

Event detection, query, and retrieval for video surveillance

Ying-li Tian, Arun Hampapur, Lisa Brown, Rogerio Feris,
Max Lu, Andrew Senior, Chiao-fe Shu, and Yun Zhai

IBM T. J. Watson Research Center,
PO Box 704, Yorktown Heights, NY 10598

Abstract

Video surveillance automation is used in two key modes: watching for known threats in real-time and searching for events of interest after the fact. Typically, real-time alerting is a localized function, e.g. an airport security center receives and reacts to a “perimeter breach alert,” while investigations often tend to encompass a large number of geographically distributed cameras like the London bombing, or Washington sniper incidents. Enabling effective event detection, query and retrieval of surveillance video for preemption and investigation involves indexing the video along multiple dimensions. This chapter presents a framework for event detection and surveillance search that includes: video parsing, indexing, query and retrieval mechanisms. It explores video parsing techniques that automatically extract index data from video, indexing which stores data in relational tables, retrieval which uses SQL queries to retrieve events of interest and the software architecture that integrates these technologies.

1. Introduction

Video analysis and video surveillance are active areas of research. The key challenges are video-based event detection and large-scale data management and retrieval. While detecting and tracking objects is a critical capability for smart surveillance, the most critical challenge in video-based surveillance (from the perspective of a human intelligence analyst) is retrieval of the analysis output to detect events of interest and identify trends. In this chapter, we describe a specific system, the IBM Smart Surveillance Solution, in order to detail an open and extensible framework for extracting events in video which can be used for real-time alerting, searching during investigations with unpredictable characteristics, or exploring normative (or anomalous) behaviors.

Current systems have begun to look into automatic event detection. These are often point solutions for detecting license plate numbers, abandoned objects, or motion in restricted locations. However, the area of context-based interpretation of the events in a monitored space is still in its infancy. Challenges here include: using knowledge of time and deployment conditions to improve video analysis, using geometric models of the environment and other object and activity models to interpret events, and using learning techniques to improve system performance and detect unusual events. The first hurdle that must be overcome is to provide extensible search capabilities based on the broadest

possible set of meaningful event metadata which can be provided by state-of-the-art point solutions.

This chapter explores these issues using as an example the IBM Smart Surveillance Solution. Its architecture is outlined as an example of a system which addresses the problems of indexing event metadata and providing extensible search. Its components provide examples of video parsing, indexing and retrieval methods which are deployed by the system. Lastly, its interface shows many examples of how an end-user may search for specific information regarding a real-world investigation.

2. Background

Video surveillance systems which run 24/7 (24 hours a day and seven days a week) create a large amount of data including videos, extracted features, alerts, statistics etc. Designing systems to manage this extensive data and make it easily accessible for query and search is a very challenging and potentially rewarding problem. However, the vast majority of research in video indexing has taken place in the field of multimedia, in particular for authored or produced video such as news or movies, and spontaneous but broadcast video such as sporting events. Efforts to apply video indexing to completely spontaneous video such as surveillance data are still emerging.

The work in video indexing of broadcast video has focused on such tasks as shot boundary detection, story segmentation and high level semantic concept extraction. The latter is based on the classification of video, audio, and text into a small (10-20) but increasing number of semantically interesting categories such as outdoor, people, building, road, vegetation, and vehicle. For broadcast video, the goal is to find a high level indexing scheme to facilitate retrieval. The task objectives are very different for surveillance video. For surveillance video, the primary interest is to learn higher level behavior patterns. In both broadcast and surveillance video, there exists a semantic gap between the feasible low level feature set and the high level semantics or ontology desired by the system users.

Because of its practical nature, surveillance video analysis has been extensively explored. However, compared to the vast amount of research in broadcast video search, such as (Hauptmann, 2006; Naphade, 2004), very few systems address the issue of search in surveillance video. Lee (2005) describes a user interface to retrieve simple surveillance events like presence of person and objects. Stringa (1998) proposed a content-based retrieval system for abandoned objects detected by a subway station surveillance system. In their system, similar abandoned objects can be retrieved using feature vectors of position, shape, compactness, etc. Berriss (2003) utilized the MPEG-7 dominant color descriptor to establish an efficient retrieval mechanism to search for the same person from surveillance systems deployed in retail stores. Meesen (2006) analyzed the instantaneous object properties in surveillance video key-frames, and performed content-based retrieval using a generic dissimilarity measure which incorporates both global and local dissimilarities between the query and target video key-frames. There is significant effort in industrial surveillance systems (ObjectVideo; PyramidVision)

targeted toward real-time event detection. Very few of these systems have focused on video search. 3VR (3VR) does provide capabilities to search for a person based on face recognition. In summary, there is a very limited number of both research and commercial systems focused on searching surveillance video. As surveillance systems grow in scale and utility, there is an increasingly critical need to provide the corollary search capabilities.

While applying video analytics to provide real-time alerting based on predetermined event definitions, such as “tripwire,” has been explored both in the research literature and in commercial systems, the challenges of searching through surveillance video remain largely unaddressed. While video analysis and pattern recognition technologies are at the core of “intelligent” or “smart” surveillance, effective search of surveillance video requires research into searchable meta-data representations for video based features, data models for indexing and correlating diverse types of meta-data, and architectures for integrating technologies into large scale systems.

Searching surveillance video essentially revolves around the following key search criteria: (1) Specific search for people and vehicles and (2) Generic search for objects and events of interest. Search applications require a combination of these criteria to create composite queries and the ability for the search to be applied across multiple cameras distributed over a spatial region.

In this chapter, we use the IBM Smart Surveillance System (SSS) as an example system for discussing various aspects of the technology involved in event detection, query and retrieval. The IBM Smart Surveillance Solution (SSS) is an IBM service offering for use in surveillance systems and provides video based behavioral analysis capabilities. It offers not only the capability to automatically monitor a scene but also the ability to manage the surveillance data, perform event based retrieval, receive real time event alerts through a standard web infrastructure and extract long term statistical patterns of activity. The IBM SSS is an open and extensible framework designed so that it can easily integrate multiple independently developed event analysis. Section 2 describes the architecture of the IBM SSS including a description of the two main components: the SSE (Smart Surveillance Engine) which takes the camera inputs and produces event metadata, and MILS (Middleware for Large Scale Surveillance) which provides data management and retrieval capabilities. Section 3 presents the underlying processes which comprise the video parsing (or analytics) performed by the SSE to create event metadata. Section 4 describes MILS in more detail including the services it provides and the data structure used. Section 5 presents the user interface of the system. Sections 6 and 7 explore the various aspects of searching for people, events and objects. Sections 8 and 9 present examples of compound queries and the concept of spatio-temporal searching. Section 10 shows some performance results for searching in the Smart Surveillance System. We conclude the chapter with a discussion of the significant research challenges that remain in enabling large scale searching of surveillance video.

3. The IBM SSS

The IBM SSS includes two components: (1) *Smart Surveillance Engine (SSE)* which provides video analysis capabilities; (2) *Middleware for Large Scale Surveillance (MILS)* which provides data management and retrieval capabilities. These two components support the following features:

- Local Real-time Surveillance Event Notification: This set of functions provides real-time alerts to the local application.
- Web-based Real-time Surveillance Event Notification: This set of functions provides a web-based real-time event notification within 3 seconds of the occurrence of a specified event in the monitored area; for example “Speeding Vehicle.”
- Web-based Surveillance Event Retrieval: This set of functions provides the ability to retrieve surveillance events based on various attributes like object type, speed, or color.
- Web-based Surveillance Event Statistics: This set of functions provides the ability to compute a variety of statistics on the event data. For example the distribution over time of arrivals and departures from a building over a day.

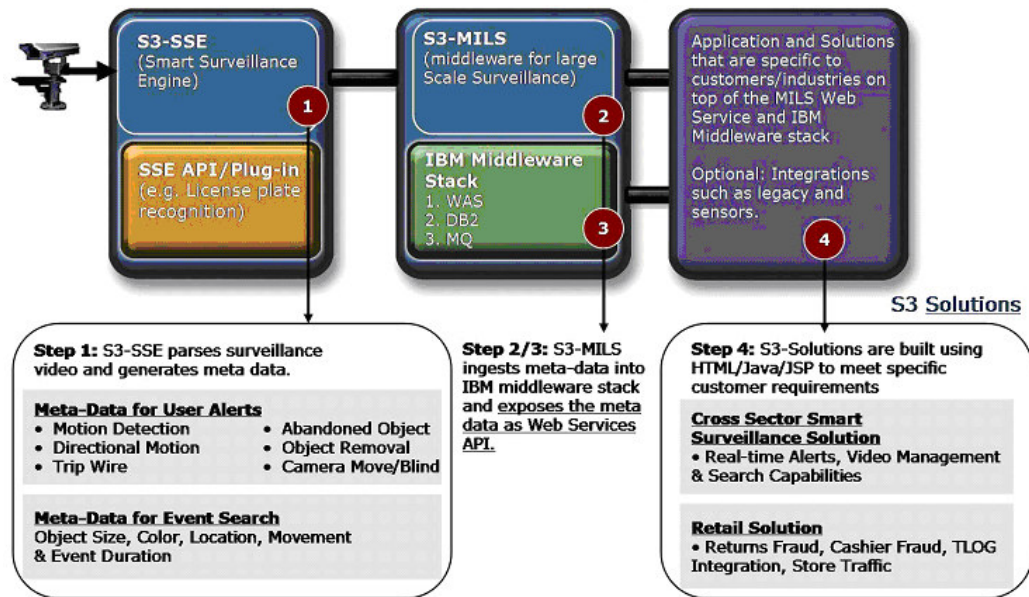


Figure 1: An open and extensible architecture for IBM SSS. The smart surveillance engine (SSE) provides a plug and play framework for video analytics. The event meta-data generated by the engines are sent to the database as XML files. Web services API's allow for easy integration and extensibility of the meta-data. Various applications like event browsing and real time alerts can use an SQL-like query language through web services interfaces to access the event meta-data from the data base.

Figure 1 shows the software architecture of IBM SSS. For details, refer to Shu (2005), which supports the above four features with the following software components:

Smart Surveillance Engine (SSE) The SSE is designed to process one stream of video in real-time, extracting object meta-data and evaluating user defined alerts. The SSE

uploads messages in XML to the central data repository. The SSE provides the software framework for hosting a wide range of video analytics like behavior analysis, face recognition, license plate recognition etc. One computer can run multiple analytics on multiple channels of video.

Middleware for Large Scale Surveillance (MILS) MILS provides the algorithms needed to take the event meta-data and map it into tables in a relational database. Additionally, MILS provides event search services, meta-data management, system management, user management and application development services. MILS uses off the shelf data management (IBM DB2), a web server (IBM Websphere Application Server) and messaging software (IBM MQ) to provide these services.

Solutions These are mainly web applications (written in HTML, Java, JSP, applets, Javascript, and AJAX) which use the web services provided by MILS to provide the functionality needed by the user to query the database and view the results.

The data flow in the IBM SSS architecture is summarized as following:

1. Sensor data from a variety of sensors is processed in the Smart Surveillance Engines (SSEs). Each SSE can generate real-time alerts and generic event meta-data.
2. The meta-data generated by the SSE is represented using XML. The XML documents have some set of fields which are required and common to all engines and others which are specific to the particular type of analysis being performed by the engine.
3. The meta-data generated by the SSEs is transferred to the backend MILS system. This is accomplished via the use of web services data ingest APIs provided by MILS.
4. The XML meta-data is received by MILS and indexed into predefined tables in the IBM DB2 database. This is accomplished using the DB2 XML extender. This allows for fast searching using the primary keys.
5. MILS provides a number of query and retrieval services based on the types of meta-data available in the database.

4. Video Parsing Performed by the SSE

In the first item in the data flow of the architecture of the SSS, the Smart Surveillance Engine (SSE) processes the sensor data (typically video from a camera) to generate real-time alerts and generic event meta-data. The basic approach used to extract alerts and events from surveillance video involves detection and tracking. The specific nature of the detection and tracking vary based on the type of video analysis technique used. For example, as a car (person) enters a camera's view, the SSE would detect the entry of the license plate (or face) and recognize and track it until the car (or person) leaves the camera field of view. In the IBM SSE, the following main steps are followed to extract important features for event detection from video and non-video information. In this chapter, we will only focus on the video-based event detection:

- Camera Stabilization
- Moving Object Detection and Tracking
- Object Classification

- Color Classification
- Alert Detection
- Compound Spatio-Temporal Event Detection
- Face Capture and Tracking
- People Counting
- Behavior Analysis

4.1 Camera Stabilization

In order to achieve robust event detection results for complex environments such as outdoor video surveillance on windy days, camera stabilization techniques (Jin, 2001) have been applied to the input video streams to correct the subtle camera motion. We use a point tracking method (tracking salient feature points from frame to frame) similar to the method used by Lucas (1981) and motion compensation to estimate the camera movement and output stabilized video for further processing.

4.2 Moving Object Detection and Tracking

The most widely applicable form of surveillance video parsing uses moving object detection and tracking. In common with most video surveillance systems, we use background subtraction (Tian, 2005) to detect changes in a video stream. Background subtraction works by maintaining a statistical model of the observed values of a pixel and modeling the variations to distinguish a change caused by a moving object from changes due to lighting changes or camera vibrations. The detected objects are tracked over their life within a single camera using a tracking system (Senior, 2006). The tracker associates multiple detections of the same object over time and constructs tracks which each represent the movement of a single object (or sometimes the coherent motion of a group of objects). Since it corresponds to a physical object, the track (which designates a time interval) is the fundamental representation in the database. For a given object, we can derive characteristics, such as the object's type, appearance and identity, which are assumed to be constant over time, although our estimates of these characteristics may be derived from accumulations of multiple observations of the object over time. The following sections discuss how the various attributes of objects can be extracted to enable searching.

4.3 Object Classification

After moving object detection and tracking have been performed, object classification is used to determine if object tracks belong to people or vehicles. We deploy a two-phase system in order to achieve classification for an arbitrary scene. In the first phase, human/vehicle recognition is attained using classical feature-based classification based on shape and motion of the detect object. Classical features include the aspect ratio, compactness (ratio of perimeter squared over area), speed and variation in speed. This phase is used to initialize view-normalization parameters by recording the observed sizes of confidently classified objects (whose real-world size is thus known or assumed) for each view. The parameters allow the second phase to perform improved classification based on normalized features, i.e, features which are scaled according to the view. The

normalization also enables absolute identification of size and speed which can be used in various ways including identifying vehicles of a given (real-world) size and searching for objects traveling at specific speeds across different locations in the image and across different viewpoints/cameras.

4.4 Color Classification

Tracked objects are also classified as one of six colors: red, yellow, green, blue, black or white. Color is computed incrementally over the life of the object. When the object first appears, a color histogram is initialized. This histogram is updated periodically if the object remains in the scene. The histogram is computed based on (1) converting RGB to HSI color space and (2) quantizing HSI space to the six colors based on user-defined parameters. These parameters include the thresholds used to determine if saturation is high enough for different bands of intensity. The ultimate dominant color of the object is determined based on ad hoc rules which take into account object type (vehicle, person) and lighting conditions. These rules are based on thresholds for each color and the balance between black and white. If the object contains large amounts of black (because of shadows or object type) then the balance between black and white can be modified. Similarly, if only a small amount of hue is necessary for it to be the dominant color of an object (as in the case of vehicles) these thresholds can be lowered accordingly.

4.5 Alert Detection

Based on the object detection and tracking outputs, eight types of basic alerts can be currently detected in our system. The parameters of these alerts can be specified on the user interface.

Motion detection: Defines an event where a specified number of moving objects, satisfying the specified parameter values, is detected in a region of interest (ROI). The parameters for this event are the ROI, the minimum and maximum sizes for the detected objects, minimum number of frames the motion should last, and the minimum number of moving objects to detect.

Directional motion: Defines an event where a moving object is detected in the specified region and in the specified direction. The parameters are the ROI, the direction of motion in that region, and the tolerance in direction angle.

Abandoned object: Defines an event where an object satisfying the desired parameters is left in the specified region. The parameters are the region of interest, the minimum and maximum detected object sizes, and the waiting time before considering the object abandoned.

Object removal: Defines an event where an object, selected by drawing a region around it, is removed. The parameters are the region drawn around the object, and the sensitivity level. The sensitivity level is the threshold used to determine if the object is removed. This threshold is based on the amount of change measured in the region.

Trip wire: Defines an event where the line drawn is crossed in the specified direction. The parameters are the line of interest, the direction of crossing, and the minimum and maximum object sizes

Region alert: This alert detects which part of the moving object enters or leaves the specified region.

Camera blind/camera moved: This primitive event detects if/when the camera is moved or blinded.

Camera motion stopped: This primitive event detects if/when a moving camera is stopped.

4.6 Compound Spatio-Temporal Event Detection

We define multiple events or activities which may occur across different times or multiple cameras based on heterogeneous meta-data as compound events. Examples include: a person leaving a building (seen from one camera) and entering a region (seen in another camera) or tailgating (one person entering using a badge entry system, followed by another not using the badge entry system). In order to provide the flexibility to specify customized events with varying complexity, and enter them to the database in a generic way, we introduce a spatio-temporal event detection system which lets the users specify multiple composite events of high-complexity, and then detects their occurrence automatically. Events can be defined on a single camera view or across multiple camera views. Semantically higher level event scenarios can be built by using building blocks which we call *primitive events* (such as the basic alerts). Primitive events are connected to each other by an operator using a user-friendly interface. Operators include: AND, OR, SEQUENCE (one event follows another), and XOR. More importantly, the newly defined composite events can be combined with each other. For example, an event may be defined as either a car OR a person in a certain region. Another example could be an event defined as a car in region 1 AND a person crossing into region 2. This layered structure makes the definition of events with ever higher complexity possible. The event definitions are written to an XML file, which is then parsed and communicated to the tracking engines running on the videos of the corresponding cameras. For example, when multiple events are combined by a *SEQUENCE* operator, a time interval can be defined among them. With the proposed system, we can not only detect “a person exiting the building,” we can also detect “a person coming from the south corridor and then exiting the building.” Later in Section 10 of this chapter, the interface and results for an example of a compound spatio-temporal event are shown.

4.7 Face Capture and Tracking

Faces are key to identifying people. Automatically recognizing people from surveillance cameras still remains a challenging problem for face recognition technologies (FRVT, 2006; Senior, 2007). The first step in achieving automatic face recognition is the indexing of video with a “presence of people” index. While face-based people detection is

valuable, in most realistic scenes, it isn't sufficient to enable people searching because people:

- people could be facing away from the camera, in which case face capture / recognition will fail
- could be entering the scene with a pose which limits the visibility of the face from the camera,

Our approach to creating a “presence of people” index uses a combination of face and person detection to ensure a very low rate of false negatives.

Our face detection method relies on extracting adapted features to encode the local geometric structures of training samples prior to learning. Local feature adaptation is carried out by a non-linear optimization method that determines feature parameters such as position, orientation and scale in order to match the geometric structure of each training sample. This non-linear optimization is similar to the Levenberg-Marquadt method which is a well-known numerical method which minimizes an objective function over a space of parameters of the function. In a second stage, Adaboost learning is applied to the pool of adaptive features in order to obtain general features, which encode common characteristics of all training face images and thus are suitable for detection. Compared to other techniques e.g., Viola (2001), our method (Feris, 2007) offers faster learning time and improved detection rate for quantitative evaluation on standard datasets).

As described in Feris (2007), after detecting a face in the field of view of a surveillance camera, we apply a correlation-based tracking algorithm to track the face in the subsequent video frames. More specifically, when a face is detected, the correlation-based tracker is triggered. For the subsequent frame, if the face detection fails, tracking is updated with the window given by the correlation tracker, i.e. the window with highest correlation to the previous window. Otherwise, if the face detector reports a window result with a close position and size to the current tracking window, then this face detection window result is used to update tracking. This mechanism is important to avoid drifting. Continuous face detection is used to re-initialize the tracker, using multiple view-based classifiers (frontal and profile) interleaved along the temporal domain in the video sequence. Each view-based classifier is based on the two stage Adaboost learning method described above – one for frontal views and another for profile views. By using two classifiers, the face detector will more robustly detect all faces regardless of pose.

4.8 People Counting

Automatic counting of people, entering or exiting a region of interest, is a very important feature for video surveillance systems. We developed an automatic and robust people counting system which can count multiple people who interact in the region of interest, by using only one camera mounted overhead. Two-level hierarchical tracking is employed. An example of hierarchical tracking can be found in Funahashi (2005). For cases not involving merges or splits, a fast blob tracking method is used. See Francois (2004) as a related example. In order to deal with interactions among people in a more thorough and reliable way, the system uses the mean-shift tracking algorithm (Comaniciu, 2000). Using the first-level blob tracker in general, and employing the mean shift tracking only in the case of merges and splits makes the system more computationally efficient. The system setup parameter can be automatically learned in a new environment from a 3

to 5 minute video with people going in or out of the target region one at a time. We tested the proposed method with video sequences which contain many interactions (such as merges/splits, shaking hands, and hugging) between people in the ROI. Most of these interactions occur right in the vicinity of the *entry/exit line*, thus successfully resolving them is essential to determine direction and perform counting accurately. The system runs at about 33fps on 320x240 images without code optimization on 2GHz Pentium machines. Average accuracy rates of 98.5% and 95% are achieved on videos with normal traffic flow and videos with many cases of merges and splits, respectively. More details of the algorithm can be found in paper by Velipasalar (2006a).

4.9 Behavior Analysis

In IBM SSS, we have a preliminary structure for detecting trajectory anomalies. This system shown in Figure 2 analyses the paths of tracked objects, learns a set of repeated patterns that occur frequently, and detects when an object moves in a way inconsistent with these normal patterns.

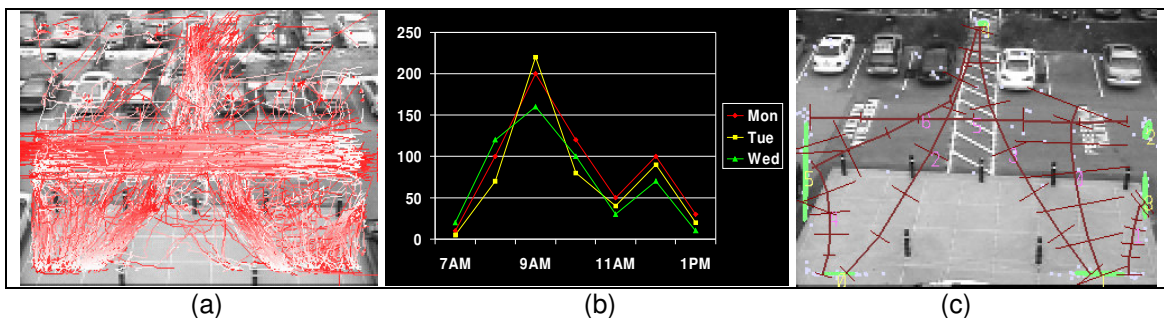


Figure 2 (a): Summary view showing the retrieval of trajectories all events that occurred in the parking lot over a 24 hour period. Trajectory color coding, start white and end is red. (b): Activity distribution over extended time period, x-axis is time, y-axis is the number of people in the area. Each day of the week is shown with a different line. (c): Unsupervised behavior analysis. Object entrance/departure zones (green ellipses) and prototypical tracks (brown curves) with typical variation (crossbars).

The system begins by detecting object entrance and exit locations. Here the start and end points of tracks are clustered to find regions where tracks often end or begin. These points will tend to be where paths or roads reach the edge of the camera’s field of view. Having clustered these locations, we have a simple classification for trajectories by labeling a track with its start and end location (or as an anomaly when it starts or ends in an unusual location such as a person walking through the bushes). For example, when we cluster trajectories for our camera which views the entrance to our building, trajectories are classified into one of 5 classes – entering/exiting into the left side (from the road on the left or from the center), enter/exiting to the right side (from the road on the right or from the center), or moving horizontally across the road. We then apply a secondary clustering scheme to further detect anomalous behavior. This scheme operates as follows: the trajectories of all tracks with a given start/end location labeling are resampled and clustered together. This gives an average or “prototypical track” together with standard deviations, as shown in Figure 3. Thus most tracks from a given entry location to a given exit will lie close to the prototypical track, with typical normal

variation indicated by the length of the crossbars. Tracks that wander outside this normal area can be labeled as anomalous and may warrant further investigation. Principal components of the cluster can also indicate typical modes of variation or “eigentracks” giving a more accurate model of normal vs. abnormal.

5. The IBM Middleware for Large Scale Surveillance

In the previous section, we described the components of the IBM Smart Surveillance Engine (SSE) which extracts event metadata from the camera input. In this section, we describe the other major component of the IBM Smart Surveillance Solution (SSS), the IBM Middleware for Large Scale Surveillance or MILS. We first describe in 5.1 the services provided by MILS which comprise the MILS Application Programming Interface. In subsection 5.2 we describe the data structures used within MILS to store the information used to index the data and perform relevant searches.

5.1 Services Provided by the MILS

MILS provides the data management services needed to build a large scale smart surveillance application and to enable extensive search capabilities. While MILS builds on the extensive capabilities of the IBM DB2 database system, it is essentially independent of this product and can be implemented on top of 3rd party relational databases. It supports the indexing and retrieval of spatio-temporal event data. MILS also provides analysis engines with the following support functionalities via standard web services interfaces using XML documents.

A: Meta-data Ingestion Services: These are web services calls which allow an engine to ingest events into the MILS system. There are two categories of ingestion services

A.1: Index Ingestion Services

A.2: Event Ingestion Services

B: Schema Management Services: These are web services which allow a developer to manage their own meta-data schemata. A developer can create a new schema or extend the base MILS schema to accommodate the metadata produced by their analytical engine.

C: System Management Services: These services provide a number of facilities needed to manage a surveillance system including

C.1: Camera Management Services

C.2: Engine Management Services

C.3: User Management Services

C.4: Content Based Search Services

5.2 Data Structures in MILS

The MILS system has three types of data structures, namely, (1) the system data structure which captures the specification of a given monitoring system, including details like geographic location of the system, number of cameras, physical layout of the monitored space, etc. (2) the user data structure which contains user names, privileges and user functionality, (3) the event data structure which contains the events that occur in a

specific sensor or zone in the monitored space. Each of these data structures is briefly described in the following subsections.

A) System Data Structure

The system data structure has a number of components, listed below.

A.1: Sensor/Camera Data Structure

A.2: Engine Data Structures

B) User Data Structure

The user data structure captures the privileges of a given user. These include

- selective access to camera views
- selective access to camera / engine configuration and system management functionality
- selective access to search and query functions.

C) Event Data Structure

This data structure represents the events that occur within a space that may be monitored by one or more cameras or other sensors. IBM SSS uses the timeline data structure which uses time as a primary synchronization mechanism for events that occur in the real world between sensors. The basic MILS schema allows multiple layers of annotations for a given time span. The following is a description of the schema:

- Event: An event is defined as an interval of time.
- StartTime: Time at which the event starts.
- Duration: This is the duration of the event. Events with zero duration are permitted, for example snapping a picture or swiping a badge through a reader.
- Event ID: This is a unique number which identifies a specific event.
- Event Type: This is a event type identifier.
- Other descriptors: Every analysis engine can generate its own set of tags such as basic types or more complex types. If the tags are basic types CHAR, INT, FLOAT, they can be searched using the native search capabilities of the database. However, if the tag is a special type (for example color histogram) the developer needs to supply a mechanism for searching the field.

The most fundamental index into surveillance video is the time of occurrence of an event. The challenge is to automatically derive the time of occurrence of “events of interest” by analyzing the video. Once an event is detected in video, the time interval of the event can be annotated with additional meta-data which captures a more detailed description of the event. Hence, the most basic data structure for surveillance events is a time interval. Table 1 below shows the basic data model for two types of surveillance events (1) a car driving through a parking lot captured on camera 23 and (2) the license plate of a car recognized on camera 35

Example Data Models	
Behavior Meta -data	License Plate Meta -data
Camera ID: 23	Camera ID: 35
Unique Event ID: 2379406	Unique Event ID: 4926402
Start: 9/10/06:02:22:15:100	Start: 9/10/06:02:12:15:100
End: 9/10/06:02:22:55:300	End: 9/10/06:02:12:25:453
Keyframe : 23567.jpg	Keyframe : 563783.jpg
Video : //mils/xx/file1.wmv	Video : //mils/xx/file3.wmv
Object Type: Car	License Plate #: 525sds
Additional Fields: (trajectory, color, shape, size, etc)	Additional Fields: (e.g State of Origin)

Table 1. Event time is used as the basis for annotation surveillance events

Each unique event that occurs within a scene is assigned an event identifier which is guaranteed to be unique across all cameras that are being indexed into a single database instance. The event ID is used as the primary key to select from and join across multiple tables in the database. The time of occurrence of the event is used to correlate events across multiple cameras that exist in the system. This data structure can easily be extended to accommodate new types of meta-data as new types of video analytics are added to the system. If the meta-data is one of the basic types (INT, CHAR, FLOAT etc.) supported by the database it can be searched using SQL. For special types of meta-data, like color histograms additional user defined search functions have to be developed.

6. Interface of IBM SSS

Figure 3 – 8 show some screen shots of the IBM SSS interface for the list of camera views, and the results of searches for car, person, face capture, license plate, and object color, as well as a summarized view of a day's traffic. In all the figures, the upper left region contains a map of the facility showing the locations of the cameras. The upper right region contains instant alerts. Alerts are updated in real-time as they occur. The lower left region contains a video player. Initially it contains a live video of the currently selected camera. But this player can also show a selected alert or event. The lower right region changes as the user selects what he/she would like to search. This region can contain either the page to specify the search criteria or the results of a search. In the figures, the lower right region differs depending on the search criteria.

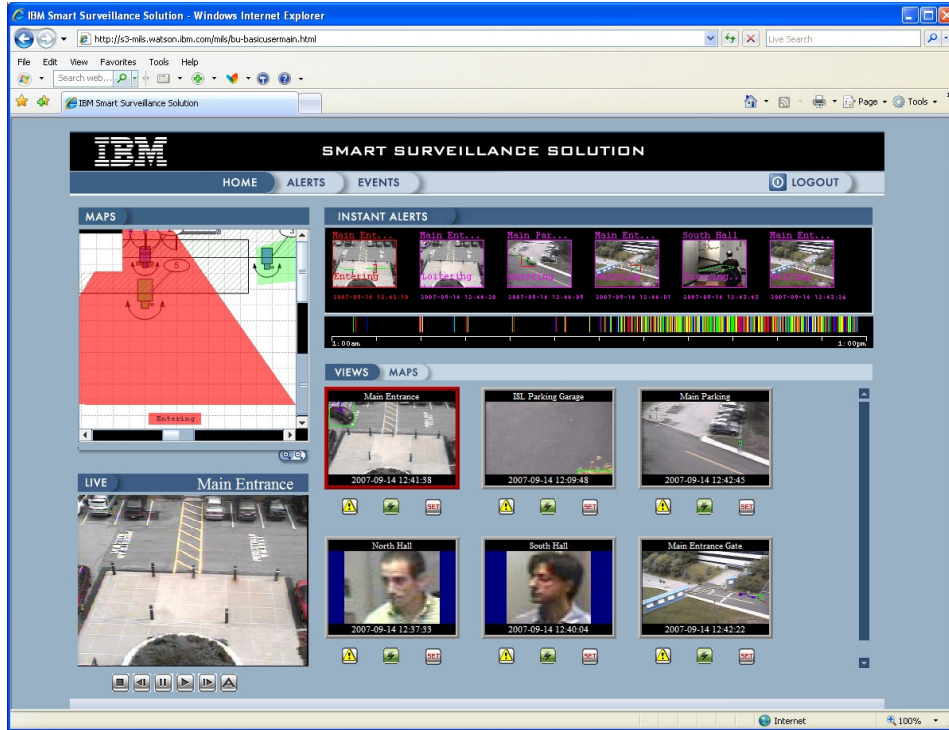


Figure 3: An Interface showing the various camera views currently available in the system



Figure 4: An Interface showing the Results from a "Find Person" Query

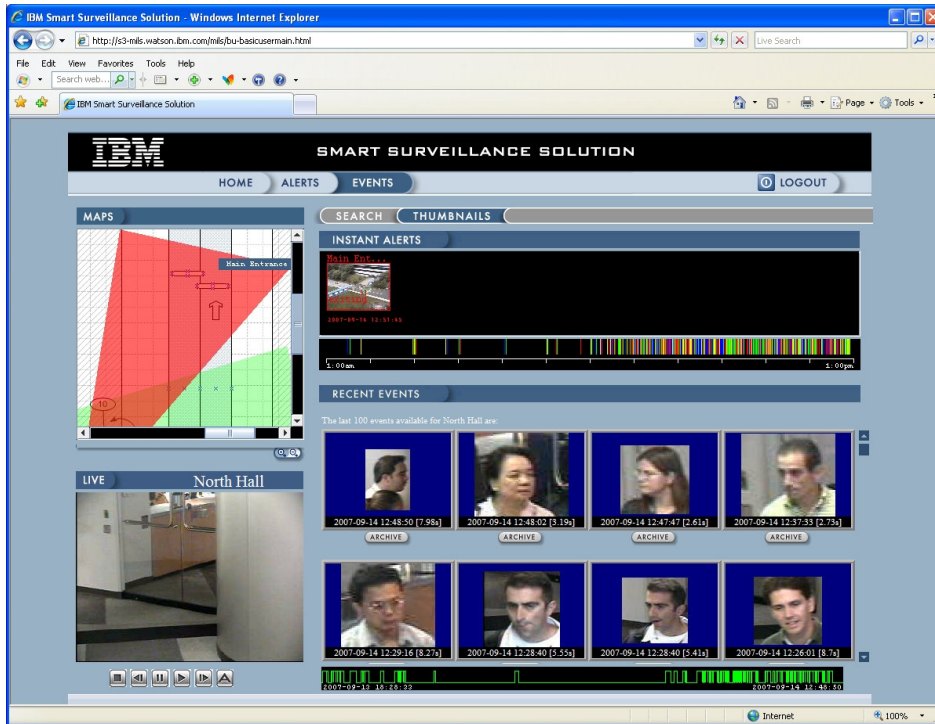


Figure 5: An Interface showing the Results of "Find Faces"



Figure 6: An Interface showing the Results of "License Plate Recognition"

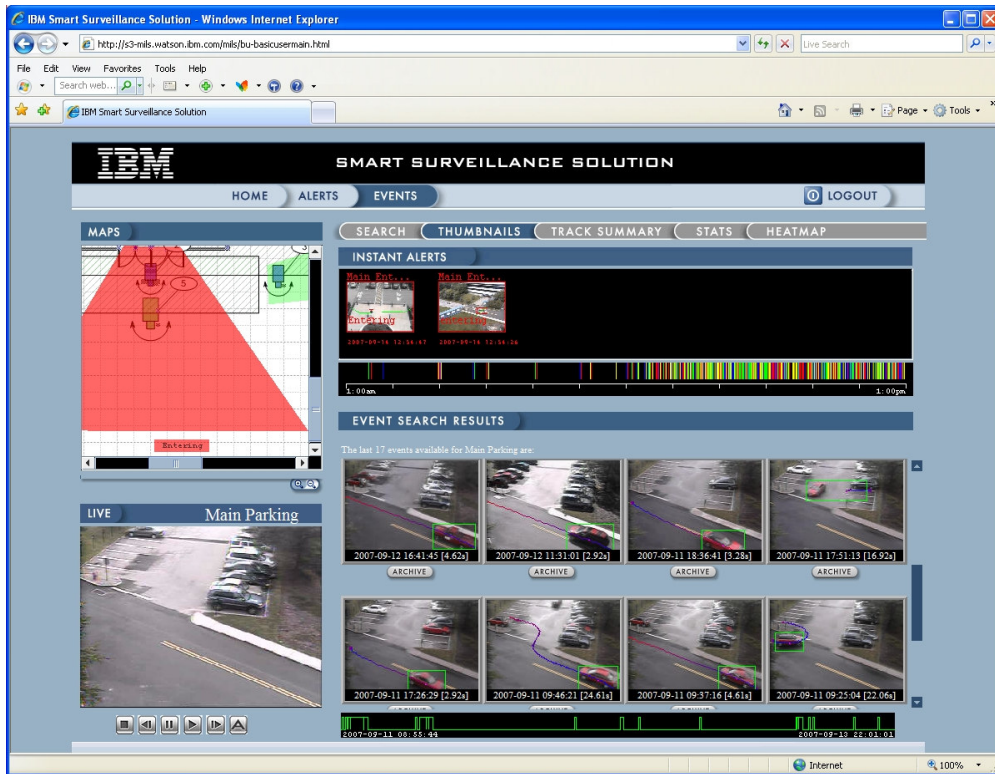


Figure 7: An Interface showing the Results of “Red Car” search

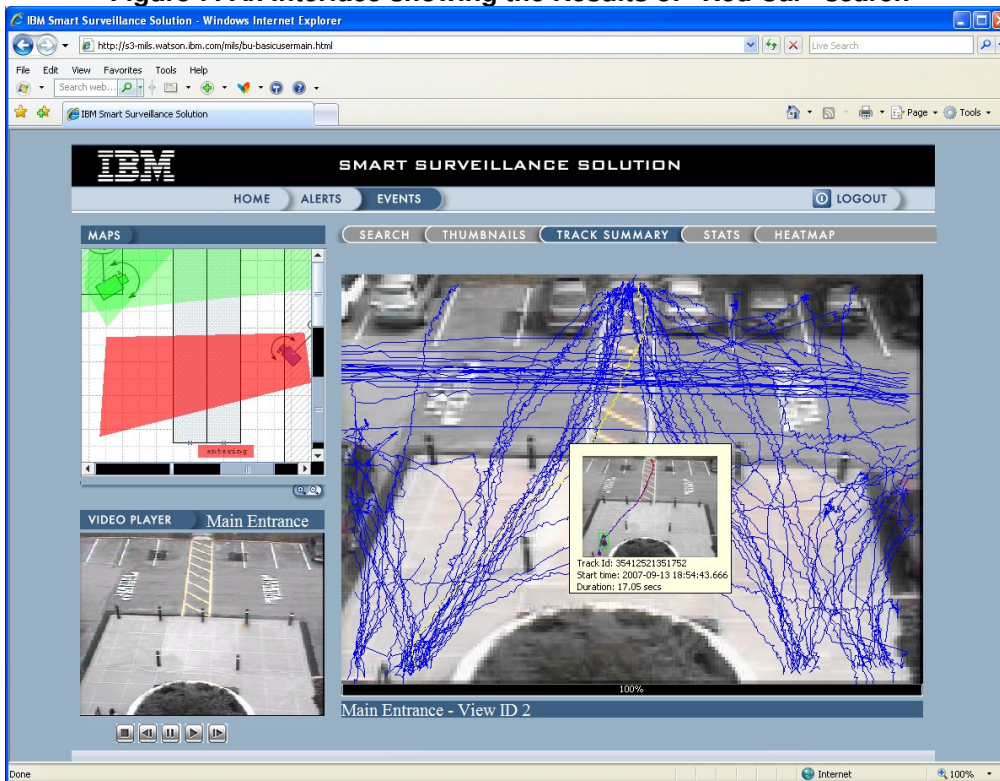


Figure 8: An Interface showing the track summary of one day data

7. Specific People and Vehicle Search

As shown in the user interface, the SSS can be used to search for events based on a large set of attributes provided by the engines (SSEs) and stored in the database and retrieval system (MILS). In this section and the next three sections, we detail the search functionality and search performance of the system. In this section, we describe the search capabilities for finding people and vehicles and determining the number of people crossing through a region. In Section 8, we describe the generic search capabilities including object color, object class, object size, object shape, object location, object movement, time of event of occurrence and event duration. In Section 9 we give an example of compound search while in Section 10 we give examples of more complex searches which we call compound spatio-temporal search. Finally in Section 11 we give some results evaluating the performance of the system in both precision and recall and for time to recall for detecting and tracking objects and executing specific search queries.

7.1 Searching for People

After detecting and tracking human faces, we also store a keyframe for each captured face image in the database, associated with a timestamp. This allows the user to issue queries like "Show me all people who entered the facility yesterday from 1pm to 5pm." An example of this search is shown at the right of Table 2.

Ideally, for every person passing through the scene, a face keyframe would be generated and stored in the database. However, due to false negatives in face detection and face pose and person orientation issues, important events might be missed. We address this problem by using a keyframe selection technique that combines a face classifier with a person classifier. If a face is detected and tracked in the video sequence, a face keyframe is stored in the database. Otherwise, a person keyframe is generated if a person is detected and tracked in the video.

We analyzed ten hours of data from one camera, taking video of the busiest hour from each of ten days. Table 2 shows our results. Out of 445 people entering the facility (not walking away from the camera), we captured 351 faces, with only 7 false positives. The reason that some faces were missed is that sometimes people enter the door looking down, occluding the face from the camera, which is placed on the ceiling. By running our keyframe selection technique (using face and person detectors), we captured all remaining 94 persons, as well as 40 persons walking away from the camera, with an additional 19 false positives.


Total # of people approaching camera		445	
Total # of people receding from camera		40	
Face Detection	Faces Captured	351	
	Faces Missed	94	
	False Poitives	7	
Person Detection	Persons captured	134	
	Approaching	94	
	Receding	40	
	False Positives	19	
Overall People False Negatives		0	
Overall People False Positives 26/445		5.6%	

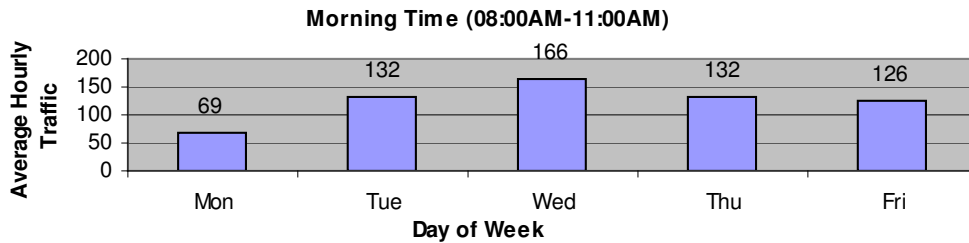
Table 2: Results obtained from ten hours of surveillance video. Example faces (frontal and profile) captured by our system (blurred to preserve privacy)

7.2. Searching for Vehicles

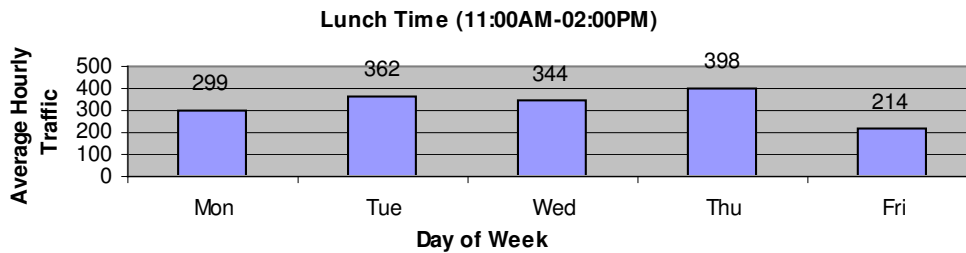
Searching for vehicles based on license plates is achieved using license plate recognition technology, which is more reliable than face recognition. An example system used in the IBM SSS was developed by Hi Tech Solution (HiTech, website). Unlike human faces, license plates vary widely based on geography. Variations include language, font, background and numbering scheme. Typically, there is no single algorithm or company which can recognize license plates across wide geographies. One approach to handling this variation is to standardize the interfaces to the license plate algorithms (such as Hi Tech's SeeCar algorithm) and standardize the meta-data representation for the license plate. The software architecture of IBM SSS supports this approach.

7.3 People Count

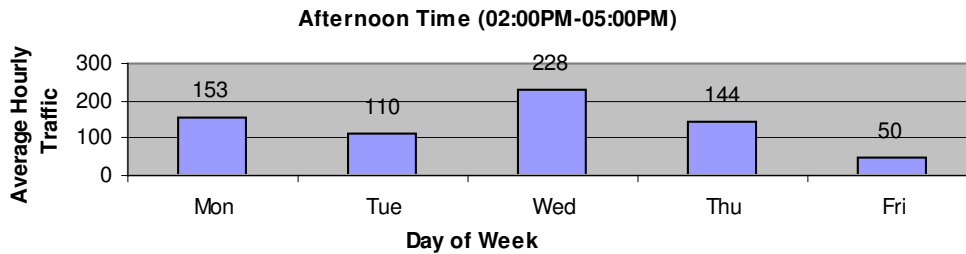
Figure 9 shows the result of one-week long test in IBM Hawthorne cafeteria, we found: 1) the morning time (8:00am-11:00am) has the lowest traffic load in a day; 2) the lunch time (11:00am-2:00pm, especially 12:00pm-2:00pm) has the highest traffic load, (as expected for a cafeteria); 3) Friday has less traffic than other week days.



(a)



(b)



(c)

Figure 9: Results of people counting for a week in the IBM Hawthorne cafeteria

8. Generic Search Criteria

Generic search includes search for objects and behaviors of objects over time. This search can be qualified by one or more of the following: object color, object class, object size, object shape, object location, object movement, time of event of occurrence and event duration.

8.1 Search by Object Color

Object color is determined by (1) converting RGB object colors to a 6 color Hue/Saturation/Intensity (HSI) space, (2) periodically updating and normalizing the 6 color HSI cumulative histogram over the life of the object and (3) determining the three dominant colors and their percentages. For vehicle color estimation, the final primary

color is determined based on hue if sufficient (regardless of the amount of achromatic pixels), and the relative amount of black and white. Table 3 shows the results of color classification for vehicles entering and exiting our facilities for a total of 8 hours (4 hours for two days). The overall correct color classification is 80%. Over half the misclassified vehicles are black or white vehicles misclassified as white or black respectively. Although this may be improved with parameter tuning taking into consideration the variations in lighting conditions, the most significant issue here is due to the variable amount of shadows included in the object segmentation and the percent of the true black components for each vehicle (i.e. windshield size, tires, accessories etc.) Figure 10 and 11 show illustrative examples of vehicles classified correctly and incorrectly.

		COLOR SEARCH □ GROUND TRUTH						
COLOR SEARCH □ RESULTS		BL	WH	RE	YE	BU	GR	
	BL	119	36	20	0	1	0	165
	WH	3	102	1	0	1	0	107
	RE	0	0	18	0	0	0	18
	YE	0	0	0	1	0	0	1
	BU	0	0	0	0	7	0	7
	GR	0	0	0	0	0	2	2
		122	138	39	1	9	2	

Table 3 Color Results: BL-Black, WH-White, RE-Red, YE-Yellow, BU-Blue, GR-Green

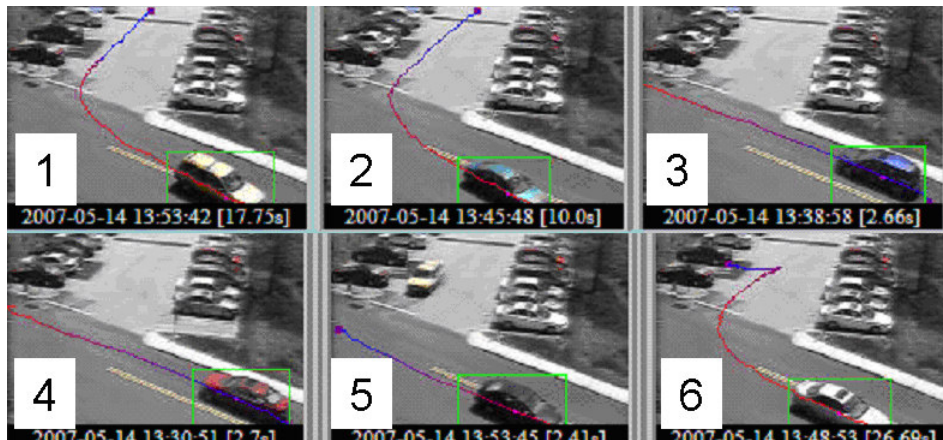


Figure 10. Retrieved keyframes (cross indexed to video by time) (1) yellow, (2) green, (3) blue, (4) red, (5) black and (6) white vehicles. (Trajectory color indicates direction of movement, blue is track start, red is track end).

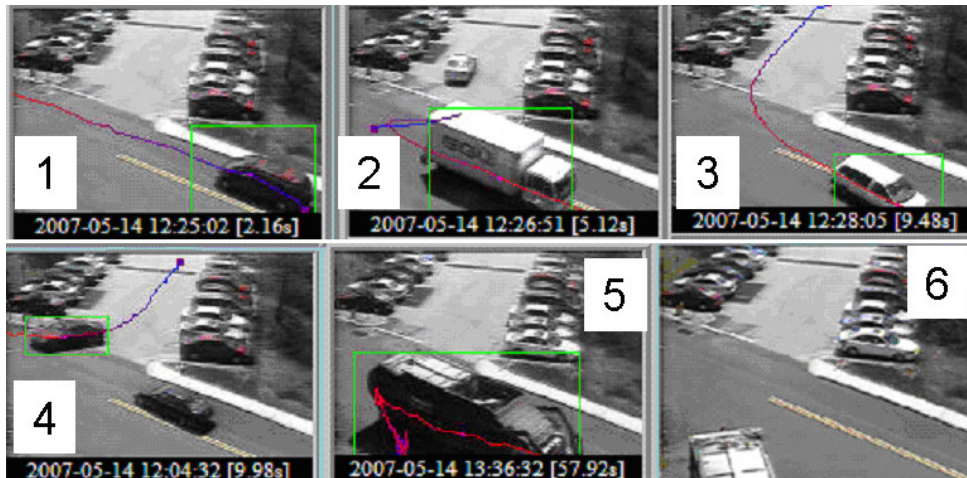


Figure 11. Keyframes 1,2,3 show errors from searching for black objects. Keyframes 4,5,6 are results of searching for white objects. For 1,2,3 notice the dark shadows and color of windows etc which lead to misclassification. In the keyframe (5), the garbage truck appears to be black on examining the original video playing , it is seen that the truck's lower body is white.

8.2 Search by Object Class

Object classification is performed using a view invariant classifier (Brown 2004). An object can be classified as either a person or a vehicle based on shape features such as compactness and principal axis ratio, and motion features such as speed and degree of recurrent motion. Table 4 (left) shows results for vehicles and people entering the front of our laboratory for 4 hours one morning. (May 16 2007, Camera #2, between 8am and 12pm). Overall 307/334 or 92% of the vehicle/person object tracks were correctly classified.

		Object Class Ground Truth		
		V	P	O
OBJECT CLASS RESULTS	V	77	0	1
	P	9	230	19
	O	1	18	53
		86	248	73

		SIZE SEARCH □ GROUND TRUTH				
		P	C	MS-T	L-T	O
SIZE SEARCH □ RESULTS	P	17	2			
	C	3	39			20
	MS-T		3	1		
	L-T				1	
	O					11
	Total	20	43	1	1	31

Table 4 Left: Object Classification Result: V: Vehicles, P: Person, O: Other Right: Search using object size. P: Person, C: Car, MS-T: Medium Sized Truck, L-T: Large Truck, O: Other.

8.3 Search by Object Size

Object size is often useful to determine object sub-class for objects moving orthogonal to the camera viewpoint. Object size was used to distinguish pedestrians from vehicles and to distinguish standard vehicles (cars, SUVs, minivans) from mid-size vehicles (delivery trucks, large pickups) and large trucks (such as garbage trucks and tractor trailers) for our

camera looking orthogonal to the entry road. Table 4 (right) shows the results of a size search used for object classification.

8.4 Search by Object Shape

Currently our system does not support explicit search by shape. However as described in the object classification section, shape of objects is used to determine the class.

8.5 Search by Object Movement

Object movement can be qualified by several parameters such as speed, acceleration, direction, and extended properties of the objects trajectory (like finding all people walking in a zigzag manner through the parking lot). The SSE computes several of these parameters for use in evaluating user specified events like directional motion of the object. At this time, our search interface only provides the ability to search based on the speed of the object.

8.6 Search by Object Location

This is achieved by storing the entire trajectory of the object into the database. The tracker (described in 4.2) generates a trajectory for each moving object in the scene in image coordinates. When the user selects a region of interest (ROI) within an image (yellow box), this is used to generate an SQL query which retrieves all the objects whose trajectories intersect the ROI. Figure 12 (left) shows the results of events recovered when the user selects the yellow region outlined in the image.

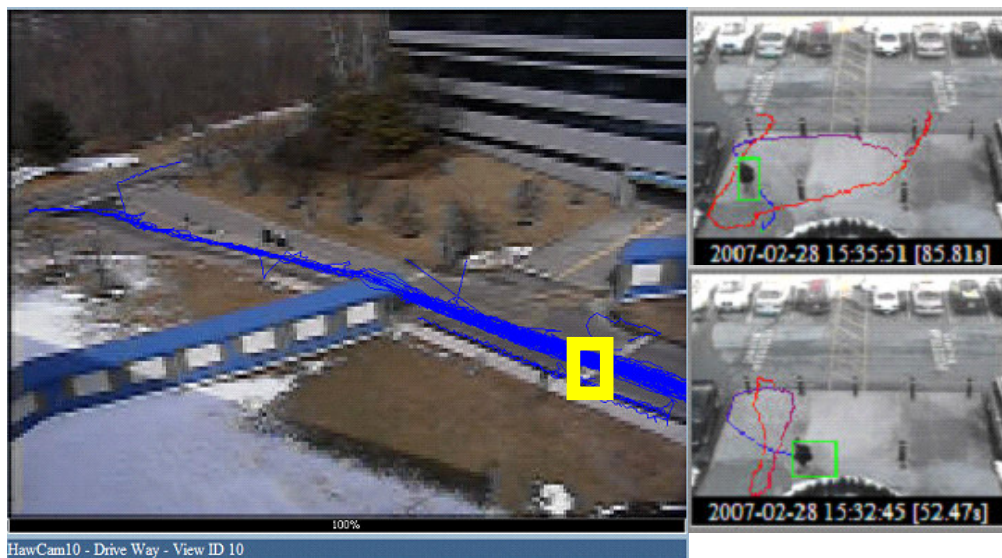


Figure 12. (Left) Results of spatial search, showing the trajectories of all objects that passed through the user-selected yellow region. The user can click on the trajectory to view the corresponding video clip. (Right) Loitering events (note the long person trajectories) retrieved by using the event duration query.

8.7 Search by Time of Occurrence of Events

Every event indexed into the database is required to have an event start timestamp and event end timestamp (see section on data structure). These time stamps are used to

retrieve events within the user specified time of interest. Currently we only support retrieval of events that occurred: a) before a user specified time b) after a user specified time c) during a user specified interval.

8.8 Search by Duration of Event

Every event recorded by the system has an associated time duration. The duration of an event can be used for multiple purposes. Below are sample events from a query for events of duration longer than 50 seconds. These sample events demonstrate how loitering can be detected by using the event duration query (figure 12 right).

9. Compound Search

All the criteria discussed above can be combined into a single query to search for events of greater complexity. Consider the following scenario: Employees at a facility have registered a complaint that one of the drivers from an express mail company is driving very fast in the parking lot. Since it is known that the delivery truck is yellow, we can use the composite query as follows:

FIND ALL, Object Type = "VEHICLE", Object Size > X1 and Object Size < X2, Object Color = Yellow, Object Speed > S1

Applying such a query to events over a month would help establish a pattern of speeding violations committed by the delivery truck, thus narrowing down the specific driver.

10. Compound Spatio-Temporal Search

In a number of applications, the events of interest are a combination of basic events over space (cameras) and time. Figure 13 shows a detected tailgate event at the entrance by using the spatio-temporal event detection method. First, the tailgate event is described by using the building blocks and operators shown in Fig. 13(a). The three primitives here correspond to the opening of the gate, detection of two cars in the ROI (Region of Interest) after the gate, and the closing of the gate respectively. The middle primitive event in Fig. 13(a) is defined so that it will be detected when there are two objects in the ROI. As can be seen in Figures 13(b) and 13(c), the second and third primitives are detected successfully. The opening of the gate cannot be detected due to weather conditions affecting the performance of the background subtraction algorithm. (This refers to subsection 4.2 on Moving Object Detection and Tracking.)Then, the description of the scenario was modified as shown in Figure 13(d) where the first primitive is changed so that it can detect a vehicle right at the gate. In this case, the first primitive can be detected successfully as well.

Many complex events, such as a person entering the building and then removing an object or a person jumping over a fence and entering a specified region, can be expressed in terms of primitive events and detected by using the proposed system. We also tested our system successfully with several scenarios like people tailgating to enter the building and a truck following an unusual path defined on the views of four cameras. These events

were defined and introduced to the system by people with no technical expertise by using the proposed scheme and the interface.

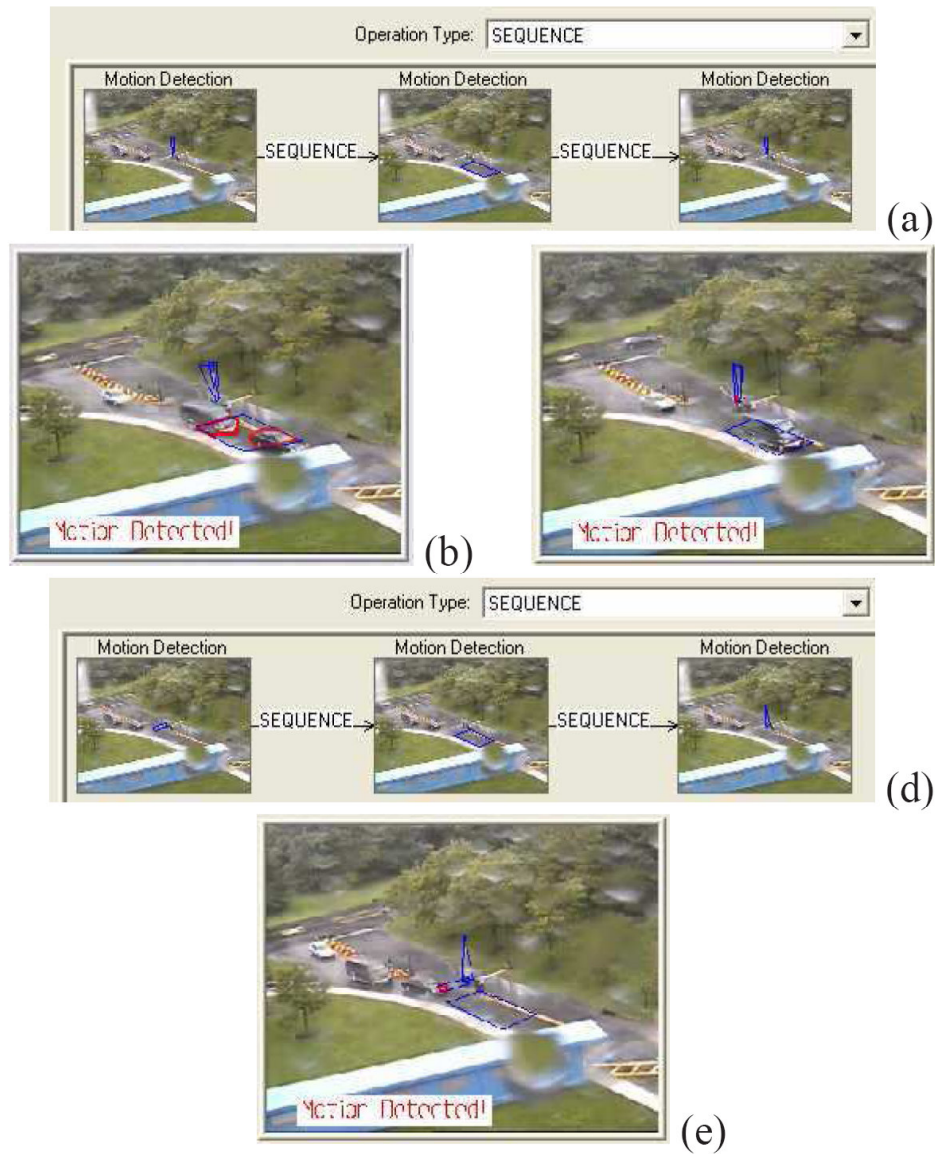


Figure 13. Detecting a “tailgate” event.

11. Search Performance

The performance of a search system can be characterized along two dimensions: precision and recall. These dimensions provide a measure of how well the system is meeting the requirements of the user’s query. The precision and recall of the overall system is a function of the precision and recall of each of the individual video parsing mechanisms (face, license plate, etc). The previous sections have presented the results for people detection. The precision and recall of all of the other retrieval techniques such as color, size, location, event time, and event duration are dependent on the precision and recall of the underlying event parsing system (object detection and tracking). A detailed

evaluation of event parsing (detection and tracking) can be found in (Brown 2005a, 2005b)

Table 5
Test Set Description

Test Data Set Description with ground truth	
Data volume	4 cameras, 10 sequences, 36 minutes (2267 frames)
Total # of hand marked objects	2964 objects in 2267 frames
Total # of objects tracks FOV	90 objects

Table 6
Object Detection Performance Summary

Object Detection or Background Subtraction Results	
False Positives	0.03 objects per frame
False Negatives (missed object)	628 of 2964 = 21.2%
Avg size of missed object	226 sq-pixels

As shown in Table 5, we used a test set of videos which was hand marked by a person for objects in each frame and tracks over the sequence. The results of running our base object detection and tracking algorithms are shown in Tables 6 and 7. At the selected operating point (set of thresholds of object size, sensitivity of detection thresholds, track match thresholds, etc.) the base performance of the detection and tracking algorithms is good on objects that are of significant size (above 169 sq. pixels). The false positives when measured at a track level tend to be very short lived trajectories (77 frames, less than 3s). Typically, events that occur in the real world are of significantly longer duration. While improvement in the base event parsing is always desirable, the current level of performance is more that adequate for a search and real-time alert in retail and city surveillance.

Table 7
Object Tracking Performance Summary

Object Tracking Results	
False Positives (spurious tracks)	25 with average length of 77 frames
False Negatives (missed tracks)	24/90 = 26.6%
Avg size of missed tracks	169 sq-pixels

Table 8
Retrieval Time Summary

Database Server, Dual Xeon, 3.8Ghz with 4GB Ram running IBM DB2	
Total number of events on Main Parking Lot Camera	From Apr 30, 07 to May 14, 07 10997 events over 15 days
Red car search	219 events retrieved in under 5secs

Table 8 shows a summary of retrieval time for the system. This is the time between the user launching a query and the system responding with results. This time varies widely based on the type of query, with location searches being the most expensive and searches based on native SQL types falling into a different bucket. Below is a sample performance result for color retrieval, which is a native SQL query.

12. Conclusions

Enabling effective search of surveillance video is a challenging problem, as it involves not only the challenges of extracting events and activities in video, but also the challenges of generating searchable meta-data, efficient indexing into a database and

intuitive search and visualization mechanisms. The current activities in research and industry have only begun to scratch the surface of the challenges involved in surveillance video.

This chapter presented a framework for addressing detection, query and retrieval issues in surveillance video using the IBM Smart Surveillance System. This system has a broad range of detection capabilities which can be used to automatically monitor a scene in real-time including person/vehicle recognition, color identification, complex alert detection, face capture, and people counting. This framework can also manage the unwieldy amount of surveillance data, perform event-based retrieval and receive real time event alerts through a standard web infrastructure. This latter capability enables large distributive systems which can scale to a large number of cameras and facilities. The system can also extract long term statistical patterns of activity to facilitate traffic monitoring and improved understanding of operational conditions. Lastly, the system is an open and extensible framework which can easily integrate multiple independently developed event analysis technologies in a common infrastructure.

Many challenges and research opportunities remain open in the space of surveillance video analysis and search. Examples include dealing with the challenges of searching for color across varying scene conditions such as time of day, camera settings, lighting etc., searching for people who have been seen earlier (in different cameras, on different days, under different lighting conditions) or dealing efficiently with indexing large amounts of video and providing intuitive interfaces for enabling search and interaction. One “Grand Challenge” is to reduce the time to investigate situations like the London bombings or the Washington sniper incident and find the perpetrators in a timely fashion.

13. Future Research Directions

We are planning to investigate *on-line learning* techniques to improve the performance of our algorithms in specific scenarios. Conventional offline methods are designed for generic scenarios, often involving large training sets with samples drawn independently from some probabilistic distribution. Offline training can be very computationally expensive, like the adaboost learning process for face detection, which can take order of weeks to be completed in conventional machines. In contrast, on-line learning methods use one example at a time to update the learning parameters and thus are more suitable to process large amounts of data. A key advantage of these techniques is *adaptation* to new environments. Consider as an example a face detector which is deployed in a particular camera. With online learning, the detector would continuously tune its parameters to adapt to the particular camera conditions (like lighting, background, etc.) as new data arrives.

Currently, our visual object tracker is restricted to a single camera. We plan to extend our approach to track objects across multiple cameras. This is a very challenging problem, as objects can change their appearance dramatically from one camera view to another, due to different camera viewpoints and camera intrinsic properties like different color responses. Obtaining reliable solutions for this problem will allow us to better monitor

sites such as retail stores, where we may desire to analyze the complete trajectories of people across multiple cameras.

We also plan to incorporate other computer vision modules in our system, such as face analysis for gender and age classification, activity recognition (e.g., detection of a person falling down), more sophisticated trajectory analysis and clustering, tracking and object classification in crowded environments, and many others. All these new modules can be easily integrated in our system framework as DLL plug-ins.

Thus far our system relies on static cameras, which allows us to use background modeling techniques to segment moving objects. In many cases, however, this segmentation may be very poor due to crowded scenarios or extreme lighting changes. A more interesting issue happens when we consider *moving cameras*, rather than static cameras. In this case, background modeling is meaningless. We are currently investigating analytics modules that work well in these circumstances.

14. References

3VR, <http://www.3vr.com/Products/#smartsearch>.

Berriss, W.P., Price, W.G., & Bober, M.Z. (2003) Real-Time Visual Analysis and Search Algorithms for Intelligent Video Surveillance, *International Conference on Visual Information Engineering* (pp. 226-229) July 2003.

Brown, L.M. (2004) View Independent Vehicle/Person Classification, *ACM 2nd International Workshop on Video Surveillance and Sensor Networks* (pp. 114-123) NY, NY October, 2004.

Brown, L.M., Lu, M., Shu, C., Tian, Y., & Hampapur, A. (2005a) Improving performance via post track analysis. *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance* (pp. 341 – 347) Beijing, China, October 2005.

Brown, L., Senior, A., Tian, Y., Connell, J., Hampapur, A., Shu, C, Merkl, H., & Lu, M. (2005b) Performance Evaluation of Surveillance Systems Under Varying Conditions, *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance* (pp79-87) Breckenridge, CO, January 2005.

Comaniciu, D., Ramesh, V. & Meer, P. (2000) Real-time tracking of non-rigid objects using mean shift, in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume II, (pp. 142–149.)

Feris, R., Tian, Y., & Hampapur, A. (2007) Capturing People in Surveillance Video, *The Seventh International Workshop on Visual Surveillance* (pp. 1-8.)

FRVT, (2006) Face Recognition Vendor Test (FRVT), <http://www.frvt.org/FRVT2006/>.

Francois, A. (2004) Real-Time Multi-Resolution Blob Tracking, *Institute for Robotics and Intelligent Systems, University of Southern California, Los Angeles, California*, from <http://handle.dtic.mil/100.2/ADA447622>.

Funahashi, T.; Fujiwara, T.; Koshimizu, H. (2005) Coarse to fine hierarchical tracking system for face recognition. *IEEE International Conference on Systems, Man and Cybernetics*, Volume 4, (pp. 3454-3459.)

Hauptmann, A. (2006) Lessons for the Future from a Decade of Infomedia Video Analysis Research, *International Conference on Image and Video Retrieval* (pp. 1-10.)

HiTech, <http://www.htsol.com/Products/SeeCar.html>.

Jin, J., Zhu, Z., Xu, G. (2001) Digital Video Sequence Stabilization Based on 2.5D Motion Estimation and Inertial Motion Filtering, *Real-Time Imaging*, Vol. 7, No. 4, Academic Press, (pp. 357-365.)

Lee, H., Smeaton, A., O'Connor, N., & Murphy, N., (2005) User Interface for CCTV Search System, *The IEE International Symposium on Imaging for Crime Detection and Prevention* (pp. 39-43.)

Lucas, B. D., & Kanade, T. (1981) An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging understanding workshop*, (pp 121—130.)

Marcenaro, L., Oberti, F., Foresti, G.L., & Regazzoni, C.S. (2001) Distributed Architectures and Logical-Task Decomposition in Multimedia Surveillance Systems, *Proceedings of IEEE*, Vol.89, No.10, (pp. 1419-1440.)

Meessen, J., Coulanges, M., Desurmont, X., & Delaigle, J.F., (2006) Content-Based Retrieval of Video Surveillance Scenes, *Multimedia Content Representation, Classification and Security*. (pp.785-792.)

Naphade, M. & Smith, J.R. (2004) On the Detection of Semantic Concepts at TRECVID, *ACM International Conference on Multimedia*. (pp. 660-667.)

ObjectVideo, <http://www.objectvideo.com/products/vew/>.

PyramidVision, <http://www.pyramidvision.com/>.

Senior, A., A. Hampapur, Tian, Y, Brown, L., Pankanti, S., & Bolle, R. (2006) Appearance Models for Occlusion Handling, in *Journal of Image and Vision Computing* , Volume 24, Issue 11, (pp. 1233-1243.)

Senior, A., Brown, L., Shu, C., Tian, Y., Lu, M., Zhai, Y. & Hampapur, A. (2007) Visual Person Searches for Retail Loss Detection: Application and Evaluation, *International Conference on Vision Systems*.

Shu, C., Hampapur, A., Lu, M., Brown, L. Connell, J. Senior, A. & Tian, Y. (2005), IBM smart surveillance system (S3): a open and extensible framework for event based surveillance, *IEEE Conference on Advanced Video and Signal Based Surveillance* (pp. 318 – 323.)

Stringa, E. & Regazzoni, C.S. (1998) Content-Based Retrieval and Real Time Detection from Video Sequences Acquired by Surveillance Systems, *IEEE International Conference on Image Processing*, Vol. 3, (pp. 138-142.)

Tian, Y., Lu, M., & Hampapur, A. (2005) Robust and efficient foreground analysis for real-time video surveillance, *IEEE International Conference on Computer Vision and Pattern Recognition*. Vol. 1, (pp1182-1187.)

Velipasalar, S., Brown, L. & Hampapur, A. (2006a) Specifying, Interpreting and Detecting High-level, Spatio-Temporal Composite Events in Single and Multi-Camera Systems, *Conference on Computer Vision and Pattern Recognition Workshop*, (pp110-116.)

Velipasalar, S., Tian, Y. & Hampapur, A. (2006b) Automatic counting of interacting people by using a single uncalibrated camera, IEEE International Conference on Multimedia and Expo, (pp1265-1268.)

Viola, P. & Jones, M. (2001) Rapid Object Detection Using a Boosted Cascade of Simple Features, *IEEE Conference on Computer Vision and pattern Recognition* (pp.511-518.)

15. Additional Reading

Ali, S. & Shah, M. (2007). [*A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis*](#), IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis.

Auvinet, E., Grossman, E., Rougier, C., Dahmane, M. & Meunier, J. (2006). *Left-luggage detection using homographies and simple heuristics*. PETS2006 Proceedings.

Bose, B. & Grimson, E. (2004). *Improving object classification in far-field video*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04) Vol. 2, Washington, D.C., pp 181-188.

Buxton, H. (2003). Learning and understanding dynamic scene activity: a review. *Image and Vision Computing*, Vol 21, pp 125-136.

Davis, J. W. (2004). Sequential *reliable-inference for rapid detection of human actions*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'04), Washington, D.C.

Ferryman, J. & Thirde, D. (2006). *An overview of the PETS2006 dataset*. PETS2006 Proceedings.

Gray, D., Brennan, S. & Tao, H. (2007). *Evaluating Appearance Models for Recognition, Reacquisition, and Tracking*. PETS2007 Proceedings.

Hongeng, S. & Nevatia, R. (2003). *Large-scale event detection using semi-hidden markov models*, IEEE International Conference on Computer Vision, Nice, France.

Joo, S. & and R. Chellappa (2006). *Attribute grammar-based event recognition and anomaly detection*. International Workshop on Semantic Learning Applications in Multimedia, New York , NY.

Katz, B., Lin, J., Stauffer, C., & Grimson E. (2005). *Answering questions about moving objects in surveillance videos*. AAAI Spring Symposium on New Directions in Question Answering.

Khan, S. & Shah, M. (2005). [*Detecting group activities using rigidity of formation*](#). Proceedings of ACM Multimedia.

Krahnstoever, N., Tu, T., Sebastian, T., Perera, A. & Collins, R. (2006). *Multi-view detection and tracking of travelers and luggage in mass transit environments*. PETS2006 Proceedings.

Martinez-del-Rincon, J., Herrero-Jaraba, J., Gomez, J. & Orrite-Urunuela, C. (2006). *Automatic left luggage detection and tracking using multi-camera UKF*. PETS2006 Proceedings.

Pound, M., Naeem, A., French, A. & Pridmore, T. (2007). *Quantitative and qualitative evaluation of visual tracking algorithms using statistical tests*. PETS 2007 Proceedings.

- Remagnino, P. & Jones, G.A. (2001). *Classifying surveillance events from attributes and behaviour*, British Machine Vision Conference, Manchester, pp. 685-694. ISBN/ISSN 1901725162.
- Ramanathan, N. & Chellappa, R. (2006). Face verification across age progression", in *IEEE Transactions on Image Processing*, Vol. 15, pp. 3349-3361.
- Rodriguez, M. & Shah, M. (2007) [Detecting and segmenting humans in crowded scenes](#). Proceedings of ACM Multimedia.
- Shao J., Zhou, S. & Chellappa, R. (2004). *Appearance-based tracking and recognition using the 3D trilinear tensor*. IEEE Intl. Conf. on Acoust., Speech and Signal Processing, Montreal, Canada.
- Sheikh, Y., Li X. & Shah, M. (2007). [Trajectory association across non-overlapping moving cameras in planar scenes](#), IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, USA.
- Stauffer, C. (2003) *Estimating tracking sources and sinks*. IEEE Workshop on Event Mining.
- White, B. & Shah, M. (2007). [Automatically tuning background subtraction parameters using particle swarm optimization](#), IEEE International Conference on Multimedia & Expo, Beijing, China.
- Yin, F., Makris, D. & Velastin, S. (2007). *Performance evaluation of object tracking algorithms*. PETS2007 Proceedings.
- Zhao, T. & Nevatia, R. (2004). *Tracking multiple humans in crowded environment*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 406-413.
- Zhou, S. & Chellappa, R. (2005). Image-based face recognition under illumination and pose variations ", *Jl. Optical Society of America, A*, Vol. 22, pp. 217-229.
- Zhu, X. et al. Video data mining: semantic indexing and event detection from the association perspective. *IEEE Trans. on Knowledge and Data Engineering*, **17**(5), 665-677.