An Interactive Semantic Video Mining and Retrieval Platform – Application in Transportation Surveillance Video for Incident Detection

Xin Chen and Chengcui Zhang
Department of Computer and Information Sciences, University of Alabama at Birmingham
Birmingham, Alabama, 35294 USA
{chenxin, zhang}@cis.uab.edu

Abstract

Understanding and retrieving videos based on their semantic contents is an important research topic in multimedia data mining and has found various real-world applications. Most existing video analysis techniques focus on the low level visual features of video data. However, there is a "semantic gap" between the machine-readable features and the high level human concepts i.e. human understanding of the video content. In this paper, an interactive platform for semantic video mining and retrieval is proposed using Relevance Feedback (RF), a popular technique in the area of Content-based Image Retrieval (CBIR). By tracking semantic objects in a video and then modeling spatio-temporal events based on object trajectories and object interactions, the proposed interactive learning algorithm in the platform is able to mine the spatio-temporal data extracted from the video. An iterative learning process is involved in the proposed platform, which is guided by the user's response to the retrieved results. Although the proposed video retrieval platform is intended for general use and can be tailored to many applications, we focus on its application in traffic surveillance video database retrieval to demonstrate the design details. The effectiveness of the algorithm is demonstrated by our experiments on real-life traffic surveillance videos.

Keywords: Multimedia data mining, spatio-temporal data mining, data mining applications

1. Introduction

The rapid growth of multimedia resources has generated an urgent need for techniques to process and analyze this special kind of data. Traditional machine learning methods cannot meet the special requirements in understanding the semantic meaning of and extracting knowledge from raw multimedia data. Image, video, audio data are specific multimedia data types. It is inherently hard to make the machine understand the meaning of these data by only reading pixels, frames or signals. There exists a "semantic gap" between the low level features and the high level semantic meaning. Therefore, it is necessary that human provide some guidance to the machine. Various techniques exist for such a purpose. In the field of image retrieval, there is a well-known technique – Relevance Feedback (RF). It is used to incorporate the user's subjective concept with the learning process [1, 2, 3] 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. The process iterates until a satisfactory result is obtained for the user. For example, if the user wants to find all the images containing "tiger" and the learning algorithm returns several "wall" images, the user can label these unwanted images "irrelevant". In this way, the user's knowledge is "transferred" to the learning algorithm, which uses it to further refine the retrieval result. As a supervised learning technique, Relevance Feedback has been shown to significantly increase the retrieval accuracy.

In this paper, we borrow the idea of Relevance Feedback from CBIR and apply it to the retrieval of video data. The purpose of the proposed platform is to automatically learn and retrieve semantic scenes from videos according to the user's query. The motivation is to use RF as a bridge between multimedia processing and information retrieval. It is different from 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 perfect 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

Proceedings of the Sixth International Conference on Data Mining (ICDM'06)
0-7695-2701-9/06 $20.00 © 2006 IEEE

generally categorized 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.

However, semantic video retrieval is even harder than the problem in image retrieval. Video is composed of running images (frames). We need to not only study the content of individual frames (still images), but also consider the temporal relations among them. With content-based image analysis, the content features of each object (e.g. its spatial location, texture features, shape, and color) in a frame can be extracted. From the perspective of each such object, its moving trajectory in consecutive frames is a kind of spatio-temporal data. In this study, the aim of semantic video retrieval is to extract semantic scenes by analyzing the spatio-temporal relations among moving and still objects in the video. The interactive video retrieval platform proposed in this paper first performs the object tracking and segmentation, which extracts content features and moving trajectory of each object in the video. 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 "relevant" scenes.

The core learning algorithm used in this paper is 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 [2, 4, 9] on applying neural network 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 [5, 6] have addressed the issue of event detection in time series data with neural network. In this paper, we explore the spatio-temporal models of a neuron for semantic event mining and retrieval from video data.

In summary, we propose a platform that can interact with the user in mining and retrieving user-interested semantic events from video data. Instead of pre-defined "expert" knowledge, individual user's subjective view serves as the guideline for learning. In this platform, the key learning mechanism, neural network, is designed to learn the spatio-temporal characteristics of user-interested video events, which is dynamic rather than static. Since we are not dealing with a prediction problem, the "prediction" neural network for time series data is adjusted to suit the needs of spatio-temporal semantic event classification for video data.

The proposed platform 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 then incorporated into such a database. Although this is a video retrieval platform of general use that can be tailored to many fields, we focus mainly on its application in traffic surveillance video database retrieval throughout the paper to demonstrate the design details. The semantic events in such a database are incidents captured by the surveillance video on the road, such as car crash, bumping, U-turn and speeding. Experimental results show the effectiveness of the proposed platform in traffic incident detection.

In the rest of the paper, Section 2 briefly introduces a semantic object extraction and tracking method for traffic videos. Section 3 exemplifies the semantic event modeling. Section 4 demonstrates the design details of the learning and retrieval process. Section 5 provides the experimental results. Section 6 concludes the paper.

2. Semantic object tracking

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 processing of transportation surveillance videos. In our previous work [15], 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 used to enhance the basic SPCPE algorithm in order to better identify vehicle objects in traffic surveillance videos. The Minimal Bounding Rectangle (MBR) of the vehicle as well as its centroid coordinates ($x_{centroid}$, $y_{centroid}$) are recorded to track the positions of vehicles across video frames.

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

2.2. Trajectory modeling

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:

$$y = a_0 + a_1 x + \ldots + a_k x^k \quad (1)$$

Given n centroids on a trajectory, the k+1 unknowns [a_0, a_1, ..., 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:

$$\begin{bmatrix} 1 & x_1 & \ldots & x_1^k \\ 1 & x_2 & \ldots & x_2^k \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & \ldots & x_n^k \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ \ldots \\ a_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix} \quad (2)$$

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. The figure below demonstrates the fitted curve of a group of centroids by a 4th degree polynomial. The small diamonds around the curve represent the original centroids.

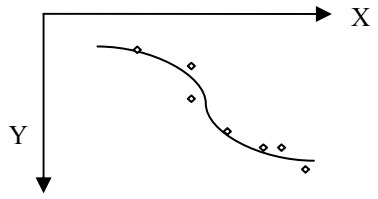

Figure 1. An example of polynomial curve fitting

3. Semantic event modeling

With different event categories, different properties 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, etc. In this study, we tested our framework on car crash events.

3.1. Single vehicle model

Examples of single vehicle accidents include sudden stops, crashes onto side walls in a tunnel, and crashes into crowd in car races, etc. For accidents involving a single vehicle, we can often observe sudden changes in vehicle behavior patterns. Along a vehicle trajectory, three properties of the vehicle are recorded: velocity, change of velocity, and change of motion vector. After the sampling rate is specified, the velocity at each sampling point can be directly calculated from the fitting curve as shown in Figure 1. The change of velocity at each point can also be easily obtained by subtracting the velocity at the previous sampling point from the current velocity. A motion vector is a two-dimensional vector, or movement, that shows the difference from the vehicle coordinate position in the previous sampling point to the coordinates in the current sampling point. As illustrated in the figure below, the change of motion vector is actually the angle between the current motion vector ($\overrightarrow{M_2}$) and the previous motion vector ($\overrightarrow{M_1}$).

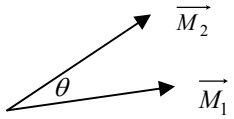

Figure 2. The change of motion vector

In Figure 2, $\overrightarrow{M_1}$ and $\overrightarrow{M_2}$ are two consecutive motion vectors. θ is the absolute angle difference between them. Since we only record the angle difference, there is no need to normalize these motion vectors along the axis. It is worth mentioning that only the absolute value of angle difference is used for accident detection, as it is sufficient to reflect the sudden change of moving directions during an accident, while the actual value of angle difference is useful for the reasoning of other types of events such as U-turns. As mentioned in Section 1, some heuristics need to be established in order to perform the initial queries. The main heuristic we used is based on the observation that the sudden change of velocity and/or driving direction may indicate an accident. At the i^{th} sampling point, the extracted property vector of a vehicle object is $\alpha_i = [v_i, vdiff_i, \theta_i]$. A series of such vectors $\alpha = [\alpha_1, \ldots, \alpha_n]$ are used to represent the trajectory of each vehicle. It is worth mentioning that this single vehicle model may also be adjusted to detect two-vehicle accidents, U-turns, speeding and any other events that involve the abnormal behavior of a single vehicle.

3.2. Two-vehicle model

Another category of accidents involves two or multiple cars. Thus, we need a model to capture the relationship between every two vehicle objects. In such a model, both the properties of an individual vehicle and the interaction between two vehicles shall need to

be considered. Two properties of the trajectory are extracted. One is velocity and another is the interaction feature between two vehicles as proposed in [10]. There are four steps in getting this feature.

1. First, rotate the motion vectors ($\overrightarrow{M_1}$, $\overrightarrow{M_2}$) of two vehicles by aligning $\overrightarrow{M_1}$ to X axis.
2. Second, get the difference vector $\overrightarrow{M_{diff}}$ = $\overrightarrow{M_1}$ - $\overrightarrow{M_2}$ as shown in Figure 3.

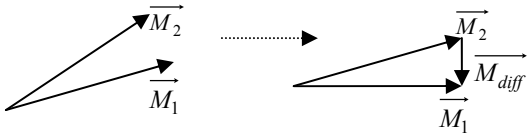

Figure 3. Motion vector rotation and subtraction

3. Third, divide the difference vector $\overrightarrow{M_{diff}}$ by the distance between the centroids of the two vehicles. It is obvious that, the smaller the distance, the bigger the chance of an accident.
4. Fourth, quantize the result from Step 3. The figure below shows the quantization schema.

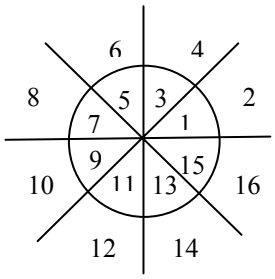

Figure 4. The quantization schema for the two-vehicle interaction feature

As shown in the quantization schema above, the interaction feature has 16 categories depending on which region the normalized vector $\overrightarrow{M_{diff}}$ after Step 3 falls into. If two vehicles are close to each other, their normalized difference vector $\overrightarrow{M_{diff}}$ will have a bigger chance of falling into the regions outside the circle (see Figure 4) because the distance between their centroids is relatively small. The regions outside the circle with labels (2, 4, 6, 8, 10, 12, 14, 16) implicate high accident potential (HAP).

If any two vehicles ever appear simultaneously in a series of consecutive frames, the start and the end of these common frames are recorded. Their trajectories within these common frames are analyzed to extract a sequence of interaction features on each sampling point. In this way, for each pair of vehicles that ever appear together, we have $\alpha = [\alpha_1, ..., \alpha_n]$, where α_i = [$v1diff_i$, $v2diff_i$, cat_i]. $v1diff_i$ and $v2diff_i$ are the velocity changes of the two vehicles at the sampling point i. cat_i is the category number from the result of quantization. It is worth mentioning that cat_i is categorical data with partial ordering because the even-number categories (2, 4, 6, 8, 10, 12, 14, 16) indicate a higher accident potential than the odd-number categories (1, 3, 5, 7, 9, 11, 13, 15).

Similar to the single-vehicle model, the two-vehicle model can be extended to model other events that involve two or more vehicles such as overtaking, multi-car crashes and bumping.

4. Semantic event retrieval

4.1. Platform overview

Neural network simulates human brain in constructing a complex, nonlinear and parallel computing environment. Knowledge is acquired by the network through a learning process. For most of the neural network analysis, the nonlinear model of a neuron is used. The limitation of this model is that it only accounts for the *spatial* behavior of a neuron.

By extending this model for *temporal* processing, the neural network method for time series prediction induces the function f in a standard Multilayer Perceptrons (MLP) or Radial Basis Function (RBF) architecture. This method is often called the sliding window technique as the N-tuple input slides over the full training set. A time series is a sequence of vectors depending on time t: x_t, t = 0, 1, …. Figure 5 shows an example of sliding window for time series data. In this example, the input is a 6-tuple extracted from time series data by sliding a window of size 6 one step a time along the time axis t.

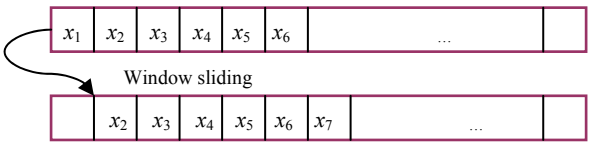

Figure 5. An example of sliding window

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}, …, x_{k-2}, x_{k-1}. m is the window size. In prediction, an exact value of x_k is required. Thus, prediction becomes a problem of function approximation which is to find an $\boldsymbol{f}$ that:

$$x_k = \boldsymbol{f}(x_{k-m}, \ldots, x_{k-2}, x_{k-1}) \quad (3)$$

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":

$$\boldsymbol{f}_c : (x_{k-m}, \ldots, x_{k-2}, x_{k-1}) \rightarrow c_i \in C \quad (4)$$

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 [11], 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 techniques as back propagation and conjugate gradient.

In this paper, our platform is based on a feed forward multilayer neural network that incorporates the technique of Relevance Feedback. The structure of the whole network is shown in the Figure 6 below. There are temporal relationships among the inputs (x_{t-1}, …, x_{t-m}) in Figure 6, which are selected by a sliding window.

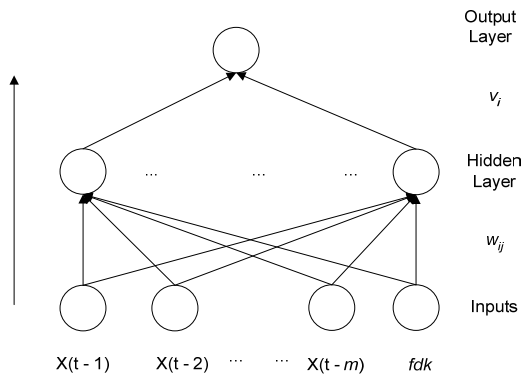

Figure 6. The proposed platform

In the proposed platform, the user's feedback is added as a node in the input layer. Other input values include a sequence of vectors selected from $\alpha = [\alpha_1, ..., \alpha_n]$ by sliding the window with a size of m, where α_i is the feature vector at each sampling pint as illustrated in Section 3. The number of input nodes corresponds to the window size plus one node for user feedback. The detailed design decisions as to the window size, the numbers of hidden units and hidden layers, weights, and learning algorithms will be described in the following section.

4.2. Network design

4.2.1. Window size and input nodes. The size of the sliding window determines the number of input nodes. If the window is too small, the sequence in the window is too "short" to show the overall pattern of a certain event. On the other hand, if the window is too big, the sequence would be too "general" to reveal the core event characteristic with too many noise data around. In order to find the optimal solution, some researchers suggest an incremental search on the size of the window. As in [12], a method called false nearest neighbor is proposed.

The advantage of video data over regular time series data is that it can be visualized, thus one can have a concrete idea of how the data is organized and flows over time. Therefore, in our platform, the size of the window can be simply decided by the typical length of an event as it can be acquired by the counting the number of frames covering this event. Take car crash as an example, the typical length in terms of number of frames for this kind of events is very short i.e. about 15 frames. Given a sampling rate of 5 frames, 3 sampling points are needed to depict a car crash event. Thus, the window size is 3 in our case.

As mentioned in the above section, each input node, excluding the *fdk* node, is a feature vector extracted according to the event model. In a single-vehicle model, the feature vector $\alpha_i = [v_i, vdiff_i, \theta_i]$ is three dimensional. In a two-vehicle model, the feature vector is $\alpha_i = [v1diff_i, v2diff_i, cat_i]$. cat_i is categorical data with (1, 2, …, 16) signifying one of the 16 categories. Since categorical data cannot be simply treated as numerical values and fed into one input node, we use a 16-dimensional vector to represent each category. Each of such vectors contains 15 zeros and 1 one in the position corresponding to the category it falls into. These zeros are the inserted dummy variables. For example, the following vector indicates category 3. This makes the input vector for each input node 18-dimensional.

0	0	1	0	… …	0

Figure 7. Categorical data representation

4.2.2. Hidden layer. Most practical neural networks have used just two or three layers. Four or more layers are rarely used. In our platform, 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 [7]. As demonstrated by our experiments, this architecture can be trained to approximate the function ***f*** well as discussed in Section 4.1.

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 hidden layer is a decision hard to make. A large number will reduce the convergence rate of learning and have a high 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 [8] is that it shall not exceed twice the size of the input nodes. Suppose the size of input is *l,* in our platform, we tested on the hidden layer the sizes of 0.5*l*, *l*, *1.5l* and 2*l* and found that *l* generates the minimum estimated generalization error.

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

$$y = \tanh(\sum_i w_i x_i) \tag{5}$$

tanh is the tangent hyperbolic function, a conventional sigmoid function. w_i is a weight for each input x_i.

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 [13], a multiple linear regression weight initialization method is proposed. It is adopted in our platform. 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 is generated, input nodes and their corresponding weights are fed to the sigmoid function to get the output values for each neuron in the hidden layer. Suppose these outputs are R_1, R_2, ..., R_l. The second layer use a linear transfer function so that

$$y = \sum_i v_i R_i \tag{6}$$

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 our platform for such a purpose.

4.2.5. Search algorithm. 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 of the algorithm. As training feedforward 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 platform, we choose the conjugate gradient.

The search 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. As 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:

$$\begin{aligned} F(w) &= \sum_i (O_i - \Phi(I_i))^2 \\ &= e^T e \end{aligned} \tag{7}$$

$\Phi(I_i)$ 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,

$$F(w) = O^T O - 2O^T Gw + w^T G^T Gw \tag{8}$$

Compare this with the following quadratic form:

$$F(w) = \frac{1}{2} w^T Aw - b^T w + c \tag{9}$$

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 search along the conjugate directions to find the point where the network output error reaches its minimum i.e. to minimize $F(w)$. In the quadratic form of Equation 8, $F(w)$ reaches its minimum at $F'(w) = 0$ since A is positive-definite. In [14], this is explained in a more intuitive way. As illustrated in Figure 8, the lowest point is where $F'(w) = 0$.

If A is symmetric, the gradient of the above equation can be reduced to:

$$F'(w) = Aw - b \tag{10}$$

which makes $Aw = b$ being the solution. In this way, the problem of finding the minimum of a quadratic form of Equation 8 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.

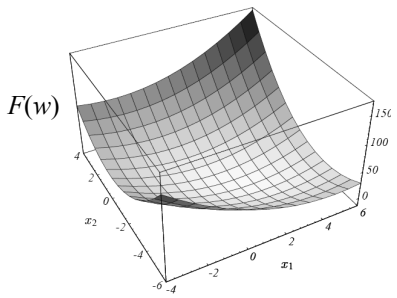

Figure 8. The plot of Equation 9 with *A* being positive-definite

The conjugate gradient method searches the surface in Figure 8 by stepping along the conjugate directions $\{d_{(i)}\}$. By updating the weights along the conjugate directions, we have:

$$w_{i+1} = w_i + s_i d_{(i)} \tag{11}$$

where s_i is the step size:

$$s_i = \frac{r_i^T r_i}{d_{(i)}^T A d_{(i)}} \tag{12}$$

r_i is the residual.

$$\begin{aligned} r_{i+1} &= -Ae_{i+1} \\ &= -A(e_i + s_i d_{(i)}) \\ &= r_i - s_i A d_i \end{aligned} \tag{13}$$

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

$$d_{(i)}\, A\, d_{(j)} = 0 \tag{14}$$

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

$$d_{(i)} = r_i + \sum_{k=0}^{i-1} \beta_{ik} d_{(k)} \tag{15}$$

β_{ik} can be obtained by Equation (14) and Equation (13):

$$\beta_{ij} = -\frac{r_i^T A d_{(j)}}{d_{(j)}^T A d_{(j)}} \quad (0 \le j < i) \tag{16}$$

With an initial search point, Equations (10)(11)(12)(13)(14)(15) together form the conjugate gradient method in searching the minimum point in Figure 8.

4.3. Event mining and retrieval with relevance feedback

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 video sequence in the database, we calculate its relevance (or similarity score) to the target query video event according to some event-specific search heuristics.

Suppose the user wants to query single car accidents such as crashes onto roadside walls, crashes into crowd in car races, or multi-car accidents which involve dramatic changes of single car trajectories. In this case, the feature vector $\alpha_i = [v_i, vdiff_i, \theta_i]$ from the single car model is used. In the initial retrieval, the maximum value of $vdiff_i \times \theta_i$ at the first sampling point of each video sequence in the database is calculated, and the retrieval results are displayed to the user in descending order of this value. It is assumed that a big velocity change and a sudden change of driving direction signify possible accidents.

For two-vehicle accident model, the initial retrieval takes into consideration the interaction feature cat_i. If cat_i is an even number (2, 4, 6, …, 16), there might be a higher possibility of an accident for the reason aforementioned. For each pair of vehicles whose cat_i is even, its $v1diff_i \times v2diff_i$ is calculated. $v1diff_i$ and $v2diff_i$ are the velocity changes of the two vehicles at the sampling point i. The maximum value of such at the first sampling point of each video sequence is used to determine the accident potential of that sequence. The initial retrieval results are returned to the user according to this value in a descending order.

After the initial query, a certain number of video sequences are presented to the user. In out experiment, the top 20 video sequences are returned for the user's 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 at hand, a set of training samples can be gathered. Take single car event as an example, each training sample is in the form of [α_{t-2}, α_{t-1}, α_t, *fdk, opt*]. α_{t-2}, α_{t-1}, and α_t are the feature vectors at the three consecutive

sampling points used to model a video sequence. *fdk* is zero if the user marks it as "irrelevant", otherwise it is incremented by a small positive number 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 [0, 1]. *opt* is the desired output with the value of one for a "relevant" sequence or zero for a "irrelevant" sequence. These training samples are then fed into the neural network based learning framework (see Figure 6), which learns and models users' interest and refines the retrieval result in the following user-feedback iterations. After several iterations, the "relevant" sequences and their common features are encouraged by incrementing their *fdk* values, while the "irrelevant" sequences are punished by keeping their *fdk* value minimal. 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 figure below shows the overall flow of the whole system proposed in this paper.

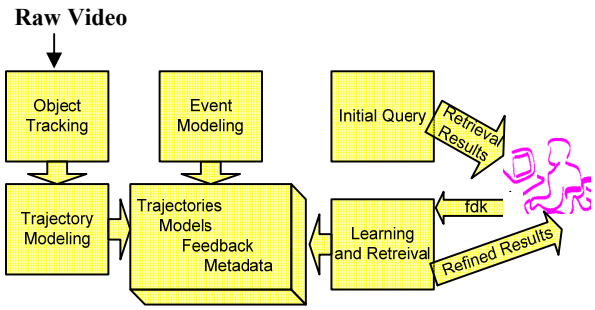

Figure 9. The system overview

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. Two event models, the single-vehicle model and the two-vehicle model, are built and the feature vectors of vehicle objects at each sampling point are extracted. When the user submits a query, the system performs an initial query based on some heuristics, depending on the event type, and returns the initial retrieval results to the user. The user responds to these results 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.

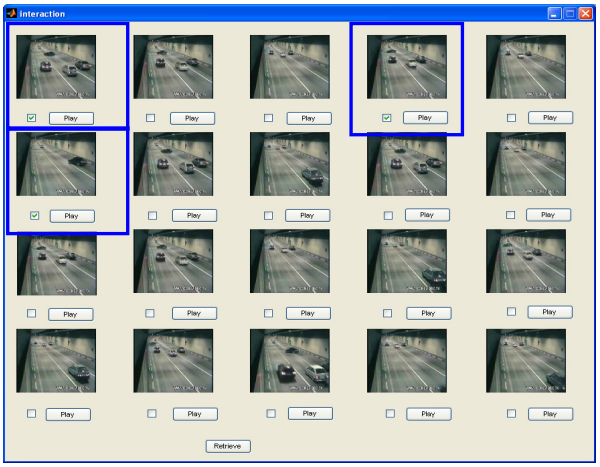

Figure 10. The user interaction interface

Figure 10 shows the interface for the user to provide feedback information. The top 20 sequences are returned to the user in each iteration. The user can play the retrieved sequences, in which the trajectories of possible problematic vehicles are traced by yellow or black dots. Some retrieved accident examples will be provided in Section 5.2. If the user thinks the marked trajectory in a certain sequence is what he/she wants, the sequence will be selected. This is equal to labeling the sequence "relevant". As shown in the interface, three sequences (in blue rectangles) are labeled "relevant" in a single car accident query.

5.2. Sample retrieved events

Figure 11. An example of single-vehicle accidents

Figure 11 is a traffic surveillance video in a tunnel. The car bounded in the white rectangle crashes into the side wall. The yellow dots illustrate the trajectory of the car in problem.

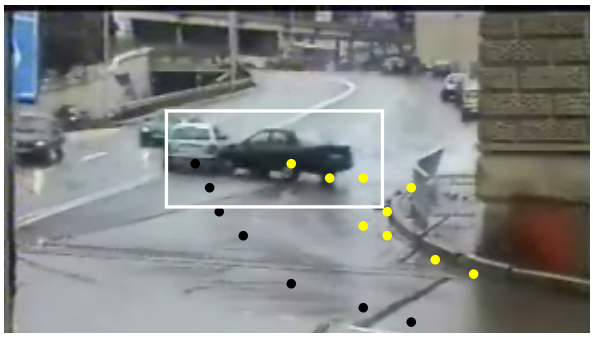

Figure 12. An example of two-vehicle accidents

In Figure 12, the two cars in the white rectangle bump into each other. The trajectory of the car on the right is traced by yellow dots. The trajectory of the other car is marked by black dots.

5.3. System performance

For the retrieval of single-vehicle events, a real-life traffic surveillance video taken in a tunnel is tested with the proposed system. Five video clips containing accident scenes are extracted with 2504 frames in total. These video clips were taken at a frame rate of 25~29 frms/sec. Our sampling rate is 5 frames per sampling point and the window size is 3, i.e. three sampling points in a window. After sampling and window sliding, there are altogether 497 sequences (15 frames each) in the database.

Six rounds of user relevance feedback are performed for single-car accident query - Initial (no feedback), First, Second, Third, Fourth, and Fifth. In each iteration, the top 20 video sequences are returned to the user. In a large-scale information mining and retrieval system (e.g. a global web-based information retrieval system), since there is no prior knowledge as to the total number of "correct" objects given a user's query, the traditional data mining performance measurements such as precision and recall are not applicable. Therefore, we use the measure of accuracy for such a purpose. The accuracy rates with different scopes, i.e. the percentage of relevant sequences within the top 5, 10, 15 and 20 returned sequences are calculated. Figure 13 shows the accuracies after the Initial, First, Third, and Fifth round of iterations.

It can be gleaned from Figure 13 that the retrieval accuracy increases across iterations with the incorporation of the user's feedback. In the third iteration, the total accuracy has already reached 85% i.e. 17 out of 20 returned sequences are regarded as "relevant" by the user. If the user is still not satisfied with the results and wants to continue the process, he/she is able to find 18 relevant sequences after the fifth iteration, making the total retrieval accuracy 90%. Notice that after the third iteration, the accuracy among the top 15 returned results has reached 100%.

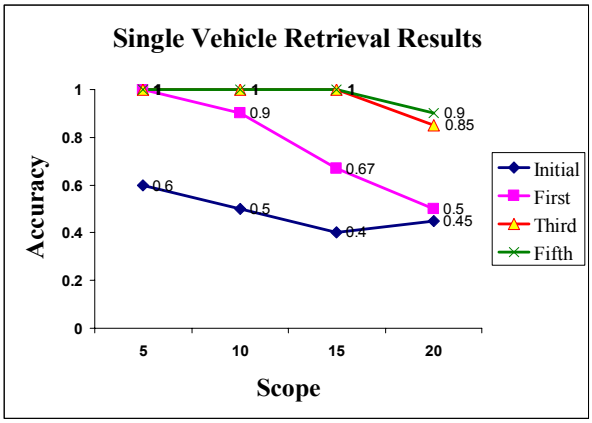

Figure 13. The accuracies of single-vehicle event retrieval across iterations

For two-vehicle events, a real-life traffic surveillance video taken at a street corner is tested. Two clips containing accident scenes are extracted with 1275 frames in total. After sampling and window sliding, 35 sequences which contain at least two vehicles across the time span of that sequence are extracted and stored in the database. Figure 14 shows the retrieval results after the Initial, First, Second, and Third round of iterations.

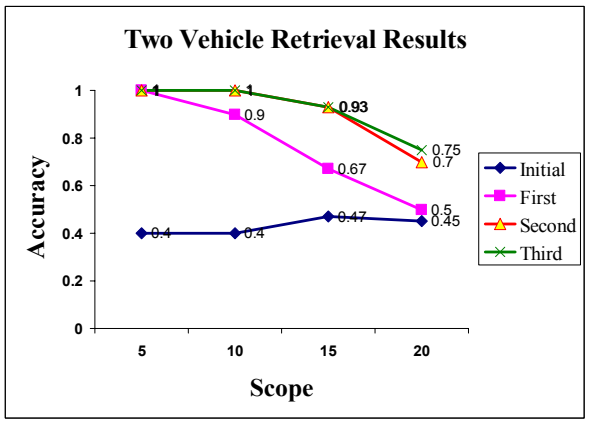

Figure 14. The accuracies of two-vehicle event retrieval across iterations

For two-vehicle events, since there are only two accident scenes in the surveillance video, the sequences involved in accidents are sparse and constitute only a small portion of the entire data set. Thus, its accuracy in each iteration is smaller than that of the single-vehicle event retrieval because the problem gets more difficult when the positive class is

small. However, it still can be seen that the accuracy increases through iterations with relevance feedback.

6. Conclusions and future work

In this paper, an interactive semantic video mining and retrieval platform is proposed. Given a set of raw videos, the semantic objects are tracked and the corresponding trajectories are modeled and recorded in the database. Some spatio-temporal event models are then constructed. In the learning and retrieval phase, with the top returned sequences in each iteration, the user provides feedback to the relevance of each video sequence. The learning algorithm then refines the retrieval results with the user's feedbacks. This platform successfully incorporates the Relevance Feedback technique in mining video data, which is a well studied topic in Content Based Image Retrieval but needs significant extensions (e.g. the modeling and incorporation of spatio-temporal characteristics) when applied to video data retrieval. For learning and retrieval, the neural network for time series prediction is adapted to fit the specific needs of event identification for video data. The platform shows its effectiveness as demonstrated by our experiments on live transportation surveillance videos.

In the future work, more general event models will be constructed and tested with the proposed platform. Currently, the platform only supports the user's query by specified event types (single-vehicle or two-vehicle events). We will extend this to include query by example, query by sketches, and a customized combination of different query types.

7. Acknowledgement

The work of Chengcui Zhang was supported in part by SBE-0245090 and the UAB ADVANCE program of the Office for the Advancement of Women in Science and Engineering.

8. References

[1] Y. Rui, T.S. Huang, and S. Mehrotra, "Content-based Image Retrieval with Relevance Feedback in MARS," in *Proceedings of the International Conf. on Image Processing*, 1997, pp. 815-818.

[2] N. Davey, S.P. Hunt, and R.J. Frank, "Time Series Prediction and Neural Networks," *Journal of Intelligent and Robotic System,* Vol. 31, 2000.

[3] Z. Su, H.J. Zhang, S. Li, and S.P. Ma, "Relevance Feedback in Content-based Image Retrieval: Bayesian Framework, Feature Subspaces, and Progressing Learning," *IEEE Transactions on Image Processing*, Vol. 12, No. 8, pp. 924-937, 2003.

[4] D.W. Patterson, K.H. Chan, and C.M. Tan, "Time Series Forecasting with Neural Nets: a Comparative Study," in *Proceedings of the International Conference on Neural Network Applications to Signal Processing,* Singapore, pp. 269-274, 1993.

[5] D. Gao, Y. Kinouchi, K. Ito, and X. Zhao, "Neural Networks for Event Extraction from Time Series: a Backpropagation Algorithm Approach," *Future Generation Computer Systems*, vol. 21, pp.1096-1105, 2005.

[6] C.L. Tsien, "Event Discovery in Medical Time-Series Data," In *Proceedings of AMIA Symp*, pp. 858-862, 2000.

[7] K. M. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, no. 5, pp. 359-366, 1989.

[8] M.J.A. Berry and G. Linoff, Data Mining Techniques, NY: John Wiley & Sons. pp. 323, 1997.

[9] S. Bengio, F. Fessant, and D. Collobert, "A Connectionist System for Medium-Term Horizon Time Series Prediction," in *Proceedings of International Workshop Application Neural Networks to Telecoms*, pp. 308-315, 1995.

[10] S. Kamijo, Y. Matsushita, and I. Katsushi, "Traffic Monitoring and Accident Detection at Intersections," *IEEE Transactions on Intelligent Transportation Systems,* Vol. 1, No. 2, pp. 108-118, June 2000.

[11] G. Dorffner, "Neural Network for Time Series Processing," *Neural Network World,* 6(4), pp. 447-468, 1996.

[12] R.J. Frank, N. Davey, and S.P. Hunt, "Time Series Prediction and Neural Networks," *Journal of Intelligent and Robotic Systems*, 31, pp. 91-103, 2000.

[13] M.-C. Chan, C.-C. Wong, and C.-C. Lam, "Financial Time Series Forecasting by Neural Network Using Conjugate Gradient Learning Algorithm and Multiple Linear Regression Weight Initialization," *Computing in Economics and Finance*, No. 61, 2000

[14] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method without the Agonizing Pain," Carnegie Mellon University Technical Report: CS-94-125, August 1994.

[15] S.-C. Chen, M.-L. Shyu, S. Peeta, and C. Zhang, "Learning-Based spatio-temporal vehicle tracking and indexing for transportation multimedia database systems", *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 3, pp. 154-167, September 2003.

[16] M. Nakazato, C. Dagli, and T. S. Huang, "Evaluating Group-based Relevance Feedback for Content-based Image Retrieval," in *Proceedings IEEE International Conference on Image Processing (ICIP'03)*, Spain, 2003, Vol. 2, pp. 599-602.