3

Case Study: IBM Smart Surveillance System

Rogerio S. Feris, Arun Hampapur, Yun Zhai, Russell Bobbitt, Lisa Brown, Daniel A. Vaquero, Ying-li Tian, Haowei Liu, and Ming-Ting Sun

CONTENTS

Urban environments present unique challenges from the perspective of surveillance and security. High volumes of activity data, different weather conditions, crowded scenes, widespread geographical areas, and many other factors pose serious problems to traditional video analytics algorithms and opens up opportunities for novel applications. In this chapter, we present the IBM Smart Surveillance System, including our latest computer vision algorithms for automatic event detection in urban surveillance scenarios. We cover standard video analysis methods based on tracking and also present non-tracking-based analytics specifically designed to handle crowded environments. Large-scale data indexing methods and scalability issues are also covered. Our experimental results are reported on various datasets and real-world deployments.

3.1 INTRODUCTION

Security incidents in urban environments span a wide range, starting from property crimes, to violent crimes and terrorist events. Many large urban centers are currently in the process of developing security infrastructures geared mainly to counter terrorism with secondary applications for police and emergency management purposes.

Applying video analytics in urban scenarios involves many challenges. Urban surveillance systems generally encompass a large number of cameras spanning different geographical areas and capturing hundreds of millions of events per day. Storing and enabling search for events in these large-scale settings is a difficult task. In addition, video analytics algorithms need to be robust to many typical urban environment conditions, such as weather changes (e.g., rain, snow, strong shadow effects), day/night times, and crowded scenes. The need to minimize cost by maximizing the number of video feeds per machine also imposes critical constraints in the time and memory complexity of the algorithms.

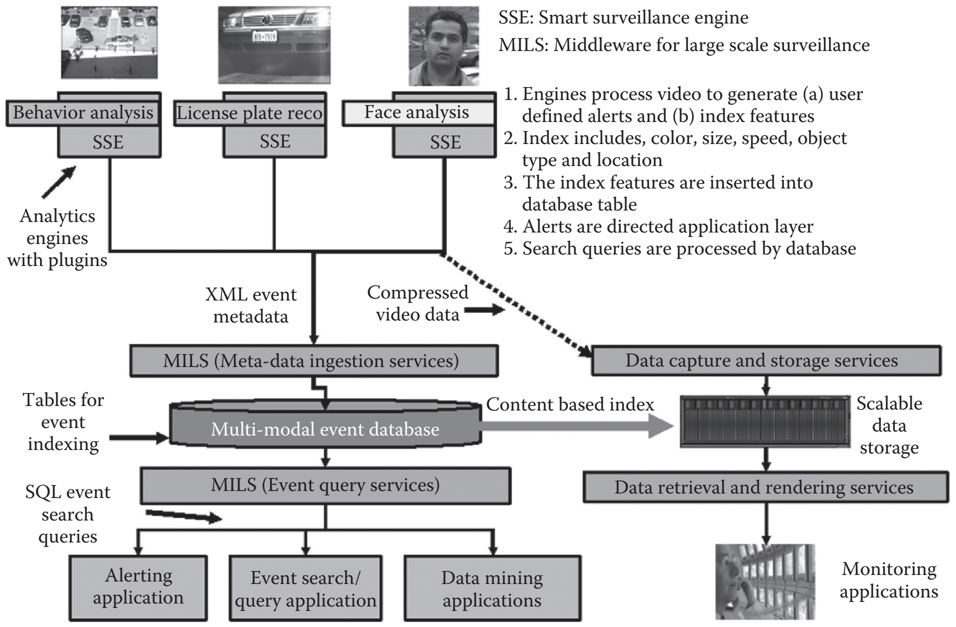
In this chapter, as part of the IBM Smart Surveillance System (IBM SSS), we present our latest video analytics methods for monitoring urban environments. We also analyze scalability factors and large-scale indexing for event search. Next, we describe the main architecture of IBM SSS and give an overview of the main sections of this chapter.

3.1.1 IBM Smart Surveillance System Architecture

Figure 3.1 shows the architecture of the IBM Smart Surveillance Solution. IBM SSS is designed to work with a number of video management systems from partner companies. The video analytics technology within the SSS system provides two distinct functionalities:

- *Real-time user-defined alerts:* The user defines the criteria for alerting with reference to a specific camera view, for example, parked car detection, tripwire, etc. (see Section 3.3).
- *Indexed event search:* The system automatically generates descriptions of events that occur in the scene and stores them in an indexed database to allow the user to perform rapid search (see Section 3.4).

The video analytics-based index refers to specific clips of video that are stored in the video management system. The SSS system has video analysis engines called SSEs (smart surveillance engines), which host a


SSE: Smart surveillance engine
MILS: Middleware for large scale surveillance
1. Engines process video to generate (a) user defined alerts and (b) index features
2. Index includes, color, size, speed, object type and location
3. The index features are inserted into database table
4. Alerts are directed application layer
5. Search queries are processed by database
Behavior analysis
SSE
License plate reco
SSE
Face analysis
SSE
Analytics engines with plugins
XML event metadata
Compressed video data
MILS (Meta-data ingestion services)
Data capture and storage services
Tables for event indexing
Multi-modal event database
Content based index
Scalable data storage
SQL event search queries
MILS (Event query services)
Data retrieval and rendering services
Alerting application
Event search/ query application
Data mining applications
Monitoring applications

AQ1 **Figure 3.1 (See color insert following page xxx.)** IBM SSS architecture.

number of video processing algorithms. The SSEs generate metadata, which gets automatically uploaded into the backend database system through a Web-based service-oriented architecture. The application layer of SSS combines the metadata from the database with video from the video management system, to provide the user with a seamless view of the environment of interest. Details of IBM SSS can be found in several previous publications (Shu et al. 2005).

3.1.2 Chapter Overview

This chapter is organized as follows. In Section 3.2, we describe our video analytics modules for urban surveillance, including moving object detection, tracking, object classification, two approaches for color retrieval targeting low- and high-activity-level scenarios, and a people search method based on fine-grained human parts and attributes. In Section 3.3, we present a set of real-time alerts which work well in crowded scenes, including parked car detection and non-tracking-based virtual boundary crossing. Finally, we cover our database design for large-scale searching of video in Section 3.4 and analyze practical challenges with respect to scalability issues in Section 3.5.

3.2 VIDEO ANALYTICS FOR URBAN SURVEILLANCE

3.2.1 Moving Object Detection

Background subtraction (BGS) is a conventional and effective approach to detect moving objects in videos captured by fixed cameras. In urban environments, more sophisticated background modeling algorithms are required to handle snow, rain, shadows, reflections, quick lighting changes, day and night transitions, slow-moving objects, and other difficult conditions that naturally arise in city surveillance scenarios.

In order to address these issues, we use an adaptive background subtraction technique similar to the method proposed by Stauffer and Grimson (1999), which relies on a Gaussian mixture model for each pixel in the scene. In this way, multimodal backgrounds can be described by the pixel-wise mixture models. For instance, during snow conditions, a particular background pixel may be represented by two Gaussian encodings, for example, the mode "road" and another mode "snow," respectively, so that only objects of interest are detected.

In contrast to Stauffer and Grimson (1999), our method is improved to remove shadows and to enable the algorithm to work for quick lighting changes by combining the intensity and texture information of the foreground pixels. In addition, the foreground regions are classified as moving objects, abandoned objects, or removed objects without using any tracking or motion information (see Tian et al. 2005 for details). This capability not only can avoid a common problem in background subtraction—fragmentation (one object is partitioned into multiple parts), but also allows real-time alert detection such as abandoned and removed object detection alerts (see more details in Section 3.3.1).

3.2.2 Object Tracking

After a number of moving objects are detected by our background subtraction algorithm, we track them along the video sequence so that each object is given a unique identification number. This is a prerequisite for other higher-level modules, such as object and color classification, and several real-time alerts. As we will show later, in crowded urban environments, tracking turns out to be a very difficult and computationally expensive task, so it can be intentionally disabled at specific hours of the day when the activity level is high. In these cases, analytics that do not rely on tracking are enabled (see, e.g., Sections 3.2.4.3 and 3.3.2).

The first step of our tracking method consists of simple association of foreground blobs across the temporal domain. In each frame, we create a list of the foreground blobs found by the background subtraction algorithm. From preceding frames, we have a list of recent tracks of objects, together with their bounding boxes. Each foreground blob in the current frame is then matched against all existing object tracks based on the distance and areas of the corresponding bounding boxes, so that tracks can be updated, created, or deleted.

In a simple scene, with well-separated tracks, the association gives a one-to-one mapping between tracks and the foreground regions in the current frame, except when tracks are created or deleted. In more complex scenes, however, we may have situations where a single track is associated with more than one foreground blob (object split), or the opposite—several tracks can be associated with a single foreground region (object merge). In addition, when one object passes in front of another, partial or total occlusion takes place, with background subtraction detecting a single moving region. The tracking method should be able to segment this region, label each part appropriately, and label the detected objects when they separate.

We resolve these issues using an *online appearance model* which consists of a 2D array of color values with an associated probability mask, as described in Senior et al. (2006). This appearance model is used to resolve object splits and merges, handle occlusions, and provide depth ordering. We refer the reader to Senior et al. (2006) for more details and performance evaluation.

3.2.3 Object Classification with Calibration

Classifying objects in urban surveillance scenes is an important task that allows searches and alerts based on object type. In this section, we address a simplified two-class object recognition problem: Given a moving object in the scene, our goal is to classify the object into either a person (including groups of people) or a vehicle. This is a very important problem in city surveillance, as many existing cameras are pointing to areas where the majority of moving objects are either humans or vehicles. The classification problem is very challenging as we desire to satisfy the following requirements:

- Real-time processing and low memory consumption
- The system should work for arbitrary camera views
- Correct discrimination under different illumination conditions and shadow effects
- Able to distinguish similar objects (such as vehicles and groups of people)

Our approach to address these issues consists of three elements: (1) an interactive interface to set regions of interest and correct for perspective distortions; (2) discriminative features, including a novel effective measurement based on differences of histograms of oriented gradients (DHOG); and (3) estimation and adaptation based on a probabilistic framework for feature fusion. We briefly describe these modules in the following text.

3.2.3.1 Interactive Interface for Calibration

In many situations, objects of one class are more likely to appear in certain regions in the camera view. For instance, in the city street environment, people usually walk along the sidewalk, while on the other hand vehicles mainly run in the middle of the road. This is a strong cue in the object classification process. In our system, we classify tracked objects into two classes, people and vehicles, and users can specify the regions of interest (ROIs) of each object class in the camera view through our calibration tool. In this interactive calibration tool, one or multiple ROIs

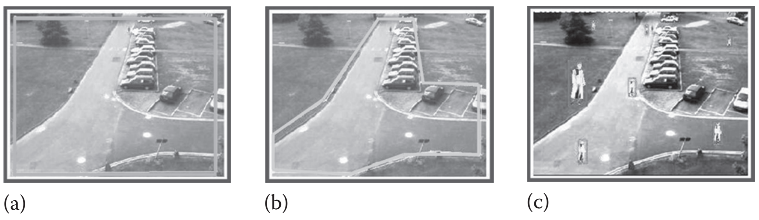


Figure 3.2 Calibration tool interface. (a) ROI for people—entire camera view. (b) ROI for vehicles–driveways. (c) Person-sized models specified by the user. In (c), the user can create, move, and resize a target size sample (the largest box in the figure).

for the target class can be created, modified, and deleted as needed. Screenshots of the interface are shown in Figure 3.2a and b. Note that AQ2 this information is just used as a prior for the location feature. The final class estimation is based on probabilistic fusion of multiple features, as we will describe later.

Another purpose of our calibration tool is to correct for perspective distortions by specifying different sizes in different locations of the camera field of view. A continuous, interpolated size map is then created from the sample sizes specified by the user. Since we use view-dependent features such as object size and velocity, among others, for classification, the use of this information in our system allows it to normalize these features and thus work for arbitrary camera viewpoints, while significantly improving the accuracy of classification. Figure 3.2c shows our interface.

3.2.3.2 Feature Extraction

Given the limited computational resources and the real-time requirement in practical video surveillance applications, the features used for object classification must be low cost and efficient for computation. In our framework, we have used four object features: object size, velocity direction, object location, and differences of histograms of oriented gradients (DHOG).

The purpose of using *object size* is that size information is the most distinctive feature to distinguish single persons from vehicles since persons possess much smaller shapes than vehicles at the same location in the field of view. The size of an object at a specific frame is computed as the area of the corresponding foreground blob and then normalized by the size obtained from the interpolated size map (previous section) at the same position. AQ3

In many scenarios, *velocity direction* of moving objects can be a distinctive feature. For instance, at a street intersection, pedestrians typically walk along the zebra crossings which are perpendicular to vehicle movements. We equally discretize the velocity direction measurement into 20 bins and normalize it in a similar way to the object size feature.

There is an additional straightforward yet very informative feature: the *location* of the moving object. The location feature relates to the context of the environment, and its usage is applied through the settings of regions of interest specified by our calibration tool. This is a strong cue for identifying people in views such as roads and building entrances where vehicles seldom appear.

Lastly, we have developed a novel view-independent feature—differences of histograms of oriented gradients (DHOG). Given an input video image with a foreground blob mask generated by the background subtraction module, the histogram of oriented gradients (HOG) is computed. It is well known that the HOG is robust to lighting condition changes in the scene. The HOG of the foreground object is computed at every frame in the track, and the DHOG is calculated in terms of the difference between HOGs obtained in consecutive frames in terms of histogram intersection. The DHOG models the intra-object deformation in the temporal domain. Thus, it is invariant to different camera views. In general, vehicles produce smaller DHOG than people since vehicles are more rigid when in motion. This feature is useful to distinguish large groups of people from vehicles, in which case they have similar shapes and sizes. Examples are shown in Figure 3.3 to demonstrate the effectiveness of using DHOG to distinguish vehicles from people (both single persons and groups of people). DHOG features have similar motivation as the recurrent motion image method (Javed and Shah 2002), but offer advantages as no precise object alignment is required.

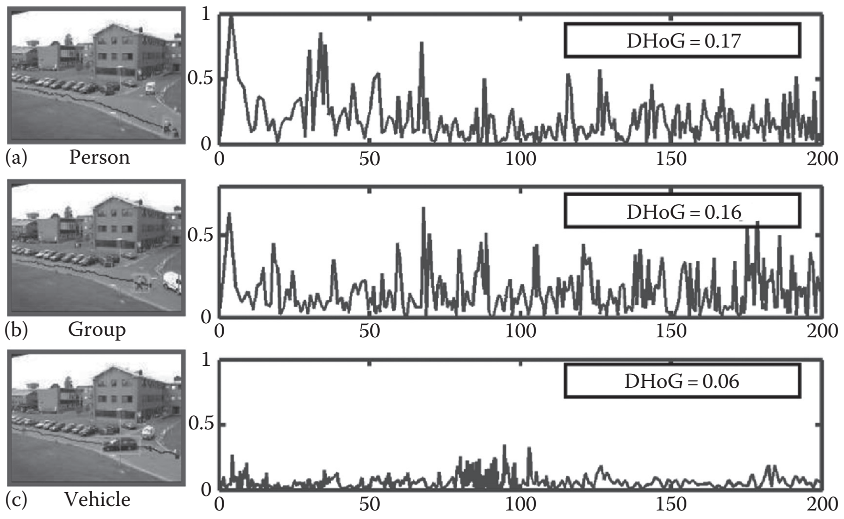


Figure 3.3 DHoG plots for three types of objects: (a) person, (b) group of people, and (c) vehicle. Horizontal axis represents the track length of the samples, and vertical axis represents the DHoG values. Note that the overall DHoG values for person and group of people are much higher than the one for vehicle, and thus, it is a very discriminative feature to separate people from vehicles.

3.2.3.3 Estimation and Adaptation

We pose the moving object classification task as a maximum a posteriori (MAP) problem. The classification is performed by analyzing the features of the entire object track, that is, the classification decision is made after the tracking is finished. We refer the reader to Chen et al. (2008) for a detailed explanation of our MAP inference and probabilistic feature fusion. In that work, we also provide details about our model adaptation mechanism, which makes use of high confidence predictions to update our probabilistic model in an online fashion.

3.2.3.4 Results

To demonstrate the effectiveness of our proposed object classification framework in far-field urban scenarios, we have tested our system on a variety of video data, including different scenes and different views of the same scene. There are two urban data sets used in our experiments. The first set is the PETS 2001 dataset. The second data set is a collection of videos obtained from different sources, ranging from 15 min to 7 h long.

Figure 3.4 presents the object classification results on both testing data sets and shows a few key frames of testing videos. The overall classification accuracy is 97.7% for over 1100 moving objects. Currently, our framework assumes tracking results are perfect before performing object classification. Thus, if the background subtraction and/or tracking module fail to produce reliable results, the object classifier will also fail due to incorrect input data. This is the main reason for the lower performance on the sequences “PETS D3TeC1” and “PETS D3TeC2.”

3.2.3.5 Discussion

Similar to far-field object classification approaches such as Bose and Grimson (2004) and Brown (2004), our method relies on background

Videos	Ground Truth		People Detection		Vehicle Detection	
	People	*Vehicles*	*True Positives*	*False Positives*	*True Positives*	*False Positives*
PETS D1TeC1	6	2	6	0	2	0
PETS D1TeC2	5	1	5	0	1	0
PETS D2TeC1	6	3	6	0	3	0
PETS D2TeC2	7	2	6	0	2	1
PETS D3TeC1	16	0	10	0	0	6
PETS D3TeC2	13	0	6	0	0	7
Sequence HW1	16	82	14	1	81	2
Sequence HW2	67	22	66	0	22	1
Traffic complex	125	696	121	0	696	4
Traffic videoD	44	46	40	1	45	4
Shadow	3	1	3	0	1	0

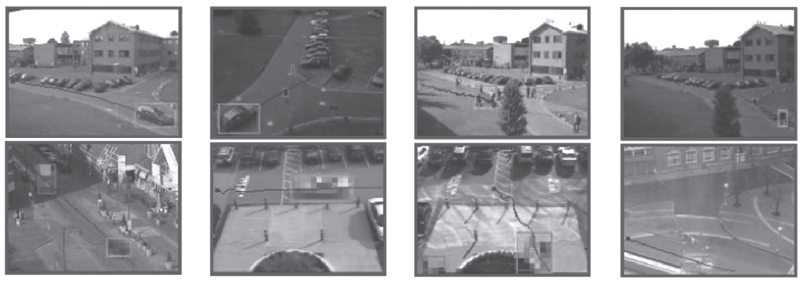


Figure 3.4 (See color insert following page xxx.) Results for our object classification approach and some testing video frames.

subtraction and object tracking to perform object classification. This works reasonably well in urban scenes with low or medium activity. However, crowded scenarios pose serious problems for our approach as many objects can be merged together in a single foreground blob, causing features like object size to be unreliably estimated from BGS. An alternative solution to this problem is to use appearance-based detectors such as those proposed by Dalal and Triggs (2005) or Viola and Jones (2001). The disadvantage of these approaches is that they often require large amounts of data to learn a robust classifier and suffer from object pose variability.

3.2.4 Color Classification

In this section, we describe our method to retrieve objects of specified colors. We start with our color quantization algorithm that operates on a frame by frame level. Then, we describe our algorithm for color classification that is built on top of the tracking module, thus providing a specific color for each object track in the scene. Finally, we show a method that enables color search without relying on tracking, which is appropriate for crowded urban environments where tracking is not reliable.

3.2.4.1 Bi-Conic Color Quantization

In our system, color information is quantized into six colors: black, white, red, blue, green, and yellow. For each video frame, we first convert the 24 bit RGB pixels to the bi-conic hue, saturation, lightness (HSL) space. This color space is used because Euclidean distances in this space closely map to human perception color differences. Figure 3.5 shows an illustration of the HSL space—the vertical axis represents "lightness"; this ranges from white (full brightness) to black. Hues are represented at different angles and color saturation increases radially.

In order to quantize this space into a small number of colors, we determine the angular cutoffs between colors. Since we use six colors (black, white, yellow, green, blue, red), we need only four cutoffs between the hues: yellow/green, green/blue, blue/red, and red/yellow. Empirically, we determined the following effective cutoffs: 60°, 150°, 225°, and –15°, which worked for most outdoor urban scenes.

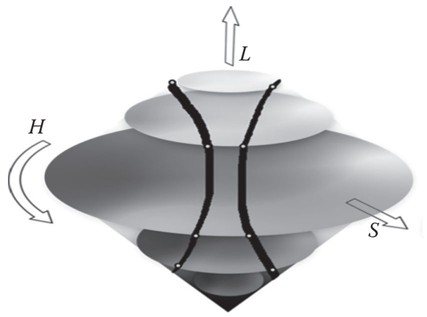


Figure 3.5 (See color insert following page xxx.) Hue/saturation/lightness (HSL) color space showing saturation/lightness curves and threshold points.

Since the time of day and weather conditions affect the brightness and corresponding color saturation, we also need to specify lightness and saturation cutoffs. It is interesting to note, here, that saturation and intensity are related. Both properties can make the hue of a pixel indiscernible. For intensity, this occurs when the light is too bright or too dark. For saturation, it happens when there is insufficient saturation. However, as the brightness (we refer to the intensity as the "lightness" or "brightness" interchangeably) gets too low or too high, the necessary saturation increases. In general, as intensity increases from zero up to halfway (the central horizontal cross section of the bi-conic) or decreases from the maximum (white) down to halfway, the range of pixels with visible or discernable hue increases.

This is shown by the 2D intensity/saturation curves in Figure 3.5. These 2D curves represent 3D curves, which are circularly symmetric since they are independent of hue. Because of this symmetry and the coarse quantization needed, these curves can be represented by a small number of 2D (lightness, saturation) points which are marked on the curve. Above the horizontal plane (i.e., for sufficient lightness), points whose intensity and saturation lie outside the curve are considered white. Below the horizontal plane, they are considered black.

In summary, we first quantize HSL space based on hue. We subsequently relabel pixels either white or black depending on whether they lie outside the lightness/saturation curve above or below the horizontal mid-plane. This is related to earlier work in color segmentation performed by Tseng and Chang (1994).

3.2.4.2 Tracking-Based Color Retrieval

Each moving object is tracked as it moves across the scene. The object is first detected by background subtraction and tracked using an appearance-based tracker, as described previously. For each track, the color classifier creates an accumulated histogram of the six quantized colors and then selects the dominant color of the object, that is, the color with the largest number of votes in the histogram. Because of the computational cost, we do not consider every frame of the tracked object. The system is performing real-time background subtraction and tracking as well as other high-level analysis such as object classification and alert notification; consequently it is important to conserve resources and computational cost. Instead, the classifier subsamples the frames of the object track in real time. Since we do not know ahead of time how long the track will be, the classifier initially acquires color information frequently and gradually decreases this frequency as the track's life continues. In this way, we ensure that there are sufficient samples for short tracks, and periodically update the samples so that a spread of data throughout the track life of the track is considered for longer tracks.

We ran tests of the color classifier for tracked objects on two datasets with low or medium activity. The first video is a 12 min scene of 194 vehicles driving along in one direction on a two-lane highway. The second scene is from a live video from a parking lot. In this scene, we captured 72 vehicles driving by. Note that in these scenes, object tracking was feasible although the actual segmentation of the vehicle from the background did include significant portions of the road. Figure 3.6 shows results for these two datasets and Figure 3.7 shows sample key frames. The results are pretty good, but there is still some confusion between black and white on the second test data. We believe this is due to the variable amount of road (that appeared white) that was present in the tracked objects.

3.2.4.3 Non-Tracking-Based Color Retrieval

In order to handle high-activity scenes, which are typical in certain hours of the day in urban scenarios, our system also has the capability to perform color retrieval without object tracking. Our method relies

	True Positives	False Positives	Bad Segments
Red	11	0	0
Yellow	2	0	0
Green	4	0	0
Blue	0	0	0
White	83	0	3
Black	94	0	2
Total	194	0	5
Percent	**100%**	**0%**	

(a)

	True Positives	False Positives	Bad Segments
Red	4	0	0
Yellow	0	0	0
Green	1	0	0
Blue	4	0	0
White	24	4	1
Black	29	6	2
Total	62	10	3
Percent	**86%**	**14%**	

(b)

Figure 3.6 Results of tracking-based color classification on low/medium activity scenes. (a) Highway test set. (b) Parking lot test set.

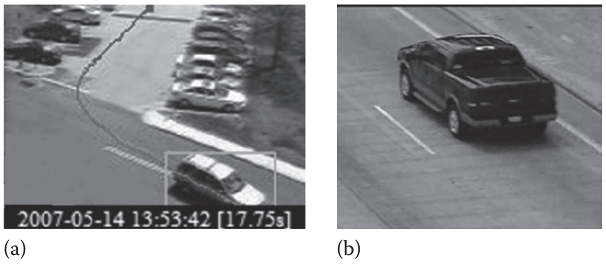


Figure 3.7 (a) Parking lot sample frame with tracking information superimposed. (b) Sample key frame of the highway scene.

on color segmentation inside the foreground objects detected using background subtraction. The system uses a time interval, typically 2 or 3 s, and a size threshold per color, set by the user.

For each foreground object, we quantize the colors, as described in Section 3.2.4.1 and perform segmentation using connected component analysis for each color. In each time interval (say, 3 s), we detect, for each color, if a connected component of that color is found which is bigger than the predefined size threshold. If such a component is found, we store the largest component for that color in the time interval as the key frame for color retrieval.

We tested our method in a crowded street intersection, as shown in Figure 3.8a. As it is difficult to obtain ground truth data for these

AQ4 **Figure 3.8** (a) Crowded scene. (b) Examples of non-tracker-based color retrieval, where red bag parts and green hats can be found by the user.

scenes, we only compute the precision for our method. We obtained 78% accuracy with few false alarms, mostly due to slow-moving objects, which generate multiple events, since tracking is not applied. Note that the method described in Section 3.2.4.2 is not suitable for this crowded scenario, due to unreliable tracking. Another advantage of our non-tracking-based color retrieval is the capability of finding colored parts of objects, such as green hats and red bags, as shown in Figure 3.8b.

3.2.5 Face Capture and Human Parsing

In urban environments, there are several scenarios in which it is desirable to monitor people's activities. Applications such as finding suspects or locating missing people involve searching through large amounts of video and looking for events where people are present. This process can be tedious, resource consuming, and prone to error, incurring high costs of hiring security personnel. In a previous work (Feris et al. 2007), we described our face capture system that automatically detects the presence of people in surveillance scenes by using a face detector. The system is suitable for environments where there is enough resolution to detect faces in images. Examples include cameras placed at ATM cabins or entrances of buildings.

The face capture system greatly reduces the search time by enabling the user to browse only the video segments that contain faces; however, finding a specific person among the extracted video clips can still be time consuming, especially in high traffic environments. Research on finding people in videos has been focused on approaches based on face recognition (Sivic et al. 2005), which is still a very challenging problem, especially in low-resolution images with variations in pose and lighting. State-of-the-art face recognition systems (http://www.frvt.org/) require a fair amount of resolution in order to produce reliable results, but in many cases this level of detail is not available in surveillance applications.

As a solution to overcome these difficulties, we have developed an approach for people search based on fine-grained personal attributes such as facial hair type, eyewear type, hair type, presence of hats, and clothing color. We leverage the power of machine learning techniques (Viola and Jones 2001) by designing attribute detectors from large sets of training images. Our technique enables queries such as "show me the people who entered the IBM Hawthorne building last Saturday wearing a hat, sunglasses, a blue jacket, and black pants," and is based on parsing the human body in order to locate regions from which attributes are extracted. Our approach is described in detail in another chapter of this book.

3.3 REAL-TIME ALERTS

In this section, we describe a set of real-time alerts provided by our system, which are useful to detect potential threats in urban environments. Later, in Section 3.4, we describe another mode of operation—indexing and search—which allows the user to find after-the-fact events for forensics.

3.3.1 Parked Car Detection

Automatically detecting parked cars in urban scenarios can help security guards to quickly identify illegal parking, stopped vehicles in a freeway, or suspicious cars parked in front of security buildings at certain times. We developed a robust parked car detection system which is currently running in the city of Chicago, using various video cameras capturing images under a wide variety of conditions, including rain, day/night scenarios, etc. Our approach is similar to the technique presented in Tian et al. (2008), but has novel aspects, which substantially reduce the rate of false alarms, as we will see next.

We use the background modeling algorithm described in Section 3.2.1 to detect foreground blobs in the scene. We also detect when these foreground blobs are static for a period of time, since our goal is to detect stopped vehicles. An important observation is that static foreground blobs may arise in the scene due to abandoned objects (e.g., parked cars) or due to removed objects (e.g., a car leaving a parking lot). As described in Tian et al. (2008), we provide a method to classify whether a static foreground region is abandoned or removed by exploiting context information about the static foreground regions. In case the region is classified as abandoned object, then a set of user-defined requirements is verified (e.g., minimum blob size and ROI) and a template-matching process is started to confirm that the object is parked for a user-defined period of time. If these conditions are satisfied, a parked car detection alert is triggered.

The method described above (Tian et al. 2008) is enhanced in two novel ways. First, we use a naïve tracking algorithm (only across a user-defined alert region of interest) to improve the classification of whether an object is abandoned or removed, for example, by looking at direction of movement and object speed as features. The second improvement addresses nighttime conditions where the template-matching process can fail due to low-contrast images. Note that this matching scheme is designed to verify whether the vehicle is parked for the minimum time specified by the user. In many nighttime cases the car leaves the scene before this specified time, but the matcher still reports the car is there due to low contrast. To handle this problem, we verify whether we have removed object notifications during the matching step, and, if so, we remove the candidate parked car alert.

Our approach substantially reduces the number of false alarms in challenging conditions, involving high volumes of data and weather changes, while keeping high detection rates. We tested our system in 53 h of data split into five different videos. Figure 3.9 shows our results, comparing with the method presented in Tian et al. (2008). We are not allowed to show sample key frames of our test data due to customer-related agreements.

3.3.2 Virtual Boundary Crossing

Virtual boundaries (often referred as tripwires) are defined as continuous lines or curves drawn in the image space to capture directional

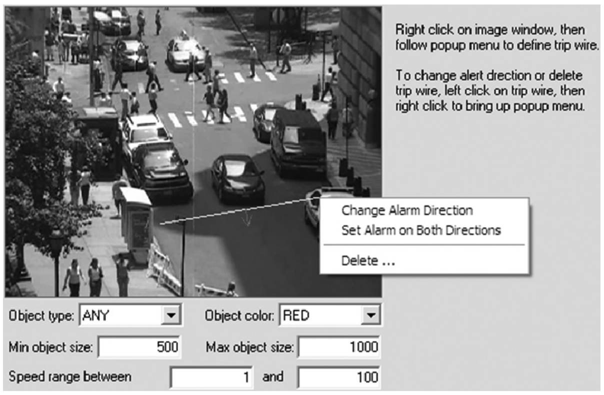
	Video 1		Video 2		Video 3		Video 4		Video 5		Total	
	Tian et al.(2008)	Our method	Tian et al.(2008)	Our method	Tian et al.(2008)	Our method	Tian et al.(2008)	Our method	Tian et al.(2008)	Our method	Tian et al.(2008)	Our method
TP	5	5	13	12	10	8	16	16	22	20	66	61
FP	5	3	10	3	16	0	8	5	14	5	53	16
FN	1	1	1	2	0	2	8	8	5	7	15	20
Detect rate	0.83	0.83	0.93	0.86	1	0.8	0.66	0.66	0.81	0.74	0.81	0.75
FP/h	0.47	0.28	0.90	0.27	1.5	0	0.76	0.47	1.33	0.38	1	0.3

Figure 3.9 Parked car detection evaluation in the city of Chicago, using 53 h of video data captured under challenging conditions (day, night, rain, etc.).

crossings of target objects. Due to their intuitive concepts and reliable performance under well-controlled conditions, they are widely used in many surveillance applications, such as detecting illegal entries in border security and display effectiveness estimation in the retail sector.

In urban surveillance, we mainly utilize the tripwires for vehicle counting, wrong-way traffic detection, building perimeter protection, etc. Different attribute-based filters can be applied to the tripwire definition to constrain the search range. For instance, the object size range specifies the type of objects to be counted, that is, single persons have much smaller sizes than vehicles, and vehicles like buses and 18-wheeler trucks are much larger than regular personal vehicles. Other types of features are also deployed, including object type (person/vehicle), object color, and minimum and maximum object speeds. The tripwire is configured as an alert feature in the IBM SSS solution, and a graphical user interface for setting up the tripwire alert is shown in Figure 3.10.

In heavy traffic situations, vehicles are often occluded from each other and also piled up together if a traffic light is present or congestion occurs. In this case, reliable object detection and tracking are very difficult to achieve. Consequently, the tripwire crossing detections that are based on tracking results are erroneous and barely meaningful. To overcome this shortcoming, particularly in the urban surveillance applications, we developed a new type of tripwire, called appearance-based tripwire (ABT), to replace the conventional tracker-based tripwires.


Right click on image window, then follow popup menu to define trip wire.

To change alert direction or delete trip wire, left click on trip wire, then right click to bring up popup menu.

Object type: ANY
Object color: RED
Min object size: 500
Max object size: 1000
Speed range between 1 and 100

Figure 3.10 Graphical user interface of IBM SSS tripwire alert.

The ABT does not rely on the explicit object tracking output. Instead of considering object trajectories in the entire camera field of view (FOV), the appearance-based tripwire reduces its focus of analysis to the surrounding area of the tripwire. This significantly eliminates the interference of the noisy data in the tripwire crossing detection inference. In the ABT, a set of *ground patches* (GP) are sampled around the tripwire's envelope, which covers a spatial extension of the tripwire (a graphical illustration is shown in Figure 3.11).

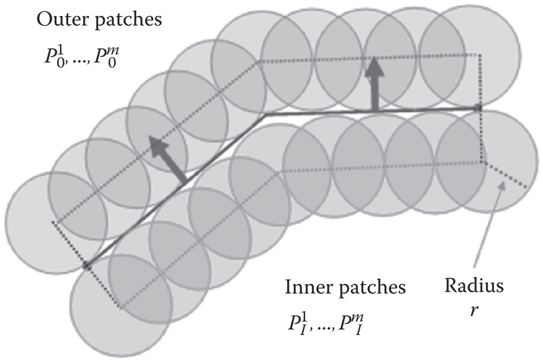


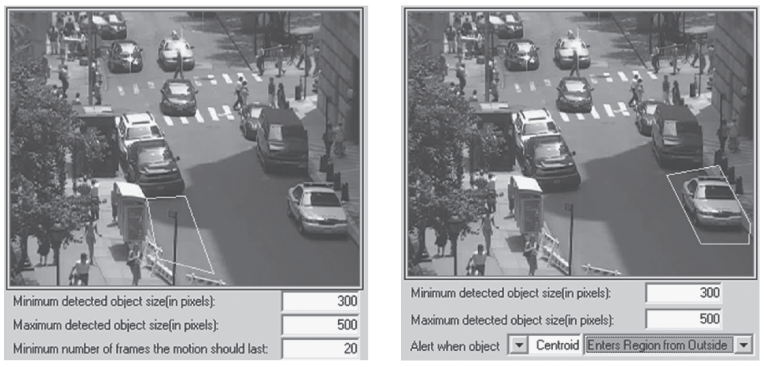
Figure 3.11 (See color insert following page xxx.) A graphical illustration of the appearance-based tripwire. Ground patches are generated to model the appearance and motion patterns of the tripwire region.

Appearance and motion direction features are extracted from each patch region over a history of time. Patches on the opposite sides of the tripwire are compared against each other. Tripwire crossings are detected by analyzing the patch matching candidates with spatiotemporal constraints. The appearance-based tripwire is compared with the conventional tracking-based tripwire in both low traffic and crowded scenes. In low traffic scenes, the object detection module is able to achieve good spatiotemporal segmentation of objects. Thus, both tripwire types have similar high accuracy of more than 90% at the equal error rate (EER). The superior performance of ABT is demonstrated in crowded scenes where tracking does not work well. Based on a comparison set for vehicle counting tasks, the proposed ABT presents its ability to capture over 20% more vehicles than tracking-based tripwires, in average. Therefore, the appearance-based tripwire can be a replacement for the conventional tripwire in high-activity scenarios, maintaining the system's performance.

3.3.3 Motion and Region Alerts

Motion detection and region alerts are two other important features offered by the IBM Smart Surveillance Solution. Motion detection alerts are triggered when the target region-of-interest (ROI) possesses a sufficient amount of motion energy that lasts within the desired temporal interval. Applications of this feature include loitering detection, ROI occupancy estimation, and object access detection. In urban scenes, it could be used as a simplified version of the abandoned object alert, where the parked vehicles are detected by specifying an ROI around the parking area. One thing to note is that motion detection alerts consider the global motion energy of the ROI without distinction of individual objects.

The region alert, on the other hand, provides a set of different functionalities that are based on the object tracking output. A ROI is configured to represent the target region. Different rules can be specified to capture the region alert, such as object initiated inside/outside the region, object passing through the region, object entering the region from outside, or object is ever being inside of the region. The location relativity can be inferred by different parts of the object, including


Minimum detected object size(in pixels): 300
Maximum detected object size(in pixels): 500
Minimum number of frames the motion should last: 20

Minimum detected object size(in pixels): 300
Maximum detected object size(in pixels): 500
Alert when object Centroid Enters Region from Outside

(a) (b)

Figure 3.12 User interfaces for configuring motion detection (a) and region (b) alerts.

the object's head (topmost point), centroid, foot part (lowest point), and whole (entirety of the object). In addition, a sizing threshold can be applied to the target objects in order to trigger alerts. Figure 3.12 shows the graphical user interfaces for configuring the motion detection and region access alerts.

3.3.4 Composite Alert Detection

The above-mentioned alerts (referred to as primitive alerts) are generated based on the visual analysis of single camera views. Given the rapid improvement of digital video technologies, large-scale surveillance networks are becoming more practical and affordable. With hundreds (or even thousands) of camera inputs, cross-camera analytics are attracting increasing attention. Due to the current development in this area and the complex settings of the scenes, direct operational multi-camera event modeling still remains a very challenging problem. In addition to this, a new trend in surveillance networks nowadays is the effective integration of non-video signals, such as combining speed detection radars with video capture from a nearby camera to determine a speeding violation. To address these problems, we have developed an open infrastructure to detect composite events with multiple cameras and auxiliary sensors.

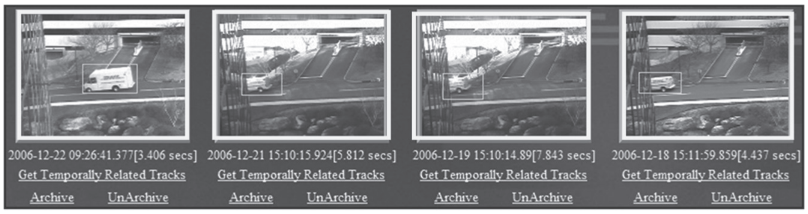
The proposed framework combines the primitive alerts to formulate high-level composite events using logical operators with applicable spatiotemporal constraints. The primitives can occur on the same camera as well as being distributed among different machines in a network. Logically, composite events are formulated as multilevel full-binary trees, where the root node represents the target composite event, the leaf nodes represent the primitive events, and the middle nodes represent the rule operators that combine the primitives. Intuitively, rules are composed of binary operators with operands that could be either primitives and/or other rules. The multilevel and distributive design enables the extensibility and scalability of the proposed system.

In our system, we incorporate four different binary operators in the rules: AND, OR, FOLLOWED BY, and WITHOUT. Other binary operators can also be easily added to the system without significantly changing its basic infrastructure. Each rule also possesses a spatial and a temporal constraint to represent the physical occurrence locations and the occurrence temporal distance, respectively. A standardized XML event language is developed for the efficient storage and transmission of the composite event and its primitives.

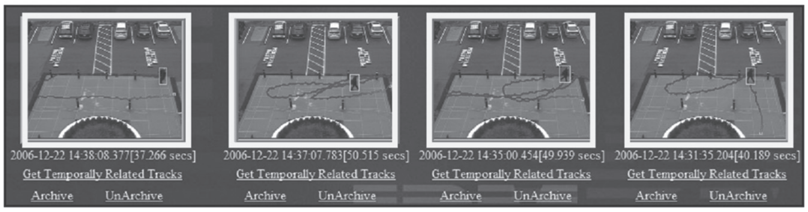
3.4 SEARCHING SURVEILLANCE VIDEOS

In addition to providing real-time alerts, IBM SSS also allows the user to search for data based on attributes extracted from analytics modules such as object type (person, vehicle), color, size, speed, human body parts, and many others. These attributes are constantly ingested as XML metadata into a DB2 database as new events are detected. In this way, our system also allows composite search by combining different visual attributes or even nonvisual data captured from multiple data sources.

Figure 3.13 shows examples of composite search results. Note that simple attribute combinations can lead to very powerful retrieval of events. If we search by [object type=car, color=yellow, size>1500 pixels] we can find all DHL trucks in the scene. Searching by [object type=person, track duration>30 s], we can find people loitering in front of a building.


2006-12-22 09:26:41.377[3.406 secs]
Get Temporally Related Tracks
Archive
UnArchive
2006-12-21 15:10:15.924[5.812 secs]
Get Temporally Related Tracks
Archive
UnArchive
2006-12-19 15:10:14.89[7.843 secs]
Get Temporally Related Tracks
Archive
UnArchive
2006-12-18 15:11:59.859[4.437 secs]
Get Temporally Related Tracks
Archive
UnArchive

(a)


2006-12-22 14:38:08.377[37.266 secs]
Get Temporally Related Tracks
Archive
UnArchive
2006-12-22 14:37:07.783[50.515 secs]
Get Temporally Related Tracks
Archive
UnArchive
2006-12-22 14:35:00.454[49.939 secs]
Get Temporally Related Tracks
Archive
UnArchive
2006-12-22 14:31:35.204[40.189 secs]
Get Temporally Related Tracks
Archive
UnArchive

(b)

Figure 3.13 (See color insert following page xxx.) Composite search in our system. (a) Finding DHL trucks based on object type, size, and color. (b) Finding people loitering based on object type and duration.

3.4.1 Large-Scale Data Indexing

A large urban surveillance system typically has the profile depicted in Figure 3.14. Hundreds of billions of events need to be archived per month, which is a very challenging task. Next, we describe our approach to handle high volumes of real-time transactions and how we minimize the transaction/inquiry response time.

3.4.1.1 Approach to High-Volume Scaling

Given the event volume in Figure 3.14, the data management layer (web application server and database) is required to support an

Estimated events from vehicle traffic in a metro city	
Number of pole mounted street cameras in a large metro city	1000–5000
Number of events per camera. **Vehicle traffic only**	50,000/day/camera
Total number of events generated by the surveillance system/day	50 M–250 M
Number of events archived per month	**150B–750B**

Figure 3.14 Typical number of events in urban environments.

ingestion rate of millions of events per day. IBM SSS accomplishes this by using a combination of web application server clustering and database partitioning. In a clustered implementation of the web application server, a central node redirects traffic from cameras to secondary nodes to balance the load. As the number of cameras in the system increases, the system can be scaled by adding new web app server nodes that run on independent hardware, thus allowing scaling. At a secondary level, a similar philosophy is used to scale the database server using database partitioning where a single logical view of the database is supported by multiple instances of the software running on independent servers.

3.4.1.2 Minimizing Transaction/Inquiry Response Time

Most transaction and inquiry response delays can be attributed to either a nonoptimized physical design or an insufficient disk storage layout for the data store. In a high volume environment, the characteristics of the metadata are highly important when designing the appropriate table and indexing mix for a highly scalable solution. We use the 20/80 rule to optimize access to the 20% of the metadata that supports 80% of the customer queries.

It is also important to utilize a disk array, which has been configured to accommodate the amount of information that is being written from the data store. The following tasks need to be reviewed and implemented prior to installation: (1) Hardware disk controller with an adequate amount of cache to easily handle the known transaction volume; (2) total size of the data store with respect to the number of physical disks in the storage array; (3) maximize the number of disk "spindles" to handle the current storage requirements; and (4) ability to add more storage space in the event that the number of cameras in scope increases.

In addition to using the above design principles, we leverage the hardware-driven scalability of the database platform (DB2). This approach has resulted in event indexing systems that can handle hundreds to thousands of cameras on high-activity urban environments.

3.5 PRACTICAL CHALLENGES: SCALABILITY ISSUES

As the need for intelligent video surveillance systems grows, with customers requiring analytics on hundreds or even thousands of cameras, scalability becomes more and more of a concern. Often, there are very strict space and cost constraints that limit the computing resources that can be deployed at a given site, so we must scale our algorithms accordingly.

We have found that the CPU consumption of our system is strongly related to the amount of activity that is taking place in the scene. We characterize activity level at a point in time as a function of the number of moving objects in the camera view and the foreground occupancy (the ratio of foreground area to background area). We provide rough definitions for three activity levels in terms of these metrics:

1. Low: Less than three moving objects AND/OR less than .05 foreground occupancy.
2. Medium: Between 3 and 10 moving objects AND/OR between .05 and .2 foreground occupancy.
3. High: Greater than 10 moving objects AND/OR greater than .2 foreground occupancy.

We purposely make the categories ambiguous in the definitions above to allow some flexibility for human judgment during the categorization process. For example, a scene may be considered to have a medium level of activity if there are three moving objects and .3 foreground occupancy by applying OR in the 2nd rule while applying AND in the 3rd rule.

In medium- and high-activity scenarios, the scene is often cluttered due to high traffic volume and crowds of people that cluster together. These situations not only create a challenging task for our object tracking, but also result in a larger computational burden. In Figure 3.15, total CPU usage is shown for 4, 8, 12, and 16 channels over a 45 min interval of a typical urban surveillance scene with activity levels that range from medium to high. The test was performed on Windows 2003 Server Service Pack 2 with Intel Xeon 2.33 GHz dual quad-core processor and 2 GB memory. Measurements were taken

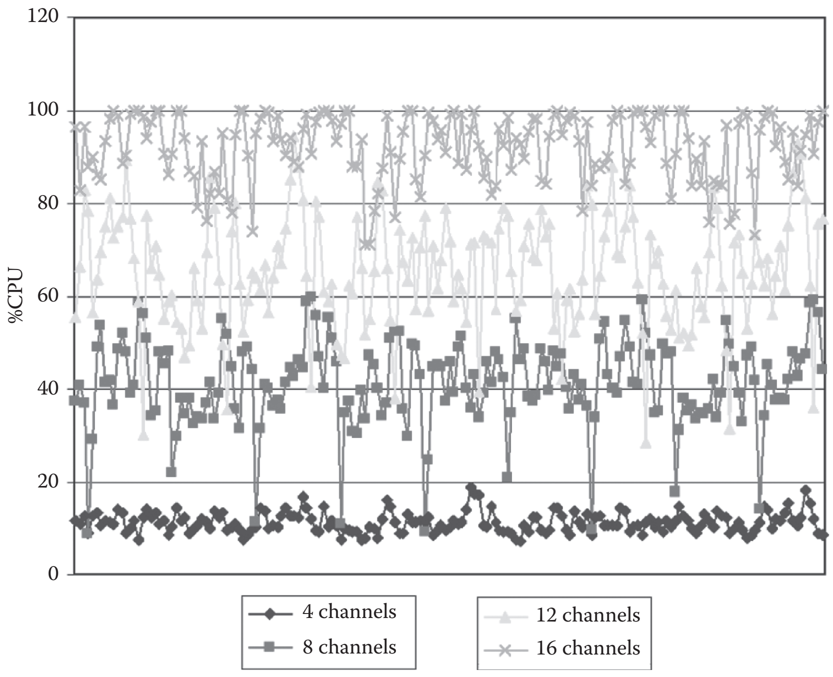


Figure 3.15 (See color insert following page xxx.) CPU load according to the number of channels in a single IBM blade.

using Windows Performance Monitor at 15 s intervals. As can be seen, CPU usage steadily rises as more channels are added, with the CPU often saturated at 16 channels. We believe the relatively large variance observed with increased numbers of channels is related to higher cache fault and page fault rates.

When the CPU becomes saturated, we begin to experience more dropped frames and decoding errors. In Figure 3.16, we show CPU usage for all 16 channels on a similar scene with frame drop rates for three of these channels. Frame drop rate is defined as

$$\text{Frame drop rate} = 1 - \frac{\text{Actual frame rate}}{\text{Requested frame rate}}$$

In this experiment, we request 15 fps from the video decoder. We have found that we often begin to experience decreases in accuracy when

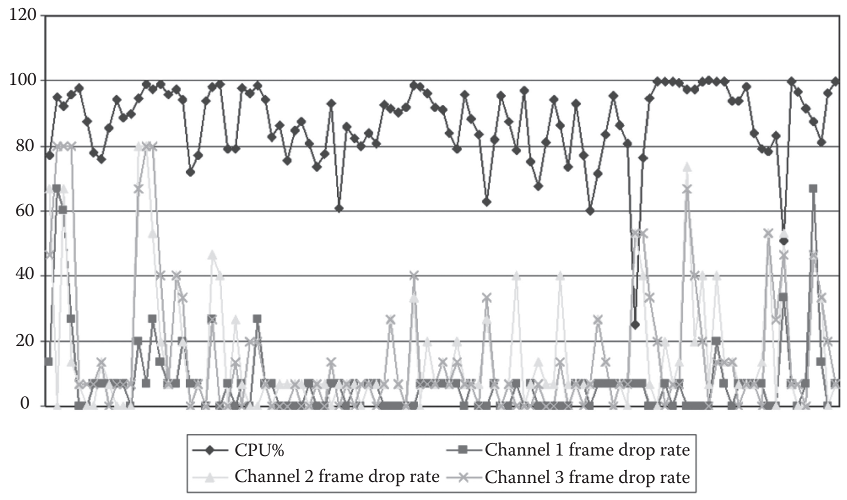


Figure 3.16 (See color insert following page xxx.) CPU load according to frame drop rate.

we fall below 10 fps, meaning that frame drop rates that are greater than .33 may result in accuracy degradations.

3.6 SUMMARY

We have presented our video analytics algorithms specifically designed to handle urban environments. Although we have already described several computer vision modules that work well under high-activity-level scenarios, our current work addresses a complete framework to maximize the efficiency and accuracy in low activity and crowded scenes.

The framework being developed has an activity level predictor module which can automatically switch the system to low or high-activity modes. In the low activity mode, we use object tracking based on detected foreground blobs by background subtraction, and higher-level analytics built on top of tracking. In the high-activity mode, we use non-tracking-based analytics which are more efficient and accurate in crowded scenes. Some of the non-tracking-based modules have been described in this chapter, but we are working on more algorithms such as appearance-based detectors for object classification, motion flow analysis, and others.

3.7 BIBLIOGRAPHICAL AND HISTORICAL REMARKS

In this chapter, we covered a broad range of video analytics for urban surveillance, as part of the IBM Smart Surveillance System. Many other commercial systems for intelligent urban surveillance exist in the market. Examples include the systems of companies such as Siemens (http://www.siemens.com), ObjectVideo (http://www.objectvideo.com), and Honeywell (http://www.honeywell.com), to mention just a few. One of the key advantages of the IBM system is its ability to index and search large amounts of video data, leveraging IBM cutting-edge technology on data storage and retrieval.

Although the main focus of this chapter was on urban surveillance, intelligent video analytics can be applied to enhance security in many other areas. As an example, IBM has developed video analytics for retail stores with the goal of loss prevention (returns fraud and cashiers fraud), store operations (customer counting), and merchandising (display effectiveness). We refer the reader to Senior et al. (2007) and Fan et al. (2009) for more information. Other application areas include fraud detection in Casinos, monitoring airports, etc.

REFERENCES

Bose, B. and Grimson, E. 2004. Improving object classification in far-field video. *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR'04*), Washington, DC.

Brown, L. 2004. View-independent vehicle/person classification. *ACM International Workshop on Video Surveillance and Sensor Networks*, New York.

Chen, L., Feris, R., Zhai, Y., Brown, L., and Hampapur, A. 2008. An integrated system for moving object classification in surveillance videos. *IEEE International Conference on Advanced Video and Signal-Based Surveillance*, Santa Fe, NM.

Dalal, N. and Triggs, B. 2005. Histograms of oriented gradients for human detection. *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR'05*), Washington, DC.

Fan, Q. et al. 2009. Recognition of repetitive sequential human activity. *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR'09*), Miami, FL.

Feris, R., Tian, Y., and Hampapur, A. 2007. Capturing people in surveillance videos. *IEEE International Workshop on Visual Surveillance*, Minneapolis, MN.

Javed, O. and Shah, M. 2002. Tracking and object classification for automated surveillance. *European Conference on Computer Vision* (*ECCV'02*), Copenhagen, Denmark.

Senior, A., Hampapur, A., Tian, Y., Brown, L., Pankanti, S., and Bolle, R. 2006. Appearance models for occlusion handling. *J. Image Vision Comput.*, 24, 11, 1233–1243.

Senior, A. et al. 2007. Video analytics for retail: The IBM smart surveillance retail solution. *IEEE International Conference on Advanced Video and Signal-Based Surveillance*, London, U.K.

Shu, C. et al. 2005. IBM smart surveillance system: An open and extensible framework for event based surveillance. *IEEE Conference on Advanced Video and Signal Based Surveillance*, Como, Italy.

Sivic, J., Everingham, M., and Zisserman, A. 2005. Person spotting: Video shot retrieval for face sets. *International Conference on Image and Video Retrieval*, Singapore.

Stauffer, C. and Grimson, E. 1999. Adaptive background mixture models for real-time Tracking. *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR'99*), Ft. Collins, CO.

Tian Y., Lu, M., and Hampapur, A. 2005. Robust and efficient foreground analysis for real-time video surveillance. *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR'95*), San Diego, CA.

Tian, Y., Feris, R., and Hampapur, A. 2008. Real-time detection of abandoned and removed objects in complex environments. *IEEE International Workshop on Visual Surveillance* (*in conjunction with ECCV'08*), Marseille, France.

Tseng, D. and Chang, C. 1994. Color segmentation using ucs perceptual attributes. *Proc. Natl. Sci. Counc.*, 18, 3, 305–314.

Viola, P. and Jones, M. 2001. Rapid object detection using a boosted cascade of simple features. *IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR'01*), Kauai, HI.

AUTHOR QUERIES

[AQ1] Please confirm permission status for all the figures.

[AQ2] Please check the phrase "…used as a prior…" for sense, in the sentence: "Note that … feature."

[AQ3] Please specify the section number.

[AQ4] This figure will not be set in color. Please repharse the caption accordingly.