CENTER FOR THE COMMERCIALIZATION OF INNOVATIVE TRANSPORTATION TECHNOLOGY
(CCITT OR "SEE IT")

USDOT UNIVERSITY TRANSPORTATION CENTER

NORTHWESTERN UNIVERSITY

Innovation Gap Research Grant Pre-Proposal
2007 - 2008

FINAL REPORT

VIDEO TRAFFIC ANALYSIS FOR ABNORMAL EVENT DETECTION

PI: A. K. Katsaggelos, Professor
Co-PI: S. Tsaftaris, Research Assistant Professor
Co-PI: Y. Wu, Associate Professor

Students: Derin Babacan and Fan Jiang

Department of EECS

DISCLAIMER *The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.*

1. Introduction

We propose the use of video imaging sensors for the detection and classification of abnormal events to be used primarily for mitigation of traffic congestion. Successful detection of such events will allow for new road guidelines; for rapid deployment of various transportation and safety agencies; for interactive displays that alert drivers to, for example, slow down or speed up, move to a different lane, or, alter their driving behavior, so as to reduce the probability of traffic congestion or the occurrence of more abnormal events. Deciding on additional road guidelines or proper display information, can be accomplished either via computer simulations or experimentation with field implementations. We extended in-house developed algorithms to detect and track vehicles in video sequences and analyze their trajectories. Through analysis we will classify their trajectories independently but also considering vehicle interactions, into abnormal and normal events. A main objective of the analysis is to determine how each type of abnormal event affects subsequent traffic, and serves as a predictor of congestion build up. Towards this task we will identify each type of abnormal event by implementing a supervised classifier, and built models to describe them and their effect (over time).

In the following we explain our results. They can be summarized as follows:

a) Phase 1(a): Collected video data from various sources

b) Phase 1(b): Implemented a vehicle tracking algorithm

c) Phase 2: Trajectory analysis and classification into abnormal and normal categories

2. Phase 1(a): Data acquisition

Initially we were planning to obtain data from the University of Illinois at Chicago Artificial Intelligence Lab, where Prof. Nelson is the director. However, after communication with the lab's manager we were informed that the frame rate of the video was below the required rate for tracking. The frame rate of the UIC Lab was approximately 1 frame per 2 mins and for our purposes the required minimum frame rate was 20fps. Therefore we had to consider other avenues to acquire the data. We approached iDOT but they mentioned that they do not archive any video data and they do not have any plans to do so. We considered recording data from the top of public buildings but due to privacy issues and the limited field of view we decided against this route.

Our current video database consists of videos collected from:

1. 6 hours of video from the Washington State Department of Transportation. [Highway traffic, low quality, severely saturated, limited field of view]

2. the traffic monitoring websites of the City of Istanbul (http://tkm.ibb.gov.tr/en-EN/yolDurumu/Kameralar.aspx) and Caltrans (http://www.video.dot.ca.gov/) [Highway and city traffic, low resolution, severely compressed, requires real-time video feed capture]

3. the 2001-2004 databases of the International Workshop on Performance Evaluation of Tracking and Surveillance (PETS). [Mostly parking lot views, high quality and resolution, limited to no compression, not highway traffic]

4. the NGSIM aerial traffic video dataset (http://ngsim.camsys.com/). [Highway traffic from multiple cameras, no compression, good resolution, vehicle position ground truth. On the other hand, NGSIM also provides ground truth tracking results, which can immediately be used in our trajectory analysis module.]

3. Phase 1(b): Vehicle Detection and Tracking

3.1. Method Overview

The approach utilized in this work to analyze traffic videos is using the following module pipeline:

1. Background Subtraction
2. Blob detection
3. Blob tracking
4. Trajectory Post-processing

Overall, the algorithm works by detecting the entering objects to the scene, and tracking them throughout the video. The input to the algorithm is the raw video data of a site. The algorithm then performs the following steps: First, a statistical background model of the scene is populated using the first few frames of the video. This background model collects the statistics of the background of the recorded scene such as road, trees, buildings, etc. This model is then used to distinguish the objects of interest (vehicles, people) from the surroundings. In the next step, the detected foreground parts of the scene are grouped together by a neighborhood analysis, and a filtering process is applied to remove noise and misdetections. The objects of interest obtained at the end of this step are then tracked throughout the video until they leave the scene. This tracking provides the trajectory of the moving object, which is then smoothed by a filter to remove jittering in the detected trajectories. Finally, the extracted trajectories are analyzed by a clustering algorithm which performs the abnormality detection.

In the following sections we will provide details on each of the modules.

3.2. Background subtraction

This stage is the first stage of the algorithm, and discriminates the foreground from the background objects. We have mainly utilized the algorithm in [1], which is robust and has high detection performance, although it has higher computational requirements than other algorithms in the literature. However, the performance achieved at this step greatly affects the subsequent stages, and therefore a robust and accurate algorithm is desired.

The background of a video is generally defined as objects staying constant (such as buildings and roads) or moving in small amounts (such as trees) in the scene. The background subtraction model first builds a histogram-based statistical model of the video image background for each pixel. This model collects the color co-occurrences in the video and applies quantization to decrease the computational load. This selection of the color co-occurrences increases the accuracy of the algorithm in the presence of moving background objects.

The algorithm then employs a Bayes decision rule for classification of background and foreground. The Bayes decision rule facilitates the use of posterior probabilities of the feature vectors, and calculates the probability of a feature vector (color co-occurrence) belonging to a specific class. The classification is then made easily comparing the probabilities. The algorithm also employs morphological filtering operations to remove incorrectly classified foreground objects, that is, after the initial classification, a filter is applied to the result to remove scattered isolated points which are very likely to be noise.

An important property of any effective background subtraction algorithm is to adapt the background model to the changing conditions in the scene. These conditions include the change of time during the day and sudden changes in illumination (such as passing clouds blocking the sun), among others. To account for these conditions, the implemented background subtraction algorithm updates the background model at each video frame as follows: After the classification, new features from the background of the scene are collected and used to update the model using a weighted average filter (an infinite impulse (IIR) filter), that is, both new and old features have an effect on the background model but with different degrees. Combined with the Bayes decision rule mentioned above, both gradual and sudden changes in the scene can be accounted for and used to update the statistics. Finally, the algorithm maintains a reference image of the background. The block diagram of the algorithm is shown in Figure 1. An example result of the background subtraction module can be seen in Figure 2.

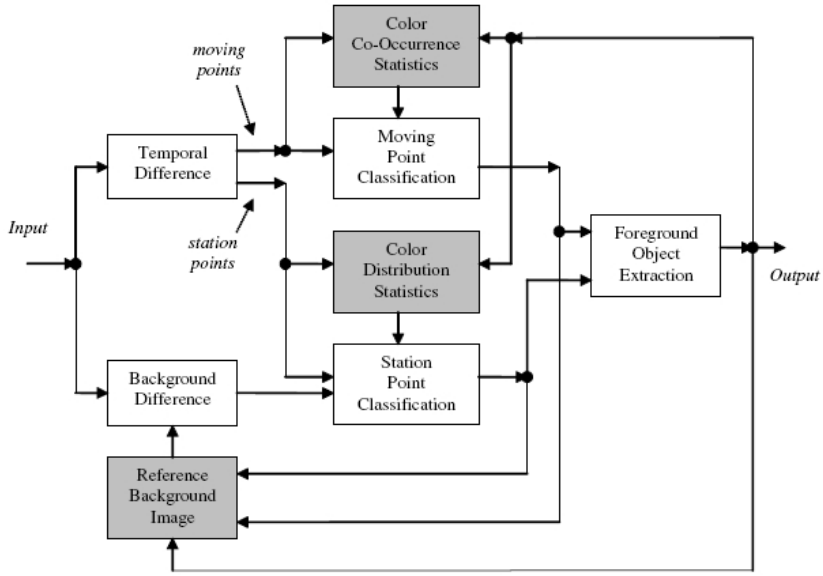


Figure 1 Block diagram of the background subtraction algorithm (from [1])

Figure 2 Left: An input video frame, Right: The result of the background subtraction module (foreground pixels are marked with white)

3.3. Blob Detection

After a background model is initialized, a classification of the detected foreground objects is carried out to differentiate between existing objects and objects just entering the scene. Existing objects are already being tracked by the system (explained below in Section 5). Entering objects must be detected to be sent for further processing.

The background subtraction model supplies the pixels detected as foreground. In the blob detection model, these pixels are grouped together by utilizing a contour detection algorithm. The contour detection algorithm groups the individual pixels into disconnected classes, and then finds the contours surrounding each class. Each class is marked as a blob. These detected blobs are then checked by their size and small blobs are removed from the algorithm to reduce false-detections.

Finally, the detected blobs are compared with the existing blobs from former video frames. This is performed in two stages. First, if a new detected blob intersects with a blob from the previous frame, this blob is removed since it is already being processed by the tracking model. Note, however, that an existing blob may move more than its size in the next frame so that there will be no intersection. To detect these cases, in the second step, the detected blob is compared with the existing trajectories being analyzed by the algorithm. This comparison is made by fitting a linear model to a few points of the trajectory and checking if the location of the blob fits to this model. This simple procedure is computationally effective and efficient in removing blobs detected falsely as new.

Figure 3 shows an example input-output pair of the blob detection module. The input, the result of the background subtraction module is shown on the left, whereas the output of the blob detection module is shown on the right.

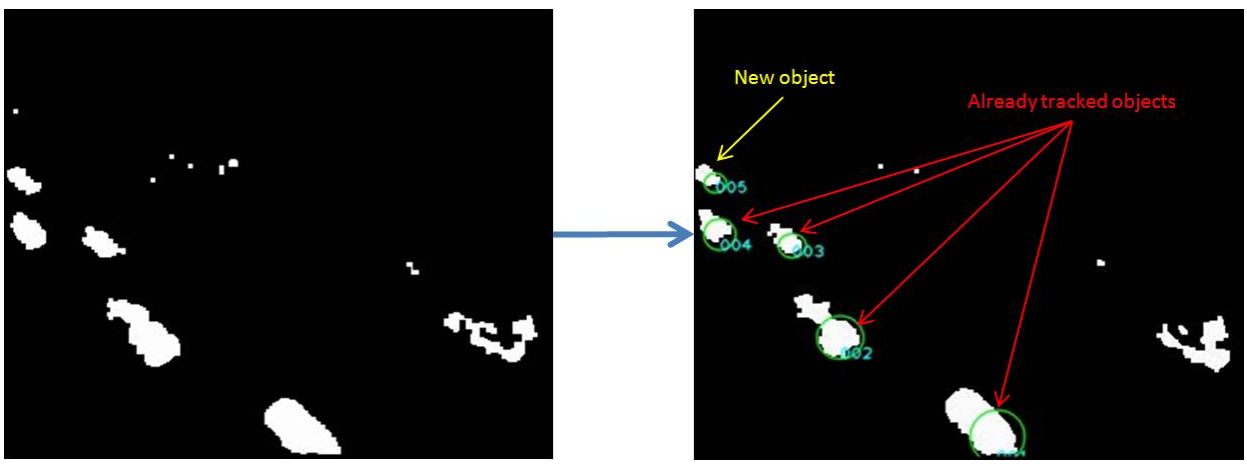


Figure 3 Example input and output of the blob detection module.

3.4. Blob Tracking

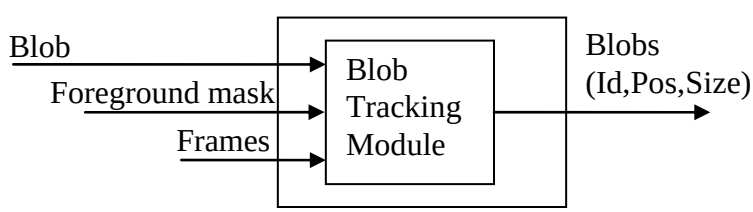


Figure 4 Blob tracking module pipeline

The blob tracking module is one of the most important stages in the pipeline. As shown in Figure 4, this module receives a blob with its position and size, the foreground mask, and the unprocessed video frames as inputs and provides the new location of the blob in the current video frame.

The literature on tracking objects in a video is very rich, and it is still a very widely investigated research topic due to its high practical value. Despite the fact that there are many complex tracking algorithms, the success rate of tracking has not reached the level where highly accurate tracking results are obtained for all tracking applications. In many cases, the tracking algorithm is chosen and tuned for specific needs. For instance, an algorithm that is accurate for tracking people may not provide accurate results when tracking cars.

We have compared several tracking algorithms for the specific task of tracking cars in traffic, and finally we have utilized a hybrid algorithm where two tracking algorithms work in parallel and their results are analyzed to achieve the final tracking result. In the following we first briefly present the individual tracking methods and then explain how their results can be combined.

Connected Component (CC) Tracking [2]. This tracking algorithm utilizes the blobs detected by the blob detection module presented in the previous section. It operates by building a 2D connection graph between the detected foreground pixels and forms the blobs. Then, the tracking procedure identifies which connected component in the previous video frame corresponds to the blob in the current frame. This correspondence is computed using the position, color and texture information. Different important situations must be accounted for by this tracking procedure. For instance, there can be many connected components corresponding to a blob in the previous frame, due to errors in foreground detection or touching objects may be moving apart from each other. These problems are largely avoided by weighing the position of the connected components over color and texture information, which might be misleading. Since the position of the blob in the previous frame is filtered by a post-processing filter (presented in the next section), it provides more accurate information than color and texture information.

There are two main characteristics of this method: First, since it relies on the foreground blob detection methods (Sections 3 and 4), it provides very accurate tracking information. However, if the foreground detection algorithm cannot detect the blob, this tracking method will also fail since it will not have the connected component corresponding to that blob. Second, since the number of the detected connected components in a frame is relatively small, the algorithm is computationally efficient than other tracking methods.

MSPF Tracking. This algorithm incorporates a hybrid model between the popular mean shift and particle filter algorithms. Let us first present how these individual trackers perform, and then present how they are combined to achieve higher quality tracking results.

The mean shift (or kernel based) tracker [3,4] is a deterministic tracking algorithm. It utilizes the color histogram of the tracked object and aims in finding the location that has the most similar color histogram. To find the location, mean shift iterations are applied using the color distribution of the

tracked object in the current frame, and the degree of similarity between the color distributions in two locations is calculated using the Bhattacharyya distance [3,4].

On the other hand, a particle filter is a sampling-based stochastic estimation method, where a group of particles is collected from the observations, and is used to build and propagate a distribution model. The sampled particles are weighted by their likelihoods and used to model the posterior distribution of the movement of the object. The new location of the object can then be found using the estimated mean value of this distribution. The estimated distribution is estimated at each frame by sampling new particles.

Mean shift trackers are computationally very efficient, but they are error-prone in the case of occlusions and high-speed movements. On the other hand, particle filters are very effective in tracking in cluttered environments and they can recover very well after lost tracks. To achieve good tracking performance, one can combine the mean shift tracker with particle filters [5]. This hybrid tracker, called MSPF, combines the advantages of both algorithms and has achieved impressive tracking results.

In MSPF, a particle filter is utilized to sample from the observation, but a lower number of particles is used. Then, using the mean shift algorithm, these particles are shifted to the estimated new location. The new location is determined like in the original mean shift algorithm by calculating the most likely place utilizing the Bhattacharyya distance in the search space.

MSPF is reported to be a very robust tracking method and is currently one of the state-of-the-art general purpose tracking algorithms.

Combining CC tracking with MSPF: CCMSPF. As mentioned above, one of the main problems in tracking is collision. Collision occurs when two tracked objects collide and start being tracked as a single object. It is different than occlusion where a single tracked object moves behind a background object and the tracker loses track. Collisions are hard to detect and recover from, and special methods have to be performed to avoid them.

In our work, we mainly utilized the CC tracking algorithm while the MSPF tracker is used to resolve the collisions. The algorithm scans the detected blobs and determines if two blobs are intersecting within a two frame period. In case there is an intersection, the algorithm switches to the MSPF tracker to calculate the new positions. Therefore, the algorithm operates by combining the results of the two trackers in the case of possible errors.

This hybrid approach combines the advantages of the CC tracker (computational efficiency) and the MSPF tracker (robustness). Overall, this approach yields very accurate tracking results. However, since this approach relies mainly on the CC tracker, it possesses some of its disadvantages. A major drawback is that the CC tracker uses the result of the background subtraction and blob detection modules without modification, and therefore the errors made at these stages propagate to the tracking stage and affect the tracking performance. A common error found in our experiments is that while an object is being tracked, the foreground detection module might not detect it in the current video frame, and the object

cannot be further tracked and will be lost. To avoid such problems, we have also included the original MSPF tracker in the software to be used as the secondary tracking algorithm.

An example tracking result with the CCMSPF tracker is shown in Figure 5. The previous frame is shown on the left, and the current video frame is shown on the right. The arrows denote the associations of the objects in these frames.

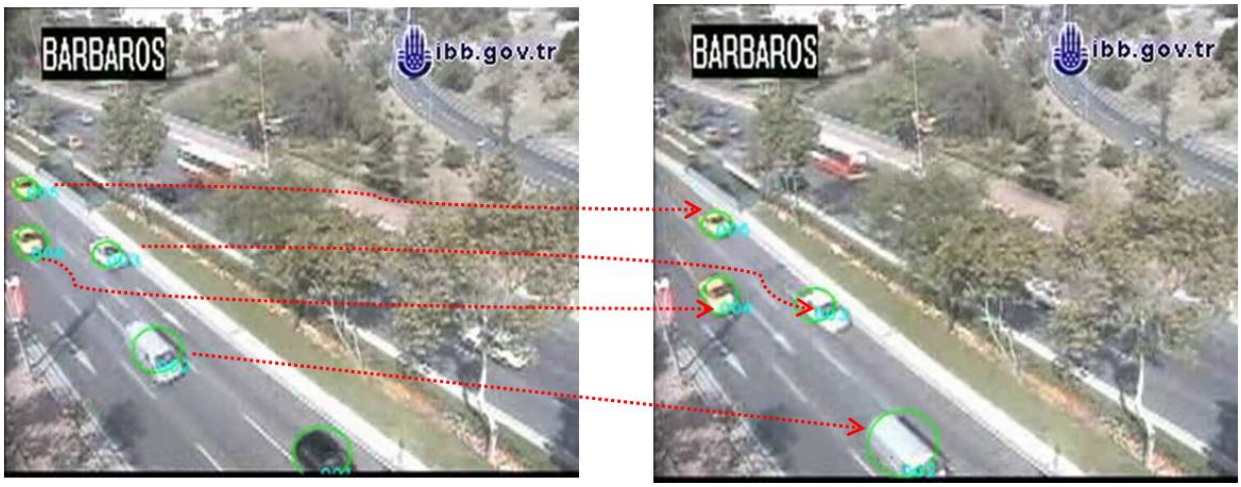


Figure 5 An example tracking result.

3.5. Trajectory Post-processing

Generally, the trajectories obtained by the tracking algorithm are not smooth. For instance, if the car is moving along a straight line, the trajectory extracted will possibly be a jittered line. This is caused by the imperfections in the tracking algorithm, the noise in the video, the compression artifacts such as blocking, color casts in the video, among others. Two ideally identical trajectories can then appear statistically very different when computed by the tracking algorithms. This can have a negative effect on the trajectory analysis module, and therefore we have utilized a post-processing module.

For post-processing we have utilized a Kalman filter to smooth the extracted trajectories. Kalman filters are very popular and effective tools in signal processing, control, and computer vision. The theory of Kalman filters is very rich, and it is generalized to be used to solve many theoretical and practical problems, including estimation, filtering, prediction, etc. In our work, we use the basic form of Kalman filters for filtering purposes.

The basic idea of a Kalman filter applied to tracking is to acquire features of the object as it is tracked throughout the video, create a model using these features, and then filter the tracking results using this model. We have utilized in our work the position and size as blob features. The filtering process can be summarized as follows: The location and size of an object is called the *state* of the object. When an object first enters the scene and the tracker determines its position and size, an independent Kalman filter is assigned to this object. This Kalman filter sets up the necessary state transition information, and

maintains this model by updating it at each frame. At each new frame, after the tracker provides the new position and size of the target, this information is checked with the state provided by the Kalman filter, and is corrected. In other words, the state provided by the tracker and the previous states of the object are compared, and corrected to make the current state consistent with the general movement of the object.

The Kalman filter is very effective in removing tracking errors since it builds a model based on the overall movement and size of the object. Tracking results that cannot be explained with this model are corrected using the filter. On the other hand, Kalman filters account for abnormalities that cannot be explained with the model. Tracking results that deviate too much from the model are not affected significantly by this correction, and the model is updated adaptively using this new information.

For instance, if a car moving on a straight path is expected to keep this path throughout the video, if the tracking indicates rapid small movements orthogonal to this path, it is very likely that they occurred as a result of noise. The Kalman filter can easily detect these errors and remove them. On the other hand, if the car changes lanes, this will be detected as a bigger and non-random movement, and the correction by the Kalman filter will not affect the detected position significantly. Moreover, this movement will be incorporated into the model and will be used in the subsequent video frames.

In addition to the position, the size of an object is expected to stay constant or change smoothly depending on the object movement and camera angle. Any rapid changes on the detected object size will be removed by the Kalman filter. On the other hand, if the size of the object is changing uniformly (for instance, if the car is moving towards the camera) this will be detected by the Kalman filter and will be incorporated into the model.

3.6. Tracking Results

In this section we present some experimental results to demonstrate the performance of the tracking module. We performed extensive experiments with real traffic footage to verify and test the performance of the method. Our current video database consists of videos collected from the traffic monitoring websites of the City of Istanbul (http://tkm.ibb.gov.tr/en-EN/yolDurumu/Kameralar.aspx) and Caltrans (http://www.video.dot.ca.gov/) and the 2001-2004 databases of the International Workshop on Performance Evaluation of Tracking and Surveillance (PETS).

Finally, we have found a new large database of aerial traffic videos from NGSIM (http://ngsim.camsys.com/). Since these videos are generated by cameras located on top of the highways, the vehicles recorded have small sizes and from the top, which required some modifications to our tracking software. On the other hand, NGSIM also provides ground truth tracking results, which can immediately be used in our trajectory analysis module.

Figure 6 Extracted trajectories

4. Phase 2: Trajectory Analysis

4.1. Method Overview

In our cases, as no prior knowledge is given for patterns of unusual video events, we aim to analyze all the trajectories extracted from existing videos, and differentiate unusual trajectories from normal ones automatically. To address this problem, a clustering-based approach has been investigated. This approach is based on the fact that a normal event is associated with the commonality of the behavior and an unusual event indicates its distinctness. For instance, people running represent an unusual event if most people in the crowd are walking, and a car moving against traffic also represents an unusual event. Clearly, what characterizes normality is the high recurrence of some similar events. Typically, there are only a few such normal patterns in a specific surveillance scenario. Therefore, unsupervised clustering can be performed on all video events, so that those events clustered into dominant (e.g., large) groups can be identified as normal, while those that cannot be explained by the dominant groups (e.g., distant from all cluster centers) are defined as unusual.

We propose a novel dynamic hierarchical clustering (DHC) method for unusual event detection from surveillance video. The proposed approach follows the common steps of the clustering-based approach and includes 2 steps:

1. Trajectory clustering
2. Unusual trajectory identification

In the following sections we will provide details on each of the steps.

4.2. Trajectory Clustering

In order to perform clustering on video events, we need to measure the similarity of two events. Usually, a trajectory can be modeled as a dynamic process with different outputs at different times. We can measure the distance between trajectories based on the similarity between their corresponding models. In our system, we simply use the 2-D coordinates of the object center at every frame as trajectory features. Hidden Markov models (HMM) with Gaussian emission probability are used to characterize the trajectories, and agglomerative hierarchical clustering can be performed using a BIC-based dissimilarity measure. Specifically, the distance between two trajectories or trajectory groups *i,j* is defined as

$$d(i,j) = \log L_i + \log L_j - \log L_{ij} - \frac{1}{2} K_0 \log N,$$

where L_i and L_j denote the likelihoods of *i* and *j* being generated by their own models and L_{ij} is the likelihood of *i,j* being generated by a model trained on themselves, *N* is the number of all trajectories, and *K* is the number of model parameters. In our algorithm, the two trajectories or trajectory groups *i,j* with smallest $d(i,j)$ as defined are continuously merged until there is no $d(i,j)<0$.

However, a model-based similarity measure as mentioned above has the overfitting problem given few training samples. This is true for the first several steps of hierarchical clustering: when clusters contain only a few trajectories (starting from only one trajectory), the trained HMMs tend to be overfitted and the dissimilarity measures based on them are quite unreliable, thus resulting in clustering errors. These errors will propagate to future clustering steps. To address this problem, we can update the clustering results at each merging step. In other words, once a new HMM is trained after trajectory merging, all the trajectories in the database are reclassified. Possibly, some incorrectly clustered trajectories at previous steps are associated with the new HMM. All the HMMs are then retrained based on the updated trajectory clusters. This is a typical data reclassification and model retraining process, which is used in many iterative algorithms such as the Expectation-Maximization algorithm. This updating also enables error correction at later clustering steps, as clusters have gathered more samples and the trained HMMs are more reliable. We refer to this process as dynamic hierarchical clustering (DHC), which is illustrated in Figure 7. It differs from typical agglomerative hierarchical clustering in that it incorporates into each clustering step the data reclassification process (step 3) and the model retraining process (step 4).

0) *Initialization:* each trajectory in the dataset forms a group and is fitted with an HMM. There are N groups with N HMMs:
1) *Dissimilarity Measurement:* calculate the dissimilarity $d(i, j)$ for every two groups i and j in the dataset by the measure in (5);
2) *Merging:* the two groups i and j with smallest $d(i, j)(d(i, j) < 0)$ are merged; if there is no $d(i, j) < 0$, the clustering terminates;
3) *Reclassification:* a new HMM $\theta_{i,j}$ is trained, replacing θ_i and θ_j; then based on the $(N - 1)$ HMMs, all trajectories are reclassified into $(N - 1)$ groups by the maximum likelihood (ML) criterion;
4) *Retraining:* $(N - 1)$ HMMs are retrained based on the updated $(N - 1)$ data groups, respectively;
5) *Update:* $N = N - 1$; go back to step 1).

Figure 7 Dynamic hierarchical clustering (DHC)

4.3. Unusual Trajectory Identification

Suppose that all trajectories in the training dataset are clustered into *C* groups with *C* corresponding HMMs. Out of these groups, normal (dominant) trajectory groups need to be identified, while those that cannot be explained by the dominant groups are detected as unusual. In our approach, a probabilistic framework is used to identify dominant groups from the clustering results.

In detail, each trajectory *i* has probability $L_i^{\,c}$ of being generated by model M_c. Hence, we may consider each trajectory as being generated by a mixture model. In this mixture model, each component is one of the *C* HMMs. Therefore, instead of checking for large size groups, normal clusters can be determined as those with high prior probabilities (higher than average) in the mixture model, while other clusters with low prior probabilities are detected as unusual trajectories.

4.4. Results

The proposed method was tested with a real traffic surveillance scene. The videos are from a new large database of aerial traffic videos from NGSIM (http://ngsim.camsys.com/) and they are generated by cameras located high above highways, monitoring all moving vehicles over long periods of time. All trajectories of the vehicles are provided by this database. We have used 1000 trajectories extracted from the video scene shown in Figure 8. In this section of the highway, there are 6 traffic lanes and 1 merging lane. We have plotted all trajectories in one graph (Figure 9). All trajectories begin on the left of the graph, with their starting points denoted by yellow circles. They move to the right, with every step denoted by blue dots. As can be seen, most vehicles are moving along their own lanes or change lanes

infrequently (which need to be identified as normal events), while only a few of them change lanes frequently (which need to be detected as unusual events).

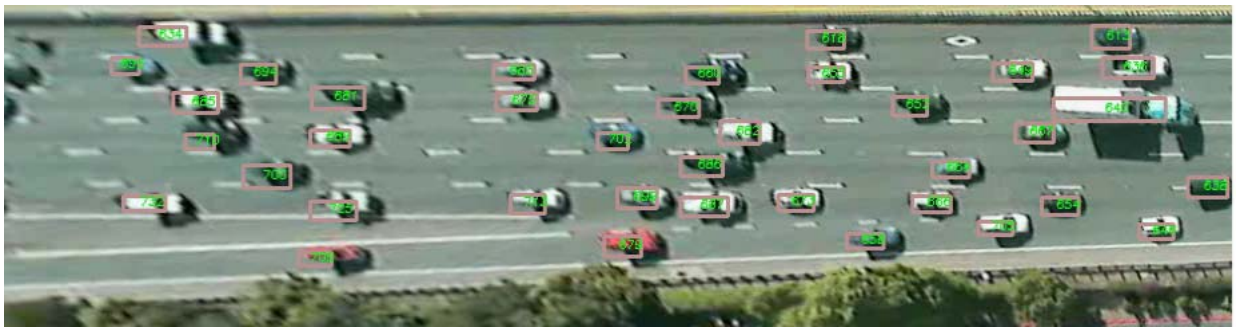


Figure 8 Example of traffic video being analyzed (red squares are tracking results of vehicles)

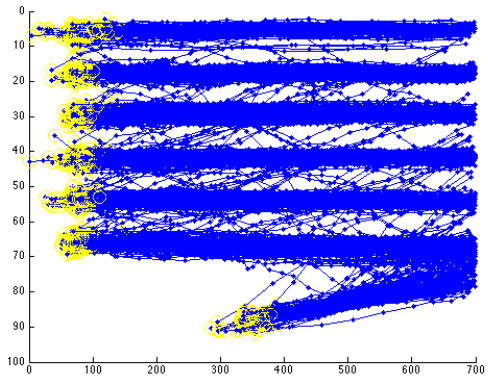
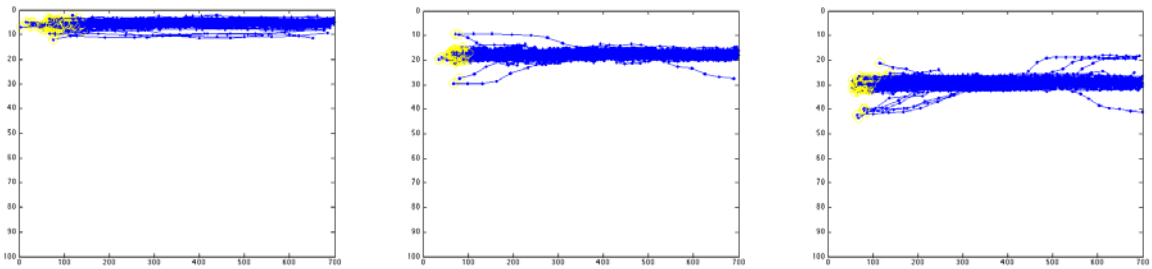


Figure 9 All trajectories extracted from the video scene shown in Figure 8

Our algorithm works on this dataset and automatically provides the results of normal trajectory clusters and unusual trajectories. In detail, it identifies 7 clusters of normal events, corresponding to the 7 lanes, as shown in Figure 10.



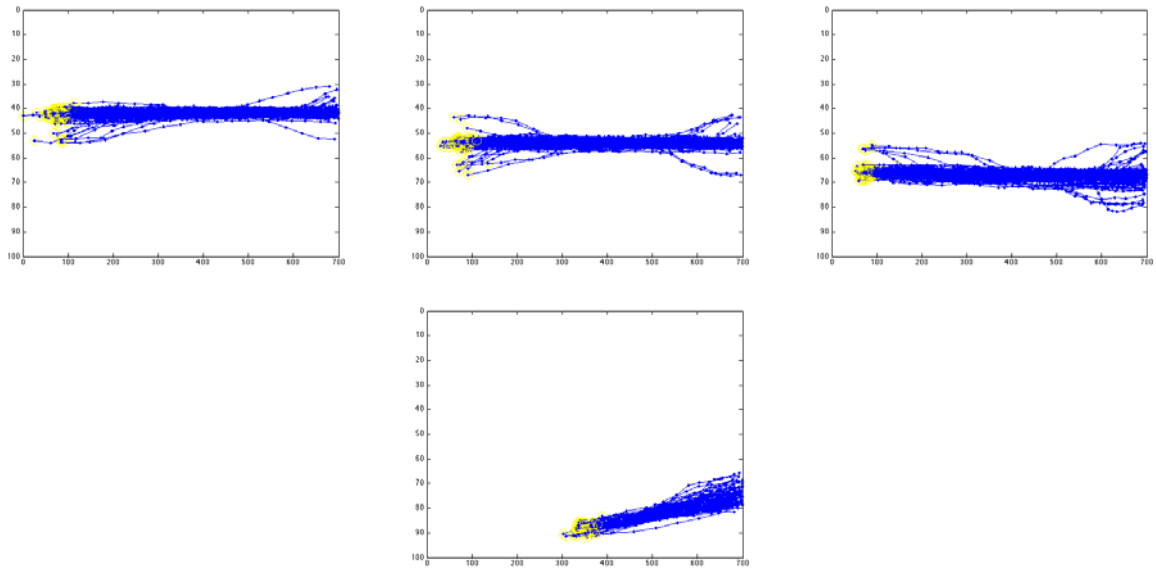


Figure 10 Seven clusters of normal trajectories

The unusual events, which represent lane changers, are shown in Figure 11.

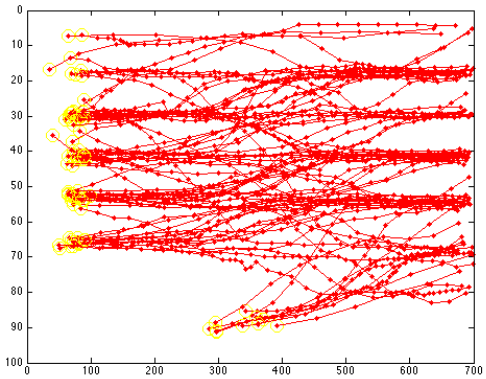


Figure 11 Unusual trajectories

To better demonstrate this, we have shown some examples of unusual trajectories in Figure 12.

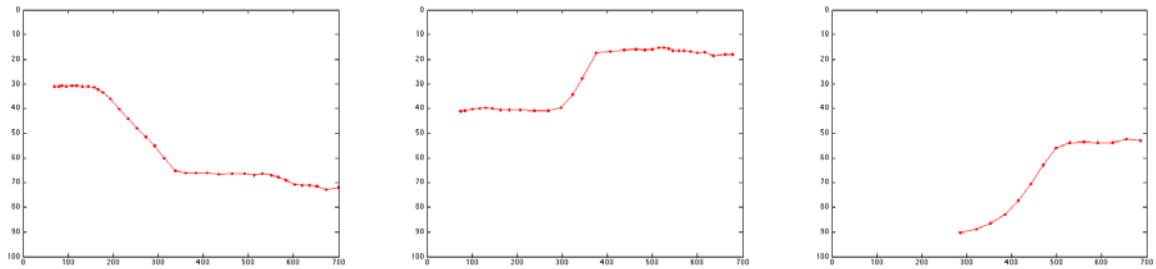


Figure 12 Examples of unusual trajectories

The experiments show that our approach works well in analyzing traffic trajectories in some simple video scenes. It can automatically detect trajectories with unusual routes, which appear rarely compared to other normal trajectories. The advantage of our video mining approach is that it does not require prior knowledge about the nature of normal or unusual events.

5. Anomaly Detection in Spatial and Temporal Context

5.1. Contextual anomaly

In our previous research, we have proposed methods for detecting anomalous behavior of a single object during its whole appearance time in the traffic video. For example, we consider the trajectory of one vehicle (from the time it appears to the time it disappears), and detect the anomalous trajectory that is different from most other trajectories. Detection of these anomalous trajectories is useful because these anomalies usually correspond to traffic rule violations and are direct causes of traffic disruption (e.g., traffic slowing, traffic congestion).

However, there are more disruptive events in traffic video that are not included in our previous framework. For example, disruption can arise due to the interaction of multiple spatially related objects. Another example is the disruptive behavior of one single object during part of its appearance (instead of during the whole time it appears on the scene). Actually, these disruptive events include more contextual information, e.g., spatially related objects and temporally related behaviors. Therefore, we need to extend the concept of anomaly to include video context information (both spatial and temporal).

We consider two different spatial contexts of a video event: single object and multiple spatially related objects. For the temporal context, we consider it to be either a short time event (taking place at one time instance) or long time event (including several short time events and their transitions). According to different spatial and/or temporal context, we can categorize video anomaly into 4 types, as shown in Table 1. Point anomaly (similar definition in [10]) is the simplest case of an anomalous single object behavior at one time instance, and is mostly studied in the literature. If the anomaly rises only within a sequence of short time events (including their ordering and transitions), it is defined as sequential anomaly. For multiple objects, if just the occurrence of them at a certain time instance results in anomaly, we define it as co-occurrence anomaly. If the anomaly arises only within a sequence of co-occurrence events, i.e., multiple spatially related objects behavior for a long time, it is defined as interaction anomaly.

Spatial context / Temporal context	Single object	Multiple objects
Short time	Point anomaly	Co-occurrence anomaly
Long time	Sequential anomaly	Interaction anomaly

Table 1. Anomaly categorization by spatial and temporal context

Under this definition, our previous work of detecting anomalous trajectory falls in the type of sequential anomaly detection. Note that sequential anomaly is a more broad definition because it does not necessarily correspond to the whole time of one object's appearance. It can be an arbitrary length that is longer than a time instance. Except for the point anomaly, the other 3 types all consider spatial/temporal context information in video, so that they can be called contextual anomaly.

In the following we present two case studies with different video scenarios. Both of them illustrate the importance and effectiveness of contextual anomaly detection in video analysis. The first case deals with anomalous motion of multiple spatial related objects in a crowd (co-occurrence anomaly). Although the experiments have been done based on pedestrian crowd video, it shows the ability in dealing with similar motion scenarios in the traffic video. The second case study is based on a real traffic video of a road intersection. Up to now, we have achieved some preliminary results that show the ability of our method to detect point anomaly, sequence anomaly and co-occurrence anomaly in traffic video.

5.2. Case study 1: detecting anomaly in spatial context of crowd motion [11]

5.2.1. Introduction

In many surveillance systems in public places such as city streets, subway stations, or malls, video is recorded depicting the movement of crowds. It would be very useful to locate and recognize hazardous and anomalous human motions from the video to alert system operators. By anomaly detection, we mean to detect motion patterns in the video that do not conform to the expected behavior.

The simplest type of anomaly, which is also the focus of the majority of the research [12-16], is to detect an individual behavior instance that is considered to be anomalous with respect to the rest of behaviors. It just considers a single object and falls into the two categories of anomalies mentioned above: point anomaly and sequential anomaly.

However, sometimes the individual behavior itself has similar features with others but it is anomalous in a specific context (e.g., when multi-objects are considered in a spatial neighborhood). This contextual anomaly is not widely studied for crowd analysis in previous research. One recent work [17] proposed an unsupervised framework adopting hierarchical Bayesian models to model activities and interactions in crowded scenes. Anomaly detection could be done for atomic activities (corresponding to point anomaly) and interactions (actually corresponding to co-occurrence anomaly in our categorization).

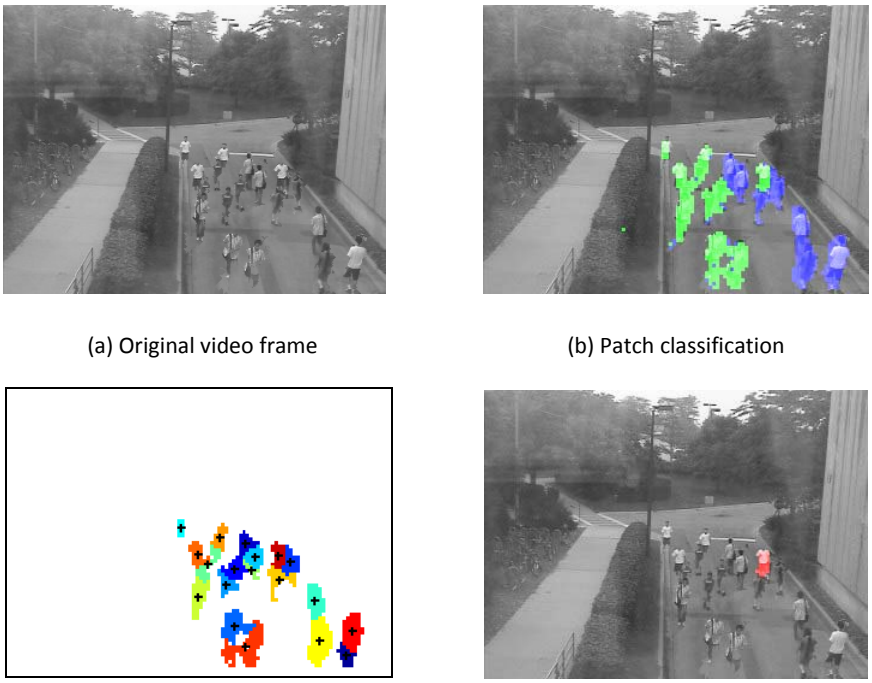
In this case study, we propose an unsupervised approach for detecting contextual anomalies in crowd motion. First, motion features in the crowd scene are represented by spatio-temporal patches, which are characterized by dynamic texture. All patches are then classified and grouped into blobs that approximately describe position and size of every pedestrian. Then based on the spatial layout of pedestrians with different motions, our system automatically discovers important contextual information and detects the blobs corresponding to contextually anomalous behaviors. Our main contribution is introducing the concept of contextual anomaly into the field of crowd analysis, and proposing an approach to automatically detect those contextual anomalies.

5.2.2. Motion representation and classification

Due to the density of objects in a crowd scene, accurately tracking individual objects is difficult. We characterize the crowd motion by the patch-based local motion representation. Similarly to [16], the non-stationary parts in the video are represented as a collection of spatiotemporal patches of dimension p*p*q, where p (spatial size) and q (temporal size) should be large enough to capture the distinguishing characteristics of the various components of the local motion field. Every patch is characterized by a dynamic texture model [18], which is actually a linear dynamic system. Given the appearance vectors, this model can be parameterized by a suboptimal (but tractable) approach in [18].

Based on this representation, we can cluster all patches into different motion categories (behaviors). In our work, we adopt the spectral clustering algorithm. The pairwise distance between two patches is defined by the Martin distance [19], which is based on the principal angles between the subspaces of the extended observability matrices of the two dynamic textures.

One example is shown in Figure 13. Figure 13a shows one frame of the crowd video, where many people walk in two directions on the road. Figure 13b shows the patch representation and clustering results for this frame. All the non-stationary parts in this video are represented by patches (with dimension 10*10*20) and all the patches are clustered into two categories (colored in green and blue).



(a) Original video frame

(b) Patch classification

(c) Blob representation

(d) Contextual anomaly

Figure 13. Example of contextual anomaly detection

5.2.3. Context representation and anomaly detection

At this patch representation level, there is no anomalous motion in the video because every patch falls into one of two normal clusters. However, there exist some spatial contextual anomalies. When we consider the context all through the whole video, we find that in this video most of the time people follow the crowd flow, i.e., they walk in the same direction as their neighbors. Only in very few cases people disobey this rule by walking in the opposite direction of the flow. One example is the green blob at the upper-right corner of Figure 13b surrounded by blue patches, where one pedestrian is walking downwards while pedestrians in his neighborhood all walk upwards. We aim to automatically detect this kind of contextual anomaly.

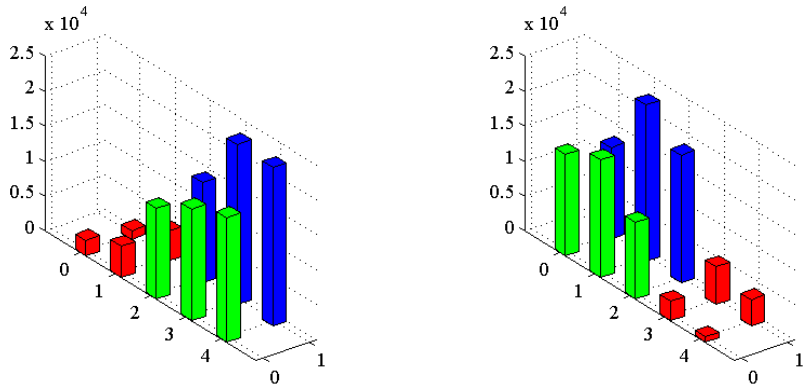
In our work, we consider the motion context of pedestrians. Ideally, we can segment every pedestrian and find its consisting patches, then use the class label (green or blue) at each pedestrian's neighborhood as the contextual attributes. However, accurate segmentation of people from crowd scenes is a difficult problem, and our goal is not pedestrian boundary detection but motion context representation. Hence, a blob representation that coarsely corresponds to pedestrians, as long as the contextual information is not changed, can serve our purposes. We manually find two sizes (in number of patches) of an average person at the nearest and the farthest end, respectively, in the video scene. Then we approximate the size of any pedestrian at any place by linearly interpolating between the two size extremes. Aided by the estimated pedestrian size, we perform region growing on the patches of different categories. Figure 1c shows all blobs (both green and blue) in pseudo-color, with the black crosses denoting their centers. Although each blob does not exactly fit the boundary of every pedestrian, it captures the correct category label (green or blue) of each pedestrian and its neighbors. Thus this blob representation has the contextual information of pedestrians and ensures meaningful contextual analysis.

In detail, the motion context for each blob can be defined based on its K nearest neighbors. In the example of Figure 13, the contextual information of each blob is coded by 1) the category label of itself (0 represents green and 1 represents blue) and 2) the number of neighboring blobs with the same label as itself (from 0 to K). For instance, when we consider the 4 nearest neighbors for each blob, a green blob with 3 green neighbors and 1 blue neighbors is coded as (0, 3). This code represents the blob motion information as well as the motion contextual information.

Once the contextual information is coded for each blob, contextual anomaly can be detected based on its statistics. For a crowd video, we process each frame and gather the contextual codes for all the blobs. The statistics of all the blob codes are represented in a 2-D histogram, where the x and y axis denote the two codes respectively and the z axis is the count of each code. The histogram naturally shows the motion context of the crowd video: high bins are repetitive (thus normal) motion contexts, while low bins are rare (thus anomalous) contexts.

The histogram in Figure 14a shows the statistic of context for the video example in Figure 13. As most pedestrians in the video walk in the same direction as their neighbors, we have high bins for green (blue) blobs with 2 or more green (blue) neighbors and have low bins for green (blue) blobs with 1 or

fewer green (blue) neighbors. By simply imposing a threshold, which can be the average height of all bins, we can discover the low bins that correspond to contextual anomalies (shown as red bins in Figure 14a. In this video, all the blobs that are coded as (0,0), (0,1), (1,0), or (1,1), which correspond to the rare cases of pedestrians walking in the opposite direction of their neighbors, are detected as contextual anomalies and labeled with red in the image as in Figure 13d.



(a) For the first video (b) For the second video

Figure 14. Contextual histogram

5.2.4. Results and conclusion

We have experimented with two crowd video sequences with different motion context. The first video is composed of 6000 frames, where crowds of people walking as two flows (downwards and upwards) on the two sides of one road. The example shown in Figure 13 is one frame drawn from this video. More results are shown in Figure 15, where the top row shows the patch representation of 4 frames selected from the video (two motion categories labeled as green and blue), and the bottom row shows the results of contextual anomaly detection (red blobs). The decision is made based on the contextual histogram shown in Figure 14a. The high bins (green and blue bins) denote the normal motion context, i.e., pedestrians (of both flows) walking in the same direction as most of its neighbors. The low bins (red bins) correspond to anomalous motion context, i.e., pedestrians (of both flows) walking in the opposite direction as most of its neighbors.

The second video has similar length and includes many people walking on the same road. However, instead of two clear motion flows shown in the first video, the second video has pedestrians walking in opposite directions intermingled with one another, i.e., random distributed positions. A normal scene is shown in the first image of Figure 16. This is a totally different scenario than in the first video. As expected, the contextual histogram of the second video (shown in Figure 14b) has high bins corresponding to pedestrians walking in the opposite direction as most of their neighbors, and low bins corresponding to pedestrians walking in the same direction as most of their neighbors. Therefore, the contextual anomaly detected in the second video is no longer those opposite walkers, but those co-walkers. As shown in the results in Figure 16, a group of people walking upwards together are detected as anomalies. This experiment shows the advantage of our unsupervised anomaly detection approach: it adaptively analyzes contextual information for different crowd videos, and anomaly detection results

are always given based on the statistics of the specific crowd scenario, with no a priori knowledge required.

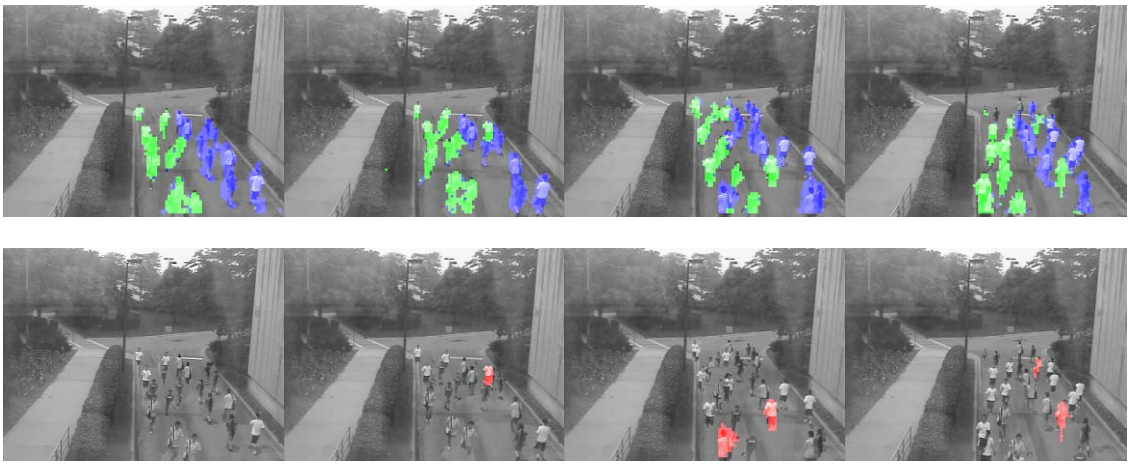


Figure 15. Anomaly results for the first video

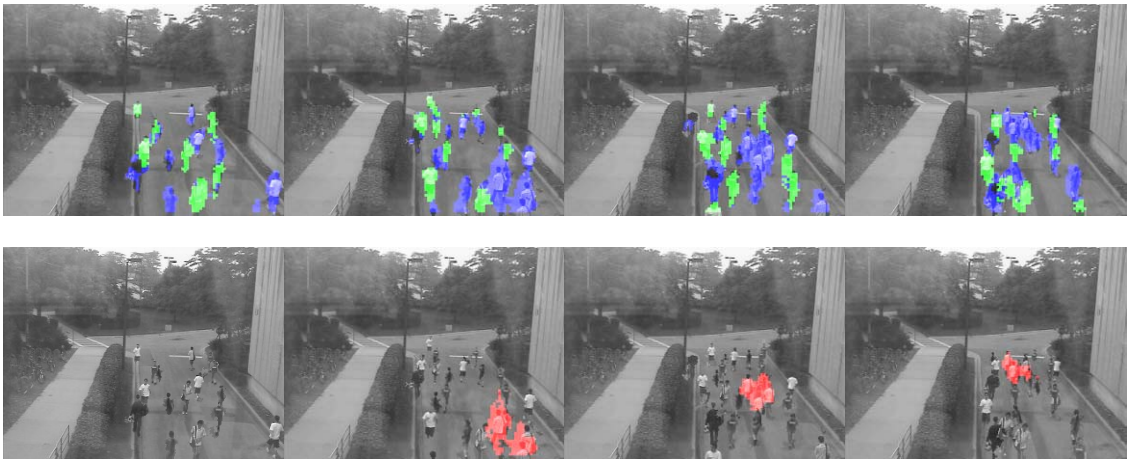


Figure 16. Anomaly results for the second video

In this case study, we introduce a new concept of contextual anomaly into the field of crowd analysis. Our system focuses on motion context of moving objects and can automatically discover anomalous motions in terms of context (neighborhood motion). This is based on statistical analysis of the contextual information for any given video, thus no prior knowledge is required. Experimental results have been provided for crowd videos with different motion contexts.

5.3. Case study 2: detecting anomalies in spatial and temporal context of traffic at road intersection

5.3.1. Introduction

In many real traffic scenarios, especially some complicated road situations, there are different types of anomalies. Therefore, video anomaly detection can be performed at different levels. In this case study,

we have selected a surveillance video monitoring traffic for a long time at a road intersection. The anomalous events should be detected from all behaviors of vehicles traveling through this area, including both point anomaly and contextual anomaly. Ideally, the detected anomalies correspond to traffic rule violators and behaviors causing traffic disruption.

This video is taken from a large database of aerial traffic videos from Next Generation Simulation (NGSIM) project (http://ngsim.camsys.com/). One example frame is shown in Figure 17. This is a half-hour-long video monitoring a 4-way intersection of Lankershim Blvd. (north-south), Campo De Cahuenga Wy. (west), and Universal Hollywood Dr. (east) in Los Angeles, California. Each of the roads is a two-way road with multi-lanes (some with right turn or left turn lanes). All moving traffic in this area is guided by traffic lights within the intersection.

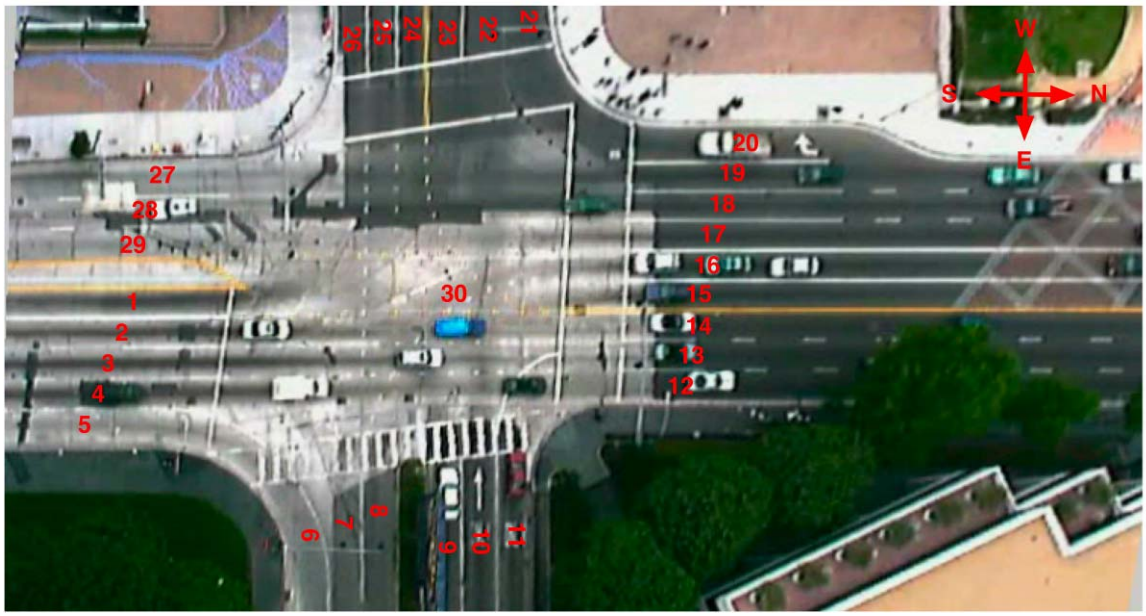


Figure 17. Example frame of video monitoring traffic at road intersection

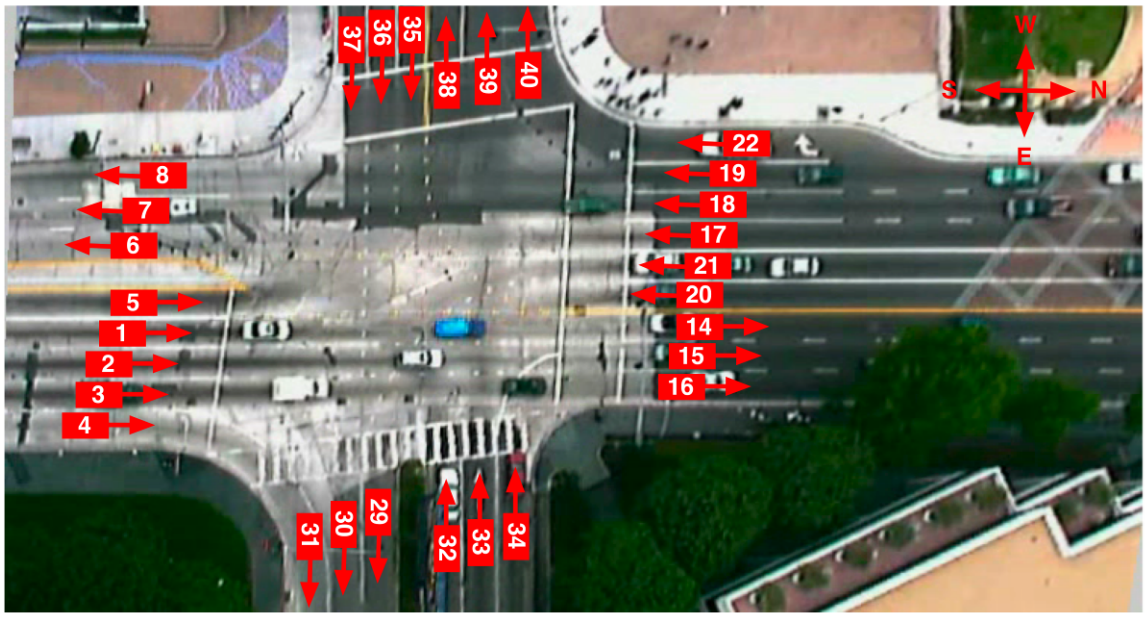
Similarly to case study 1, we assume that no prior knowledge is available about the traveling routines in this area. In other words, no training samples are available for either normal events or anomalous events. We need to take an unsupervised approach to automatically detect anomalies from video data itself. With no prior knowledge of this specific traffic scenario, our approach is data-driven and can be extended to other traffic scenarios.

5.3.2. Point anomaly detection

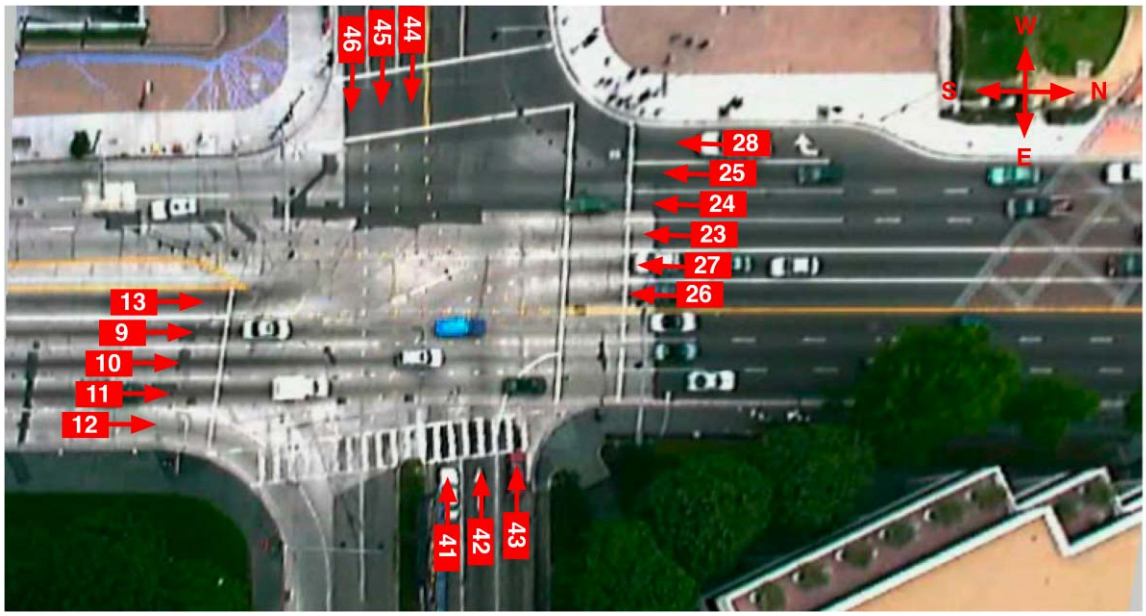
Similar to case study 1, an anomalous event in the traffic scenario is associated with its rareness. When considering traffic in a long time window, most vehicles behave normally so that traffic conditions generally remains good. Only a few vehicles behave anomalously for some time (e.g., violating traffic rule), which are important causes of occasional traffic congestion. Based on this fact, when considering the behavior of each single vehicle at each time instance, we calculate the frequency of all instant behaviors and detect those behaviors with small support as point anomalies.

In the traffic scenario, the instant behavior of one vehicle can be described by its position and its motion. In our case, 4 features are used: the section and lane number of the vehicle position, the driving direction of the vehicle, the turning direction of the vehicle, and the vehicle speed. To make it simple, every feature is quantized to discrete values. As shown in Figure 17, all lanes are numbered from 1 to 29. The intersection area is numbered as 30. The driving direction has 4 possible values (north, south, west, east). The turning direction has 3 possible values (straight, left turn, right turn). The speed feature is discretized to either moving (speed >> 0) or stopping (speed ≈ 0).

For the vehicles traveling on lanes, the turning direction is almost always "straight", thus the other 3 features are used. We make a 3-D histogram for all the instant behaviors on lanes through the video. By simply applying a threshold, we detect 46 frequent (normal) behaviors, as shown and numbered in Figure 18. Specifically, Figure 18a and Figure 18b show the moving normal behaviors (speed >> 0) and stopping normal behaviors (speed ≈ 0), respectively. The red squares show the vehicle positions. The arrows show the moving directions.



(a) Moving behaviors



(b) Stopping behaviors

Figure 18. Normal instant behaviors for vehicles on lanes

For the vehicles traveling within the intersection, the section and lane number of the vehicle position cannot be used, thus the other 3 features are used. Similarly, we make a 3-D histogram for all the instant behaviors within the intersection. By simply applying a threshold, we detect 20 frequent (normal) behaviors, as shown and numbered in Table 2. Note that for moving vehicles, driving in all directions while making turns or going straight can be normal. But for stopping vehicles, their behavior is normal only if it is making turns, because one rarely stops while going straight.

	Eastward	Northward	Westward	Southward
Straight	47	48	49	50
Left turn	51	52	53	54
Right turn	55	56	57	58

(a) Moving behaviors

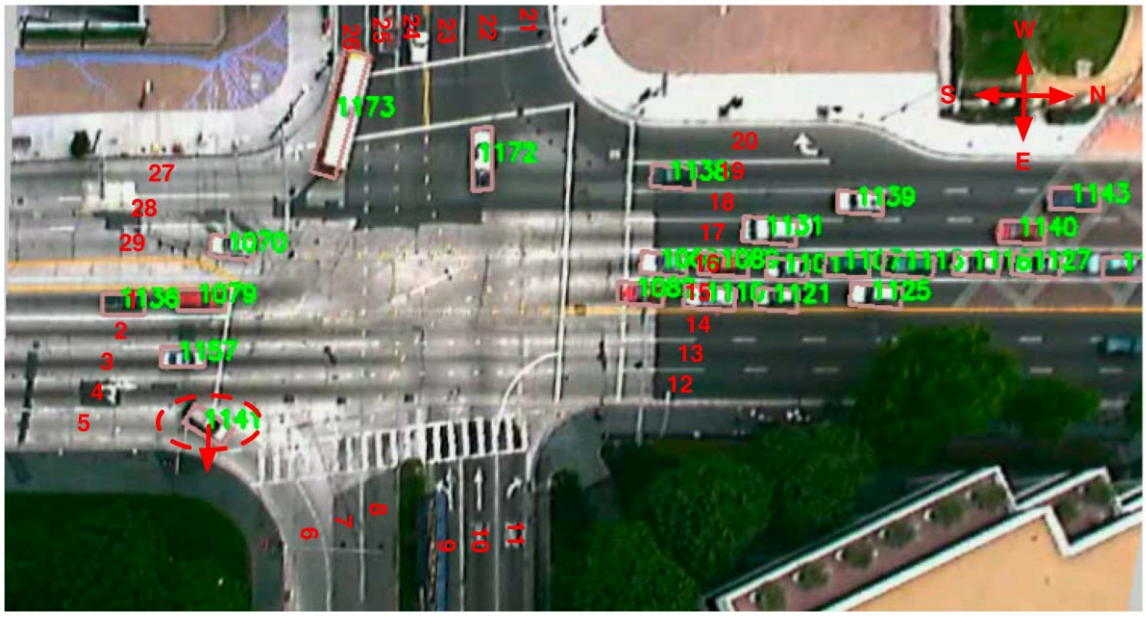
	Eastward	Northward	Westward	Southward
Left turn	59	60	61	62
Right turn	63	64	65	66

(b) Stopping behaviors

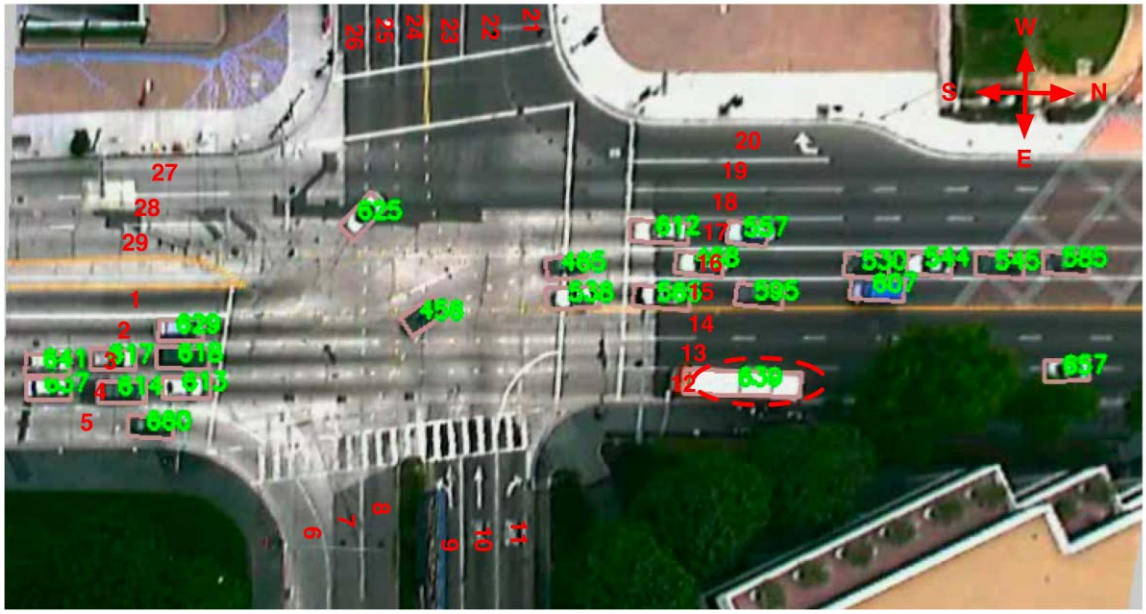
Table 2. Normal instant behaviors for vehicles within intersection

In total we have 66 (46 shown in Figure 18 a and b plus the 20 behaviors shown in Table 2) normal instant behaviors. It is observed that these 66 normal instant behaviors include all legal driving conditions allowed in every lane. Consequently, the instant behaviors that do not fall in any of these normal ones correspond to illegal driving behavior and thus they are detected as anomalies. Two examples are shown in Figure 19. Figure 19a shows a vehicle moving eastward in lane 5 but making a late lane change so as to make a right turn. This abrupt lane change is so close to the intersection that it

is a disruptive event. Figure 19b shows another disruptive behavior, i.e., one vehicle stopping in lane 12 and thus blocking traffic.



(a) Moving eastward at lane 5



(b) Stopping at lane 12

Figure 19. Examples of point anomalies

5.3.3. Sequential anomaly detection

Point anomalies do not include all anomalous events as they neglect temporal information. There exist cases when all instant behaviors of one vehicle are normal but the ordering or transitions of these behaviors are anomalous. As an example, consider vehicles making a right or left turn illegally, i.e., from the wrong lane. Similarly to point anomaly detection, sequential anomaly can also be determined based on its low frequency of occurrence. However, as a sequential anomaly may last for an arbitrary length of time, techniques are needed to deal with this time variation.

Based on the 66 normal instant behaviors we obtained, all point anomalies can be excluded from the database. Sequential anomaly can then be detected from the rest of the data. For each vehicle, from the time it appears in the video to the time it disappears, a sequence is formed as the concatenation of all instant behaviors. The sequence of behaviors contains information about the temporal relationship of instant behaviors. For example, most of the vehicles starting with behavior 1 (in Figure 18a) are going to proceed with behavior 48 followed by behavior 14, because lane 2 (in Figure 17) is only for vehicles going straight to lane 14. The sequence 1-48-14 should appear many times (possibly as subsequence) within all the sequences collected from the video. Note that its appearances with some insertions should also count (e.g., 1-9-48-14 is possible because vehicle may wait for a traffic light before it enters the intersection). On the other hand, sequence 2-48-14 appears rarely if at all, because few vehicles change lane within the intersection (which is actually illegal).

In order to automatically detect sequential anomaly, we can first discover all normal (sub)sequences. We adopt the technique of frequent subsequence mining on all the sequences collected from the video. The subsequence relationship is defined between two sequences s1 and s2 as follows: s1 is a subsequence of s2, if and only if a monotonically-increasing index mapping I() for each element in s1 can be established, such that each element s1(k) is a subset of s2(I(k)). That is, a sequence is a subsequence of another, if it can be matched with arbitrary long gaps but preserving the order, such that the matched elements satisfy a subset relationship. We use the PrefixSpan algorithm [20] and automatically find all frequent subsequences with their lenght above a given threshold. The frequent subsequences are listed in Table 3. From the illustration in Figure 20, we observe that all possible traveling routes permitted in this area are included.

1-48-14	2-48-15	3-48-16	17-50-6	18-50-7
19-50-8	32-49-38	33-49-39	36-47-29	37-47-30
4-56-55-31	34-57-56-15	34-57-56-16	19-22-58-37-40	37-55-58-8
5-52-53-38	5-52-53-39	20-54-51-29	21-54-51-30	32-53-54-6
35-51-52-14	35-51-52-15	35-51-52-16		

Table 3. Frequent subsequences of behaviors

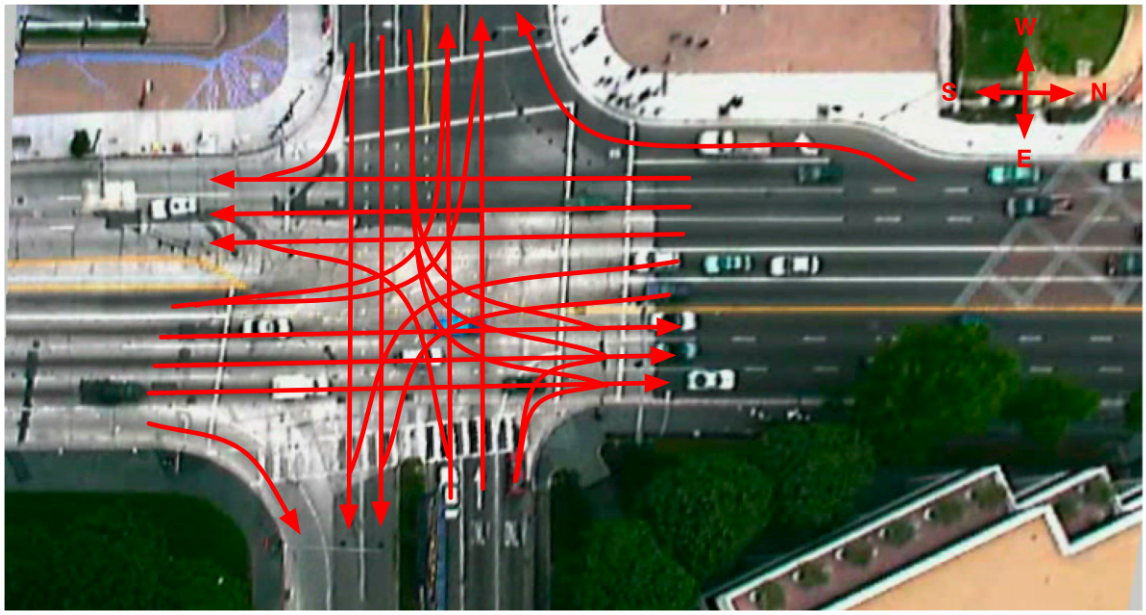
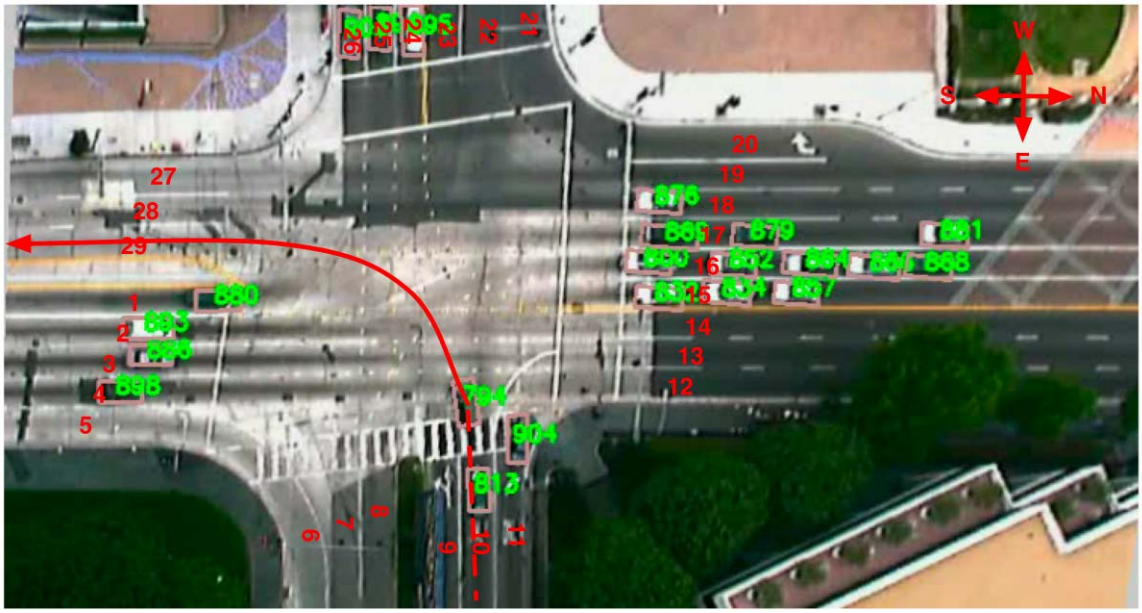


Figure 20. Illustration of frequent subsequences of behaviors

The frequent subsequences are actually normal behavior patterns. Sequential anomalies can be detected by simply removing frequent subsequences from all the sequences collected. Those remaining infrequent subsequences are detected as sequential anomalies. Two examples are shown in Figure 21, with the anomalous part shown in red dashed line. Figure 21a shows a vehicle changing lanes within an intersection. Figure 21b shows a vehicle making a left turn from a straight only lane. Both of them are illegal behaviors. Note that the sequential anomaly is not necessarily a whole sequence. In fact our method is able to detect a sequential anomaly from any part of a sequence with arbitrary time length.

(a) Changing lane within intersection



(b) Making left turn from wrong lane

Figure 21. Example of sequential anomalies

5.3.4. Co-occurrence anomaly detection

The co-occurrence of multiple vehicles at a certain time instance also results in an anomaly. To be more specific, in this video scenario, as a result of the traffic signals, all traffic movement as a whole has different states. There are 5 states in total: 1) North-south traffic turning left; 2) Southward traffic going straight and turning left; 3) North-south traffic going straight; 4) Westward traffic going straight and turning left; 5) Eastward traffic going straight and turning left. These states represent 5 basic patterns of co-occurrence of multiple vehicles. In most cases, certain vehicle movements occur at a certain traffic state. Those vehicle behaviors occurring at a wrong state often means either blocking of traffic or collision of multiple vehicles. For example, two conflicting behaviors occur simultaneously: straight north movement and left turn east. These conflicting spatial movements should be detected as co-occurrence anomaly.

In order to automatically detect co-occurrence anomaly, we can first discover all normal co-occurrence patterns (related to traffic states). We build up a set of documents by collecting the instant behaviors of all vehicles at each time instant into a document. Spatial context information is included in these documents. In fact, there are some groups of behaviors occurring together a number of times, which correspond to the normal co-occurrence patterns. We adopt the technique of frequent itemset mining (LCM by [21]) to automatically discover these groups. Co-occurrence anomaly can be detected by simply removing frequent itemsets from all the documents collected from the video. Those infrequent items remained are detected as co-occurrence anomalies.

5.4. Conclusion and future work

Discovering suspicious or anomalous events from video streams is an important yet challenging problem for many video surveillance applications. By automatically finding suspicious events, it significantly reduces the cost to label and annotate the video streams of hundreds of thousands of hours. We propose in this report a complete system to detect and classify anomalous events from surveillance video.

First, as the video camera is fixed and the site being monitored is mainly static, by modeling the statistics of the background and the appearance and dynamics of the foreground (objects such as a person, car, airplane), various features of the objects, such as location and motion at different times, can be extracted from the video data. These object features are used to characterize video events.

Second, multi-step analyses are performed on all acquired trajectories (representation of video events), in order to detect and classify different kinds of anomalous events. With no prior knowledge about anomalous behaviors in a specific video scenario, it is difficult to define anomaly in an explicit manner. It is possible that we may need to identify an anomalous event when it appears, despite the fact that it had never occurred before. New algorithms are proposed in this report to deal with this problem following different approaches.

A *clustering-based trajectory analysis approach* is based on the fact that normal events appear frequently and dominate the data, while anomalies are different from the commonality and appear rarely. Therefore, unsupervised clustering can be performed on all video events. Those events clustered into dominant (e.g., large) groups can be identified as normal, representing the regular rules. Those that cannot be explained by the regular rules (e.g., outliers distant from all cluster centers) are detected as anomaly. The new algorithms developed in this part that were published include dynamic hierarchical clustering [8][9], and 2-depth greedy search [7].

A *contextual information mining approach* results in analyzing video events at different levels considering both spatial and temporal context. In detail, considering spatial context, we analyze events of a single object and events of multiple spatially related objects. Considering temporal context, we analyze both short time (instant) events and long time events (including several short time events and their transitions). Accordingly, anomalous video events can be detected at different levels. One application of this algorithm on detection of abnormal crowd motion is presented in [11]. This complete approach has been summarized in our recent paper [22] submitted to the journal of Computer Vision and Image Understanding.

As already mentioned we categorize video anomaly into 4 types, according to different spatiotemporal context considered. We have investigated the detection of point anomaly, sequential anomaly, and co-occurrence anomaly. The detection of interaction anomaly involves multiple objects with complicated temporal logic and will be our future work. Furthermore, we will consider how to incrementally update

the current models as new video observations stream in, so that the model can efficiently adapt to visual contextual changes over a long period of time.

We believe that significant results have been obtained with the conclusion of this project. Such results can have a great impact in the automated analysis of video traffic data in detecting abnormal event. We have provided our results to INEX/Zamir so that they evaluate them and decide on the next steps to be followed. INEX/Zamir has been providing traffic monitoring solutions for many years and this research is closely aligned with their market interest and customer needs.

References

1. L. Li, W. Huang, I. Y. Gu, and Q. Tian, “Foreground object detection from videos containing complex background,” *Proc. of the Eleventh ACM international Conference on Multimedia*, Berkeley, CA, USA, November 02 - 08, 2003.

2. D. da Silva Pires, R. M. Cesar, M. B. Vieira, and L. Velho, "Tracking and Matching Connected Components from 3D Video," *Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on* , vol., no., pp. 257-264, 09-12 Oct. 2005

3. D. Comaniciu and P. Meer, “Mean shift analysis and applications,” *Proc. of the Seventh IEEE International Conference on Computer Vision,* Vol. 2, 1999, pp. 1197–1203.

4. D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift,” *IEEE International Conference on Pattern Recognition*, Vol. 2, 13–15 June, 2000, pp. 142–149.

5. B. Zhang, W. Tian, and Z. Jin, “Joint tracking algorithm using particle filter and mean shift with target model updating, “ Chin Opt Lett 4, 569–572, 2006.

6. Chris Stauffer, W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99)* - Volume 2, pp. 2246, 1999.

7. F. Jiang, Y. Wu, and A.K. Katsaggelos, "Abnormal event detection based on trajectory clustering by 2-depth greedy search," *IEEE Int'l Conf. on Acoustics, Speech, and Signal Process. (ICASSP’08),* Las Vegas, NV, March 2008.

8. F. Jiang, Y. Wu, and A.K. Katsaggelos, "Abnormal event detection from surveillance video by dynamic hierarchical clustering," *IEEE Int'l Conf. on Image Process. (ICIP’07)*, vol. 5, San Antonio, TX, 145-148, September 2007.

9. F. Jiang, Y. Wu, and A.K. Katsaggelos, " A dynamic hierarchical clustering method for trajectory-based unusual video event detection," *IEEE Trans. Image Process.*, vol. 18, issue 4, 907-913, 2009.

10. V. Chandola and A. Banerjee and V. Kumar, “Anomaly detection: a survey,” to appear in *ACM Computing Surveys*, 2009.

11. F. Jiang, Y. Wu, A. K. Katsaggelos, “Detecting contextual anomalies of crowd motion in surveillance video”, *Proc. IEEE Int'l Conf. on Image Processing (ICIP’09)*, Nov. 2009.

12. S. Ali and M. Shah, “A Lagrangian particle dynamics approach for crowd flow segmentation and stability analysis,” *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition*, June 2007, pp. 1–6.

13. M. Hu, S. Ali, and M. Shah, “Learning motion patterns in crowded scenes using motion flow field,” *Proc. IEEE Int’l Conf. on Pattern Recognition*, Dec. 2008, pp. 1–5.

14. N. Ihaddadene and C. Djeraba, “Real-time crowd motion analysis,” *Proc. IEEE Int’l Conf. on Pattern Recognition*, Dec. 2008, pp. 1–4.

15. A.M. Cheriyadat and R.J. Radke, “Detecting dominant motions in dense crowds,” *IEEE Journal of Selected Topics in Signal Process.,* vol. 2, no. 4, pp. 568–581, Aug. 2008.

16. A. B. Chan and N. Vasconcelos, “Modeling, clustering, and segmenting video with mixtures of dynamic textures,” *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 30, no. 5, pp. 909–926, May 2008.

17. X. Wang, X. Ma, and W. E. L. Grimson, “Unsupervised activity perception in crowded and complicated scenes using hierarchical Bayesian models,” *IEEE Trans. Pattern Anal. Mach. Intell*., vol. 31, no. 3, pp. 539–555, Mar. 2008.

18. S. Soatto, G. Doretto, and Y. N. Wu, “Dynamic textures,” *Proc. IEEE Int’l Conf. on Comput. Vision*, July 2001, vol. 2, pp. 439–446.

19. R. J. Martin, “A metric for ARMA processes,” *IEEE Trans. Signal Process.*, vol. 48, no. 4, pp. 1164–1170, Apr. 2000.

20. J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal and M-C Hsu, “Mining sequential patterns by pattern-growth: the PrefixSpan approach”, *IEEE Trans. Knowl. Data Eng*., vol. 16, no. 11, pp. 1424 – 1440, Nov. 2004.

21. T. Uno, M. Kiyomi, H. Arimura, “LCM ver.2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets,” *Proceedings of IEEE ICDM'04 Workshop FIMI'04*, Nov. 2004.

22. F. Jiang, J. Yuan, S. Tsaftaris and A.K. Katsaggelos, “Anomalous video event detection using spatiotemporal context,” submitted to Computer Vision and Image Understanding Special Issue on Feature-oriented Image and Video Computing for Extracting Contexts and Semantics, 2010

APPENDIX: MANUAL FOR THE TRACKING SOFTWARE

The tracking software, called **multitrack**, can be called using the following scheme:

Multitrack [fg=<fg_name>] [bd=<bd_name>] [bt=<bt_name>] [btpp=<btpp_name>] [bta=<bta_name>] [bt_corr=<bt_corr_way>] [btgen=<btgen_name>] [track=<track_file_name>] [FGTrainFrames=<FGTrainFrames>] [btavi=<avi output>] [fgavi=<avi output on FG>] <input_avi_file>

A typical run of the algorithm is as follows:

multitrack FGTrainFrames=30 btpp_name=Kalman track=trajectory_output.txt fgavi=foreground_video.avi btavi=video_output.avi video_input.avi

All options except the *input_avi_file* are optional. The optional parameters are used to change parameters of the modules which are presented above. The explanations of these parameters are given below for each module:

<fg_name>: "FG/BG Detection" module

Algorithm selection:

1. *FG_0* - Foreground Object Detection from Videos Containing Complex Background [1] (see Section 3)
2. *FG_0S* - Simplified version of FG_0
3. *FG_1* - Adaptive background mixture models for real-time tracking. [6]

Common Algorithm Parameters:

<FGTrainFrames> : number of frames for FG training

<bd_name>: "Blob Entrance Detection" module

Algorithm selection:

1. *BD_CC* - Detect new blob by tracking connected components (CC) in the FG mask (see Section 4)
2. *BD_Simple* - Detect new blob by uniform moving of connected components of FG mask

---

<bt_name> : "Blob Tracking" module

Algorithm selection:

1. *CCMSPF* - connected component tracking with MSPF for collision resolution (see Section 5)
2. *CC* - Simple connected component tracking
3. *MS* - Mean shift algorithm
4. *MSFG* - Mean shift algorithm using the FG mask
5. *MSPF* - Particle filtering based on MS weight (see Section 5)

<btpp_name>: "Blob Trajectory Post Processing" module

Algorithm selection:

1. *Kalman* - Kalman filtering of blob position and size (see Section 6)
2. *None* - No post processing filter

<btgen_name>: "Blob Trajectory Generation" module

Algorithm selection:

1. *RawTracks* - Generate raw track record (x,y,sx,sy),()... in each line
2. *YML* - Generate track record in YML format as synthetic video data

Common Algorithm Parameters:

<track>: file name to save the tracked trajectories.