S. Content-based image retrieval with relevance feedback in mars. In *Proceedings of the International Conference on Image Processing*, (1997), 815-818.

[19] Shewchuk, J. R. *An introduction to the conjugate gradient method without the agonizing pain*. Carnegie Mellon University, Technical CS-94-125, August 1994.

[20] Su, Z., Zhang, H. J., Li, S., and Ma, S. P. Relevance feedback in content-based image retrieval: Bayesian framework, feature subspaces, and progressing learning. *IEEE Trans. on Image Processing*, 12, 8 (2003), 924-937.

[21] Tsien, C. L. Event discovery in medical time-series data. In *Proceedings of the AMIA Symposium*, (2000), 858-862.

[22] Wang, H. and Li, H. A spectral clustering approach to motion segmentation based on motion trajectory. In *Proceedings of the International Conference on Multimedia and Expo.* (Baltimore, MD, USA, 2003), 793-796.

[23] Zhang, H. J., Low, C. Y., Smoliar, S. W., and Zhang, D. Video parsing, retrieval and browsing: An integrated and content-based solution. In *Proceedings of ACM Multimedia* (San Francisco, CA, USA, 1995), 15-24.

[24] Zhang, W., Wu, X., Kamijo, S., and Sakauchi, M. Semantic video database system with semi-automatic secondary-content generation capability. *Multimedia Tools and Applications*, 30, 1 (2006), 27-54.