Engineering Applications of Artificial Intelligence 23 (2010) 331–345

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Scenario-based query processing for video-surveillance archives ☆

Ediz Şaykol, Uğur Güdükbay *, Özgür Ulusoy

Department of Computer Engineering Bilkent University, 06800 Bilkent, Ankara, Turkey

ARTICLE INFO

Article history:
Received 9 February 2009
Received in revised form
16 June 2009
Accepted 4 August 2009
Available online 19 September 2009

Keywords:
Video surveillance
Scenario-based querying and retrieval
Visual query specification
Event-based querying
After-the-fact analysis

ABSTRACT

Automated video surveillance has emerged as a trendy application domain in recent years, and accessing the semantic content of surveillance video has become a challenging research area. The results of a considerable amount of research dealing with automated access to video surveillance have appeared in the literature; however, significant semantic gaps in event models and content-based access to surveillance video remain. In this paper, we propose a scenario-based query-processing system for video surveillance archives. In our system, a scenario is specified as a sequence of event predicates that can be enriched with object-based low-level features and directional predicates. We introduce an inverted tracking scheme, which effectively tracks the moving objects and enables view-based addressing of the scene. Our query-processing system also supports inverse querying and view-based querying, for after-the-fact activity analysis. We propose a specific surveillance query language to express the supported query types in a scenario-based manner. We also present a visual query-specification interface devised to facilitate the query-specification process. We have conducted performance experiments to show that our query-processing technique has a high expressive power and satisfactory retrieval accuracy in video surveillance.

1. Introduction

In a traditional surveillance system, a human operator monitors multiple environments simultaneously to detect, and possibly prevent, a dangerous situation. Human perception and reasoning, however, are limited in their ability to process the amount of spatial data perceived by the senses. These limits may vary, depending on the complexity of the events and their time instances. In recent years, the acceleration in capabilities of communication equipment and in automatic video-processing techniques, combined with the decreasing cost of technical devices, has resulted in increased interest in video surveillance applications. In turn, these applications have augmented the capabilities of the human operators.

An automated video surveillance system should support both real-time alarm generation and offline inspection components to satisfy the requirements of the operators (Regazzoni et al., 2001). In either side, the input video stream should be processed appropriately so that the actions are correctly analyzed. The primary challenges are the large input size and the high variability of the audio-visual features, hence it still remains a challenging issue to access the semantic content of the videos automatically.

Video surveillance process generally consists of the following stages: modeling the environment, detecting moving objects (also called regions), classifying objects, tracking objects, and understanding behavior (Hu et al., 2004). Background/foreground subtraction (Collins et al., 2000; Kim et al., 2005; Gutchess et al., 2001; Li et al., 2003; Paschos and Valavanis, 1999; Duque et al., 2006) or temporal template-based methods (Haritaoglu et al., 2000; Bobick and Davis, 2001) are widely used to detect moving objects. One of the basic aims in understanding the objects' behavior is detecting the anomalies in the objects' actions (Duque et al., 2006; Xiang and Gong, 2006, 2008; Zhong et al., 2004; Hamid et al., 2005; Duong et al., 2005). Abnormal situations and anomalies are reported to the operator and/or stored in a database for later inspection (Durak et al., 2007; Şaykol et al., 2005a). One of the basic tasks in offline inspection is the content-based retrieval of surveillance videos from the database. In the literature, the researchers generally assume simple data structures for the semantic content: events and objects. The event descriptors contain time information and the objects acting in the event. Object indexing is not as frequent as event indexing and lacks low-level (or object-based) features.

There are specific situations that can be considered as a sequence of events, and the existence of the whole sequence is of interest. We propose *scenario-based query processing* to detect

☆This work was supported in part by the European Commission's Sixth Framework Program's MUSCLE Network of Excellence Project, with Grant number FP6-507752, and by the Scientific and Technical Research Council of Turkey (TUBITAK), with Grant number EEEAG-105E065.

* Corresponding author. Tel.: +90 312 290 13 86; fax: +90 312 266 40 47.
E-mail addresses: ediz@cs.bilkent.edu.tr (E. Şaykol), gudukbay@cs.bilkent.edu.tr (U. Güdükbay), oulusoy@cs.bilkent.edu.tr (Ö. Ulusoy).

0952-1976/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2009.08.002

such sequences in video-surveillance archives, and hence to reduce the gap between low-level features and high-level semantic content. Our system provides support for querying by event-based and object-based features. The supported query types can be ordered to form a scenario-based query, where the temporal information among the (sub)queries is also included in the query expression.

In our earlier work (Şaykol et al., 2005a), we provided the preliminaries of a system architecture along with a brief introduction on the meta-data extraction process. The architecture of our system is shown in Fig. 1. The focus and contributions of this paper are on the query-processing component, which provides support for scenario-based, event-based, and object-based queries, where the low-level object features and directional predicates can be used to improve expressiveness and effectiveness. Our query model also supports *inverse querying* as well as some statistical view-based query types that can be used as tools for activity analysis in various domains, e.g., video forensics. Our Video Surveillance Query Language (VSQL) has been designed specifically for scenario-based querying purposes. A query-specification interface has also been developed, which can be considered as the visual counterpart of VSQL. The query-specification interface is as generic and flexible as our query model.

The main contributions of our work can be listed as follows:

- The query-processing system we propose provides support for a wide range of query types valuable for video surveillance. The supported query types include event and object queries enriched with the low-level feature descriptors (e.g., color, shape) and directional predicates. The system allows *scenario-based querying*, where a scenario is a sequence of events ordered temporally. This type of querying increases the retrieval quality of offline inspection, and to the best of our knowledge, no video surveillance system has been introduced in the literature that supports scenario-based querying enhanced with object-based low-level features and directional predicates. Our observations have shown that scenario-based querying provides an effective medium for after-the-fact activity analysis, since the abnormalities can be expressed in an effective form while preserving the temporal relations among events with a wide set of low-level subqueries.
- In our data model, we introduce a new data representation scheme for region tracking, which we call *inverted tracking*. The main benefit of this scheme is that it addresses the scene in a view-based manner with respect to the human operators' points of view. Since the operators inspect the scene to determine abnormalities performed by the moving objects, fixed view-based addressing is employed as a part of our data model to enhance its expressive power. For example, the most popular paths of the moving objects and the number of objects entering the scene from the left/right side can be queried.
- We present a rule-based query-processing module devised to provide efficient processing for scenario-based querying. We use Prolog as our inference engine. The meta-data that we extract are generic in the sense that the predicates are valid and valuable in almost every type of video-surveillance application. Our rule-based query-processing module provides a flexible mechanism for external predicate definition (i.e., simply by rule injection) so that the system can be tailored to various domains.
- We provide a textual query language (VSQL) and its visual counterpart to provide complete querying and retrieval capability. These components are designed in a generic and flexible manner so that they can be used in a variety of video-surveillance domains.

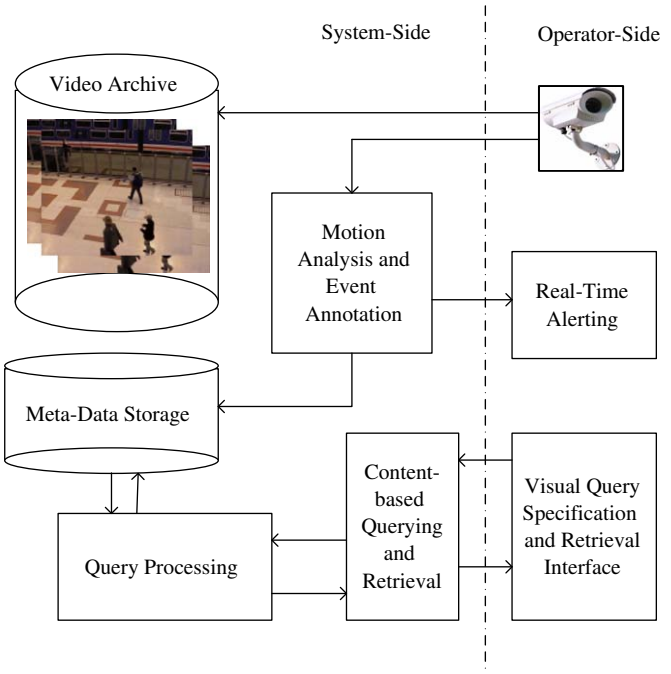

Fig. 1. The architecture of our system. The queries are handled by the query-processing module, which communicates with both the meta-data store and the content-based retrieval module. The meta-data contains event and object features extracted by automated tools. These automated tools also trigger the real-time alerting module. The visual query interface is used to submit queries to the system and to visualize the results.

The rest of the paper is devoted to the detailed description of the approaches we propose for data modeling and querying in video surveillance. Section 2 provides a discussion of the related work. Section 3 presents the data model proposed to achieve querying and retrieval of video surveillance archives. The query-processing capabilities of our work are described in Section 4. Section 5 discusses the visual query-specification interface designed to express scenario-based queries. The results of the performance experiments are presented in Section 6. Section 7 provides a discussion of the unique features of our approach as compared to the state of the art. Section 8 concludes the paper and provides future research directions.

2. Related work

Although there are a significant number of approaches to dynamic scene segmentation (Collins et al., 2000; Kim et al., 2005; Gutchess et al., 2001; Li et al., 2003; Paschos and Valavanis, 1999; Haritaoglu et al., 2000; Bobick and Davis, 2001; Thirde et al., 2006) and abnormal action detection (Duque et al., 2006; Xiang and Gong, 2006, 2008; Zhong et al., 2004; Hamid et al., 2005; Duong et al., 2005), the offline query-processing capabilities are rather limited in most of the existing video-surveillance systems. Retrieving video sequences related to a previously generated alarm is the basic way of querying the semantic content of surveillance videos (e.g., Lyons et al., 2000; Shet et al., 2005). In the following we provide a brief description of the existing video-surveillance systems.

Collins et al.'s (2000) video surveillance and monitoring (VSAM) system is one of the complete prototypes for object detection, tracking, and classification as well as for calibrating a network of sensors for a surveillance environment. The hybrid algorithm developed in that work is based on adaptive background subtraction by three-frame differencing. The background maintenance scheme is based on a classification of pixels

(either moving or non-moving) performed by a simple threshold test. A model is provided on temporal layers for pixels and pixel regions in order to better detect stop-and-go movements.

Stringa and Regazzoni (2000, 1998) and Regazzoni et al. (1998) propose a real-time surveillance system employing semantic video-shot detection and indexing. In their system, lost objects are detected with the help of temporal rank-order filtering. The *interesting* video shots are detected by a hybrid approach using low-level (color) and semantic features. Retrieving all the clips related to an alarm is the basic way of querying the system.

Kim and Huang (2002a, b) present an object-based video abstraction model, where a moving-edge detection scheme is used for video frames. The edge map of a frame is extracted and compared with the background edge map to detect the moving edges and regions. A semantic shot-detection scheme is employed to select object-based key-frames. When a change occurs in the number of moving regions, the current frame is declared as a key-frame, indicating that an important event has occurred. If the number of moving objects remains the same in the next frame, a shape-based change detector is applied to the following frames.

Rivlin et al. (2002) propose a real-time system for moving object detection, tracking, and classification where the video stream originates from a static camera. Effective background initialization and background adaptation techniques are employed for better change detection. The target detection phase also benefits from a color table representing object data. The detected moving objects are classified as *human*, *animal*, and *vehicle* with the help of an expressive set of feature vectors. The authors initiate their feature-vector selection process with a wide set of object-appearance and temporal features. A reduced set, which leads to the best classification accuracy in their experiments, is used for classification. The authors also present a classification approach that combines appearance and motion-based features to increase the accuracy (Bogomolov et al., 2003).

IBM's MILS (MIddleware for Large Scale Surveillance) (Hampapur et al., 2005) system provides a complete solution for video surveillance, including data-management services that can be used for building large-scale systems. MILS also provides query services for surveillance data, including *time, object size, object class, object motion, context-based object content similarity* queries, and any combination of these. The system employs relevance feedback and data-mining facilities to increase its effectiveness.

Lyons et al. (2000) developed a system called video content analyzer (VCA), the main components of which are background subtraction, object tracking, event reasoning, graphical user interface, indexing, and retrieving. They adapt a non-parametric background-subtraction approach based on Elgammal et al. (1999). VCA differentiates between people and objects and the main events it recognizes are *entering scene*, *leaving scene*, *splitting*, *merging*, and *depositing/picking-up*. Users are able to retrieve video sequences based on event queries whose categories are similar to those we use.

Brodsky et al. (2001) designed a system for indoor visual surveillance, specifically for use in retail stores and homes. They assume a stationary camera and use background subtraction. A list of events that the object participates in is stored for each object, simply, *entering, leaving, merging*, and *splitting*.

Shet et al. (2005) presented a visual surveillance system, VidMAP, that combines real-time video-processing algorithms with logic programming to represent and recognize activities. They used Prolog for the high-level rules that correspond to their supported query set. *Entry violation*, *theft*, and *possess* are examples of rules they use to answer specific queries.

3. Data model

Our multi-layered data model contains region-level information at the lowest level; e.g., pixel-data for moving regions and motion orientation. At the next level, object-level information is extracted; e.g., class type (e.g., human, non-human, object group) and color and shape feature vectors. At the third level, primitive object actions are annotated; e.g., stop, move, and enter. At the highest level, conceptual event predicates are detected to identify activities, e.g., crossover, deposit, approach. This information-extraction scheme, shown in Fig. 2, is explained in the following subsections. *Annotated actions* and *conceptual predicates* are the primary aspects of our meta-data and used directly in the query model.

3.1. Extraction of moving regions

We employ an adaptive background maintenance scheme to extract the moving regions (or moving objects) in a video frame, similar to the one proposed in Collins et al. (2000). This technique is combined with three-frame differencing to detect the moving pixels. These pixels are then passed through region grouping methods and morphological operations to identify the moving regions. This technique can be formulated as follows:

Let $I_f(x,y)$ denote the intensity value of a pixel at (x,y) in video frame f. Hence, $M_f(x,y)=1$ if (x,y) is moving in frame f, where M_f is a vector holding moving pixels. A threshold vector $T_f(x,y)$ for a frame f is needed for detecting pixel motions. The basic test condition to detect moving pixels with respect to $T_f(x,y)$ can be formulated as

$$M_f(x,y)=\begin{cases}1 & \text{if } (|I_f(x,y)-I_{f-1}(x,y)|>T_f(x,y)) \text{ and } (|I_f(x,y)-I_{f-2}(x,y)|>T_f(x,y)),\\ 0 & \text{otherwise.}\end{cases} \tag{1}$$

The (moving) pixel intensities that are greater than the background intensities ($B_f(x,y)$) are used to fill in the region of a moving object. This step requires a background maintenance task based on the previous intensity values of the pixels. Similarly, the threshold is updated based on the observed moving-pixel information in the current frame. A statistical background and threshold maintenance scheme is employed, as presented in the following equations:

$$B_0(x,y)=0, \tag{2}$$

$$B_f(x,y)=\begin{cases}\alpha B_{f-1}(x,y)+(1-\alpha)I_{f-1}(x,y), & M_f(x,y)=0,\\ B_{f-1}(x,y), & M_f(x,y)=1,\end{cases} \tag{3}$$

$$T_0(x,y)=1, \tag{4}$$

$$T_f(x,y)=\begin{cases}\alpha T_{f-1}(x,y)+(1-\alpha)(k\times|I_{f-1}(x,y)-B_{f-1}(x,y)|), & M_f(x,y)=0,\\ T_{f-1}(x,y), & M_f(x,y)=1,\end{cases} \tag{5}$$

where α is the learning constant, and the constant k is set to 5 in Eq. (5). $B_f(x,y)$ is analogous to the local temporal average of pixel intensities, and $T_f(x,y)$ is analogous to k times the local temporal standard deviation of pixel intensities computed with an infinite impulse (IIR) filter (Collins et al., 2000).

3.2. Tracking moving regions

We employ a view-based motion tracking approach similar to the motion history image (MHI) technique proposed in Bobick and

Fig. 2. The semantic flow in information extraction.

Davis (2001). MHI detects and tracks the parameters (structure and orientation) of the moving regions. In an MHI, the pixel intensity is encoded as a function of the temporal history of the motion at that pixel, where the pixels that moved more recently are brighter. This technique enables us to track the trajectories of regions. Here, the MHI ($MHI_f(x,y)$) of a frame f is constructed by the update rule in Eq. (6):

$$MHI_f(x,y) = \begin{cases} \tau, & M_f(x,y) = 1, \\ \max(0, MHI_{f-1}(x,y) - 1), & M_f(x,y) = 0, \end{cases} \quad (6)$$

where τ denotes the temporal extent of a motion. Dynamic schemes are available for selecting a better value for τ (e.g., backward-looking algorithm in Bobick and Davis, 2001).

We also provide a scheme to store the object appearance history on a video frame, which we call *inverted tracking*. The name inverted is generally used to imply that a mapping from the content is stored along with the actual content. Here, the actual content is the extracted moving regions, and a view-based mapping is hold along with the content. Fig. 3 illustrates the inverted-tracking technique that we employ on a sample video frame. The video frame $I(x,y)$ is divided into 16 cells corresponding to four subdivisions in x and y directions. The number 16 is selected not only to decrease the computational and storage costs of the system, but also to provide effective positional object-history tracking. We validated the selection of 16 cells by also experimenting with values 4 and 64. If we use four subdivisions, the effectiveness of the tracking is lower. When we use 64 subdivisions, the effectiveness of the tracking is higher but the computational complexity and storage costs are higher. The inverted-tracking technique improves region-tracking capability because object locations are tracked with respect to a fixed view-based reference—the cell in which the object appeared. Since more than one object might appear in a cell, the technique holds object lists for each cell.

While computing a region's appearance in a cell, we consider its center of mass (c_m). The $c_m = (x_{c_m}, y_{c_m})$ is computed as

$$x_{c_m} = \frac{\sum_i^n x_i}{n}, \quad y_{c_m} = \frac{\sum_i^n y_i}{n}, \quad (7)$$

where n is the total number of pixels in a region.

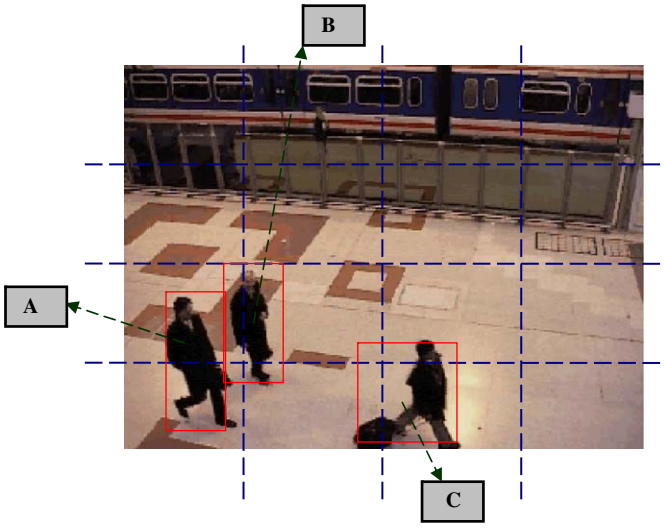

Fig. 3. The inverted tracking scheme illustrated on a sample video frame (Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), 2006). The boxes A, B, and C hold for the structures keeping track of object appearance lists along with a time stamp to keep track of the duration of an object within a cell.

Definition 1 (*Region appearance within a cell*). The region r has appeared in a cell i if c_m of r is inside the boundaries of i inclusively. To break ties, the boundaries are assumed to belong to the cell on the left and down.

If an object is moving in a cell i, it is kept in the object-appearance list of the cell i until it leaves the boundaries of the cell. If the object stops, it is not dropped from the list until a pre-specified time duration passes. A specific type of motion where an object enters the scene, stops for a certain time, and then leaves the scene is called *loitering* (Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS), 2007); this activity is considered as a potential abnormal situation. Hence, our inverted tracking scheme does not drop objects until the pre-specified amount of time (generally taken as 60 s) passes in order to detect loitering, simply by tracing the object appearance lists. However, if the object stops and then starts moving before loitering happened, we can detect that the previously stopped region is moving again by tracing back the object list belonging to that cell. Based on the assumption that the objects will be seen after occlusion before loitering, this delayed dropping of objects from appearance lists also helps to handle the occlusion problem of tracking, and to cope with object tracking errors.

The object-appearance lists are updated under two conditions: the first one is when a new region r is detected. Based on the value of the c_m of r, the cell i that r has appeared in is found and r is appended to that cell's list with a time stamp. The second update condition is when a region r has passed through the boundary between two cells. In this case, r is moved from the list of the cell in which it previously appeared to the list of the newly entered cell.

This scheme helps to address the scene in a view-based manner. Since operators look at a fixed scene and inspect moving objects while trying to figure out the abnormal situations, a fixed view-based addressing provides a medium for view-based querying of the scene content. Examples of the types of queries supported include: the most popular path of the moving objects, the number of objects entering the scene from the left/right side, and the cell in which a specific event happened most frequently.

3.3. Classification of objects

To keep our system generic, we categorize the (moving) regions into three classes: *human*, *non-human*, and *object-group*. The low-level color and shape features and the aspect-ratio of the regions' MBRs are used for classification. The low-level features are stored in normalized color and shape feature vectors, as described in Şaykol et al. (2005b). The color vector stores the distance-weighted intensity values in order to take the color distribution around a pixel into account. This type of color-feature encoding is different from traditional color vectors, and aids the object-classification process. The shape vector is a composition of a region's *angular* and *distance* spans, which are computed based on the region's center of mass (c_m) (Şaykol et al., 2004). These encoding schemes are invariant under scale, rotation, and translation, and effective for object classification.

The color and shape features, as being the most frequent descriptors, are used in most of the existing systems in a combined way. The main reason for this combined usage is to improve the retrieval effectiveness. In our k nearest neighbors based classification scheme, the color and shape information can be linearly combined with proper weights, and we use the temporal averages of these vectors computed during tracking. A global distance value can be obtained by linear combination of three partial distances with appropriate weights during the classification process (Jain and Vailaya, 1996). A possible set of weights can be determined by performing similarity calculations

for each of the two feature vectors separately. The average-case retrieval accuracies as a result of these two similarity calculations are normalized and used as feature weights in the linear combination. This pre-computed weight assignment provides more effective results since this approach reflects the characteristics of the datasets.

For the similarity calculations between test objects and trained objects, we used the *histogram intersection* technique (Swain and Ballard, 1991). In this technique, two normalized histograms (i.e., combined feature vectors) are intersected as a whole to determine a distance value. Both of the histograms must be of the same size, and the distance between the histograms is a floating point number between 0 and 1. Equivalence is designated with similarity value 1, and the similarity between the two histograms decreases in parallel with the similarity value approaches to 0.

Let $H_1[1 \dots n]$ and $H_2[1 \dots n]$ denote two normalized histograms of size n, and S_{H_1,H_2} denote the similarity value between H_1 and H_2. Then, S_{H_1,H_2} can be expressed as

$$S_{H_1,H_2} = \frac{\sum_i^n \min(H_1[i], H_2[i])}{\min(|H_1|, |H_2|)}, \tag{8}$$

where $|H|$ denotes the L_1-norm (i.e., length) of an histogram H.

We used the PETS 2006 and PETS 2007 benchmark datasets for the training phases. A fivefold cross-validation method is employed and k is chosen as 10. The preprocessing steps include the extraction of the moving regions and their corresponding color and shape vectors. The weights of the color feature vectors are found to be lower than the weights of the other two feature vectors (i.e., distance span and angular span) as expected, since the color values of the objects are less discriminative in video surveillance domain. The classification algorithm outputs the percentages for the class values (e.g., 47% human, 34% non-human, 19% object-group), and the moving region is classified by the highest percentage value. To improve our classification scheme, we are planning to embed more effective clustering techniques to further split the object groups.

3.4. Annotation of events

Having classified the moving objects in a video sequence, we annotated the actions of objects automatically. Counting the number of moving objects gives an important clue about the annotation of an event, since the number of moving objects changes at the time of an event. Hence, we utilized a keyframe-based annotation process, where a keyframe is identified when the total number of moving objects changes or the cell-id of an object changes. This keyframe-based processing provides an easier way for detecting conceptual abnormalities. This type of annotation and meta-data usage reduces the search space and enables efficient querying when the video archives are large (Dönderler et al., 2004).

3.4.1. Single object annotation

Our system detects the following basic single-object actions: *enter*, *leave*, *stop*, *stop-and-go*, and eight directional forms of *move* coupled with the directional predicates (see `<direction>` in Appendix A). The orientation of a moving region, as suggested by our tracking algorithm, is used to directly annotate move actions. When an object stops and moves again later, we annotate the action as a stop-and-go type after detecting the object's next move within a certain time interval. If this type of action takes more time than allowed, then we annotate the action as loitering. The other types of single-object actions are rather easier to detect using the inverted tracking algorithm.

3.4.2. Multi-object annotation

Multi-object actions are also annotated in our framework. These actions are *approach*, *depart*, *deposit*, *pick-up*, *crossover*, and *move-together*. The first two can be identified by tracking the Euclidean distance between two objects with respect to the center of mass of the objects in consecutive frames. If the distance between two objects in a previous frame is greater/smaller than the distance in the current frame, these two objects are approaching/departing to/from each other. The remaining four types of multi-object actions are the primary sources for anomalous situations. They are relatively harder to detect and require complex mechanisms. The directional relations between the moving objects are also handled by the help of the tracking scheme. Fig. 4 illustrates the methods that we used to detect these events.

If more than two objects are detected in a frame, the multi-object actions (predicates) are extracted for all of the object pairs. Single-object predicates are also extracted throughout the appearance of the objects in the scene. The following sequence of operations illustrates the event-based meta-data extraction for two moving objects:

- Two objects O_1 and O_2 are identified. The annotation scheme throws *enter*(O_1) and *enter*(O_2).
- On the next frame, *move* predicates are thrown with the valid directions for O_1 and O_2.
- Based on the Euclidean distance between O_1 and O_2, *approach* or *depart* predicate is thrown.
- If O_1 is human and O_2 is non-human, there is potential for a *deposit* or *pick-up* event (see Fig. 4(a) and (b)).
- If both O_1 and O_2 are human, the event could be *crossover* and/or *move-together*, depending on the humans' orientations (see Fig. 4(c) and (d)).
- If both O_1 and O_2 are non-human, the event could be *move-together*, assuming that the two objects are thrown by a human outside the camera's field of view.
- If O_2 stops and O_1 continues its motion at a later frame, *deposit* event is detected. Throughout this detection, corresponding *move* and *depart* predicates for both objects are thrown by the algorithm as the position, velocity, and orientation of the moving objects are tracked.
- To improve the accuracy of retrieval, the directional relations for multi-object actions are also identified (e.g., *deposit*($O_1, O_2, south$) is extracted).

3.5. Meta-data extraction

The meta-data is generic in the sense that the extracted predicates appear in almost any type of video-surveillance application. The set of rules used in querying can be tailored to specific applications by defining domain-specific predicates in terms of these basic ones. The extracted meta-data includes both object-based facts (i.e., class value, color, shape) and event-based facts (event-label, acting objects, frame number), which are stored at keyframes in a video. The cell-id suggested by the inverted-tracking scheme is also stored with the object-based and event-based meta-data facts to provide view-based querying support. We inserted a video-id descriptor to each of the facts to discriminate the meta-data with respect to different video files in an archive.

The extracted meta-data can be partitioned into two parts: *object-based* and *event-based*. The object-based facts contain the class value of the object as well as high-level color and shape descriptors (see `<classdesc>`, `<colordesc>`, and `<shapedesc>` in Appendix A). The class value is determined

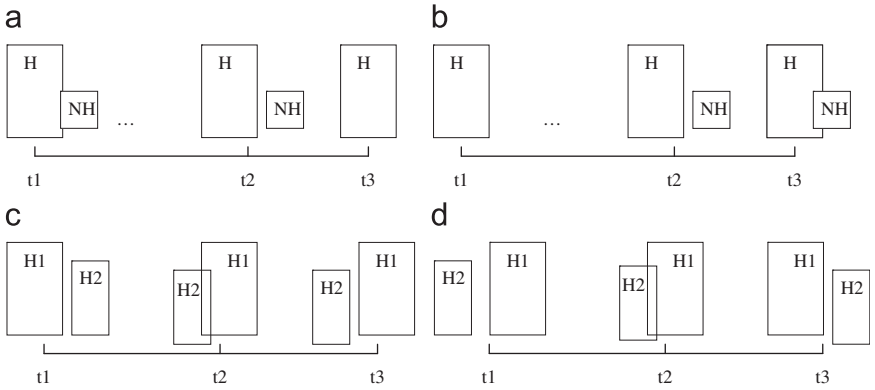

Fig. 4. Annotation of events. (a) Two objects are detected at t1, and they are classified as one human (H) and one non-human (NH). Then, they are detected as two single objects at t2, and when NH stops at t3, *deposit* is identified. (b) Similar to (a), but *pick up* is identified. (c) H1 and H2 are classified as human at t1, and an object group is detected at t2. By tracking the orientations of the objects within t1–t3 time interval, *crossover* is identified. (d) Similar to (c), except that *move together* is identified because H1 and H2 move in the same direction.

by the classification algorithm. High-level feature descriptors are determined by performing similarity calculations between the object feature vectors and the pre-defined vectors. For the color descriptor, the primary colors are encoded in a color vector that we use in object classification. The primary color vectors and the object color vectors are intersected (using the histogram intersection technique), and the most similar primary color label is chosen as the high-level color descriptor for the object. Similar computations are performed for the high-level shape descriptors. Since our feature vectors are scale and rotation invariant, sample figures for *box*, *cone*, *cylinder*, and *sphere* are used to encode shape vectors. The most similar pre-defined shape label is chosen as the high-level shape descriptor for the object. Since the color and shape feature vectors of the objects may change in time (especially color), this object information is stored along with the keyframe number and cell-id. The object information fact is specified as follows:

```
object-info(video-id, object-id,
  object-class-desc, object-color-desc,
   object-shape-desc, keyframe-number, cell-id).
```

Event-based meta-data is composed of facts for both single-object and multi-object events. Since an event occurs at a specific keyframe, the meta-data for event-based facts are stored at keyframes. The event labels for both single and multi object event types are also stored separately with the video-ids and keyframe number to be used within the inverse querying support. These event facts are specified as follows:

```
event-info(video-id, event-label,
  keyframe-number, cell-id).
single-object-event-label(video-id, object-id,
  keyframe-number, cell-id).
multi-object-event-label(video-id,
  first-object-id, second-object-id,
   direction, keyframe-number, cell-id).
```

We used an interval-based extension scheme to utilize the fact storage mechanism. If the same event is triggered for at least two consecutive keyframes, a keyframe interval is stored with the event fact instead of separate facts with consecutive keyframe numbers. This interval extension is applied for all facts where it is applicable. This type of fact storage reduces the storage costs of the system significantly. Query processing mechanism is also capable of processing interval based fact storage. A sample listing for the facts-base to clarify the meta-data storage mechanism that we use is given in Appendix C.

4. Query model

One of the most important tasks in automated video surveillance is query processing. Existing systems generally support textual searches for event queries (Shet et al., 2005; Stringa and Regazzoni, 2000). Some systems also support object queries to some extent (Hampapur et al., 2005). However, not every abnormal situation can be queried by keywords, predicates, etc. Some situations can be treated as a sequence of events and the whole sequence is of interest. One of the main contributions of our work is the *scenario-based querying* capability to detect such sequences, which is not easy to handle in real time. By ordering the events in a scenario, the temporal information about the events is included in the query specification.

Scenario-based querying by an effective set of semantic and low-level features improves the retrieval effectiveness of the framework and decreases the time needed for offline inspection. These gains are more meaningful when the number of events to be searched is relatively large and hard to identify as suspicious in real time, as in after-the-fact analysis in video forensics.

We observe that there is a need for enhancing object queries with low-level feature descriptors. When suspicious events occur, directional specifications about objects give valuable information. Our approach provides support for a wide range of event and object queries, including low-level features and directional predicates, to be posed as a part of the scenario. We also include some specialized query types to provide coherent support for after-the-fact activity analysis. *Inverse querying*, *most popular path*, and *most abnormal region* query types are currently supported. Due to the flexible nature of our data and query models, more complex queries can be formulated.

4.1. Query expression and processing

There are two types of queries: *simple* and *complex*. Simple queries have 12 single-object and six multi-object query types. Single-object queries are *enter/leave scene*, *stop/stop-and-go*, and *move* in eight directions. Multi-object queries include *approach/depart*, *deposit/pick-up*, and *crossover/move-together*. Simple query types can be ordered semantically to form a scenario query, and a *timegap* value can be specified between simple subqueries.

Complex queries are high-level queries that enable scene analysis and statistical offline inspection. *Inverse querying*, *most popular path*, and *most abnormal region* are specific complex query types currently supported in our model.

Our video surveillance query language (VSQL) provides support for integrated querying of video surveillance archives by semantic and low-level features. Semantic subqueries contain 12 single-object event types and six multi-object event types, which can be combined to form more complex queries. Descriptive keywords can be supplied for the color and shape features of objects. Instead of a detailed expression of these low-level features, an intuitive way of query specification is chosen in our model, since it is more realistic that the human operators inspecting (i.e., querying) the videos would choose these features themselves from a set of pre-specified labels corresponding to primary colors. VSQL also provides support for inverse querying. The grammar for VSQL is given in Appendix A. We are also planning to include query-by-example and query-by-sketch types of strategies in the later stages of our work.

Rule-based modeling is effective for querying video databases (Dönderler et al., 2004). Hence, a rule-based model has been designed for querying video surveillance archives. Fig. 5 shows the flow of execution in our query processing scheme. A VSQL query is sent to our inference engine, Prolog, which processes the meta-data (i.e., fact-base) by using a set of rules (i.e., rule-base). Our rule-base is customizable to specific applications since external predicates can be defined in terms of the existing events. Prior to this rule-based processing, the submitted query string is parsed using a lexical analyzer. Variables (objects unbounded to a value prior to querying) can be specified as part of the query that is to be bound to meta-data after query processing. *Scenario with bounded atoms/variables* is processed by the inference engine to produce the result set. The following examples show the formulation of scenario-based queries in VSQL.

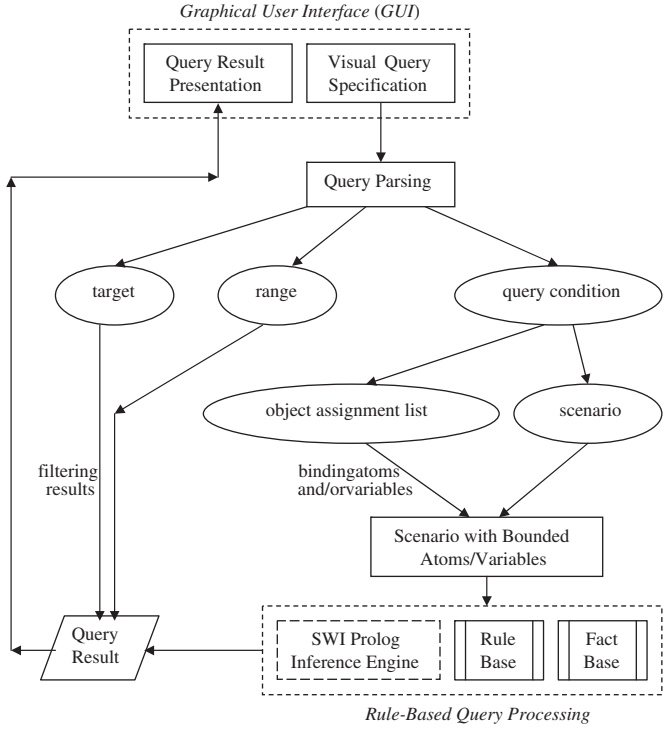

Fig. 5. The query-processing flowchart. A VSQL query submitted to the GUI passes through the parsing, binding, and processing steps; the results are then presented to the user.

Query 1. *A person enters a lobby with a bag, deposits his bag, and leaves the lobby.*

```
select    segment
from      all
where     objectA = objdata(class=human),
          objectB = objdata(class=non-human)
          and enter(objectA) enter(objectB)
          deposit(objectA,objectB) leave(objectA)
```

The query strings are formed by object conditions and event conditions. Object conditions are expressed by $<\texttt{objcondition}>$, and event conditions are expressed by $<\texttt{event}-\texttt{condition}>$, as stated in the grammar for VSQL in Appendix A. Based on the fact-base in Appendix C, the result of the scenario-based query is segment `[12,17]`.

Query 2. *Two people enter a lobby, they meet, shake hands, and then leave.*

```
select  segment
from    all
where   objectA = objdata(class=human), objectB =
        objdata(class=human)
        and enter(objectA) enter(objectB)
        crossover(objectA,objectB) leave(objectA)
        leave(objectA)
```

Our query model can be easily tailored to various video-surveillance domains. The extracted semantic predicates are the basic ones that can happen in almost every video-surveillance application, and since a rule-based model is chosen for query processing, different context models for different domains can be generated by extending the predicate specifications. This extensible nature of our system makes it more expressive and more practical compared to most of the existing systems.

4.2. Scenario-based querying

A scenario-based query consists of a sequence of single-object and/or multi-object event subqueries ordered temporally. Satisfying a scenario-based query means that all subqueries in the scenario have to occur in a specific temporal interval. Hence, the result of a scenario-based query is a set of intervals, where each subquery is satisfied in an interval element in the result set.

Definition 2 (*Result of a scenario-based query in a video*). The result $R_{S,i}$ of a scenario-based query S in a video v_i is a set of intervals specified as follows:

$R_{S,i} = \{[s_S, e_S] |$ all the events in S exist in order within $[s_S, e_S]$ in video $v_i\}$.

Definition 3 (*Result of a scenario-based query*). The result R_S of a scenario-based query S for all the videos in the archive is a set of pairs specified as follows:

$R_S = \{(i, R_{S,i}) | i$ denotes the index of video v_i in the archive$\}$.

The intervals in the result sets can be categorized into two types: *non-atomic* and *atomic*. Non-atomicity implies that the condition holds for every frame within the interval. Thus, the condition holds for any subinterval of a non-atomic interval. Conversely, the condition associated with an atomic interval does not hold for all its subintervals. The intervals in the results of scenario-based queries are atomic, hence they cannot be broken into parts. With this fact in mind, logical conjunction and disjunction operations can be applied to the results.

$R_C = R_1 \wedge R_2$ is computed as follows:

1. initialize the conjunction R_C to $\emptyset$,
2. $\forall_i$ $(i, R_{1,i})$-pair in R_1
   - 2.1 if $\exists$ $(i, R_{2,i})$-pair in R_2, then
     - 2.1.1 compute $R_{C,i}$ as $R_{1,i} \wedge R_{2,i}$,
     - 2.1.2 append $(i, R_{C,i})$ to R_C

$R_{C,i} = R_{1,i} \wedge R_{2,i}$ at line 2.1.1 is computed as follows:

1. initialize the conjunction $R_{C,i}$ to $\emptyset$,
2. $\forall_j$ where $[s_{1,j}, e_{1,j}]$ is the j^{th}-interval in $R_{1,i}$
   - 2.1 $\forall_k$ where $[s_{2,k}, e_{2,k}]$ is the k^{th}-interval in $R_{2,i}$
     - 2.1.1 if $s_{2,k} < s_{1,j}$ & $e_{2,k} > e_{1,j}$, then append $[s_{2,k}, e_{2,k}]$ to $R_{C,i}$,
     - 2.1.2 if $s_{2,k} > s_{1,j}$ & $e_{2,k} < e_{1,j}$, then append $[s_{1,j}, e_{1,j}]$ to $R_{C,i}$,
     - 2.1.3 otherwise $R_{C,i}$ remains unchanged.

Fig. 6. The conjunction operation applied on query results.

$R_D = R_1 \vee R_2$ is computed as follows:

1. initialize the disjunction R_D to $\emptyset$,
2. $\forall_i$ $(i, R_{1,i})$-pair in R_1
   - 2.1 if $\exists$ $(i, R_{2,i})$-pair in R_2, then
     - 2.1.1 compute $R_{D,i}$ as $R_{1,i} \vee R_{2,i}$,
     - 2.1.2 append $(i, R_{D,i})$ to R_D
   - 2.2 else append $(i, R_{1,i})$-pair to R_D

$R_{D,i} = R_{1,i} \vee R_{2,i}$ at line 2.1.1 is computed as follows:

1. initialize the disjunction $R_{D,i}$ to $\emptyset$,
2. $\forall_j$ where $[s_{1,j}, e_{1,j}]$ is the j^{th}-interval in $R_{1,i}$
   - 2.1 append $[s_{1,j}, e_{1,j}]$ to $R_{D,i}$
3. $\forall_k$ where $[s_{2,k}, e_{2,k}]$ is the k^{th}-interval in $R_{2,i}$
   - 3.1 append $[s_{2,k}, e_{2,k}]$ to $R_{D,i}$,
4. remove $R_{C,i} = R_{1,i} \wedge R_{2,i}$ from $R_{D,i}$.

Fig. 7. The disjunction operation applied on query results.

Assume two scenario-based queries and their results R_1 and R_2, that contain atomic intervals. Figs. 6 and 7 present the pseudo-codes for obtaining the conjunction $R_C = R_1 \wedge R_2$ and the disjunction $R_D = R_1 \vee R_2$ of the query results, respectively. An example is provided in Appendix B to elaborate on these operations.

4.3. Event-based querying

Our query model provides support for event-based querying. For example, users may want to query all the occurrences of a single event of any single-object or multi-object type in the archive. The result of an event-based query is a set of frames where the event specified in the query has occurred, rather than a set of intervals. Query 3 is an example of event-based querying. Since the result of an event-based query is a set of frames, the logical conjunction, and disjunction operations can be applied directly on the results.

Definition 4 (*Result of an event-based query in a video*). The result $R_{E,i}$ of an event-based query E in a video v_i is a set of frame numbers, specified as follows:

$R_{E,i} = \{f_E |$ event E occurs in $f_E\}$.

Definition 5 (*Result of an event-based query*). The result R_E of an event-based query E for all the videos in the archive is a set of pairs, specified as follows:

$R_E = \{(i, R_{E,i}) | i$ denotes the index of video v_i in the archive$\}$.

Query 3. *Where have all the crossovers happened in videos* 1 *and* 4?

```
select      frames
from        1,4
where       objectA = objdata(class=human),
            objectB = objdata(class=human)
            and crossover(objectA, objectB)
```

4.4. External predicate definition

Our query model allows for defining external predicates in terms of the existing ones. The following is a simple example of external predicate definition following the Prolog conventions. The 16-cell grid is ordered row-wise, starting from the top-left corner, with cells 1, 5, 9, and 13 on the left. Thus, *enter-left* specifies a predicate of entrances from the left of the scene.

```
enter-left(X, F, G) :-      enter(X, F, 1);
                            enter(X, F, 5);
                            enter(X, F, 9);
                            enter(X, F, 13).
```

Once the *enter-left* predicate has been specified, it can be used as the existing predicate in scenario-based and event-based querying, as shown in Query 4. It should be noted that *move-any* is another external predicate defined to query the *move* action in any of the eight directions.

Query 4. *List all the loitering intervals caused by the entrances from the left in video* 3.

```
select      frames
from        3
where       objectA = objdata(class=human) and
            enter-left(objectA) stop(objectA) 60
            move-any(objectA)
```

4.5. Object-based querying

Our query model supports object-based queries in various ways. First, the existence or appearance of an object can be queried. The result of an object-based query is a set of frames where the object has appeared. The low-level features (color, shape) and class values (human, non-human, object-group) of the objects can be used to enrich the query. As in event-based querying, logical conjunction and disjunction operations can be applied directly since the result of an object-based query is a simple set.

Query 5. *List the frames where a black object has appeared.*

```
select frames
from   all
where  objectA = objdata(class=non-human,
       color=black)
```

Another type of object-query specification uses the *unification* concept in Prolog, a mechanism that binds variables to atoms. The query processor returns all the objects satisfying some pre-specified conditions, such as color, shape and class. The result of this type of querying is a list of object labels bound to the variables in the query. This type of querying is more meaningful when the video archive is well annotated, which means that the objects in videos have been assigned labels (e.g., domain-specific names) *a priori*.

Query 6. *List the names of all the persons with a black coat.*

```
select OBJECTX
from   all
where  OBJECTX = objdata(class=human,
       color=black)
```

4.6. Complex querying

In addition to simple queries, we provide a set of complex queries to provide coherent support for after-the-fact activity analysis. *Inverse querying*, *most popular path*, and *most abnormal region* query types are currently supported. Due to the flexible nature of our data and query models, more specific types of complex queries can be introduced at any time; we are planning to add more queries in the later stages of our work.

4.6.1. Inverse querying

Inverse querying means retrieving the list of events (see Query 7) or objects (see Query 8) appearing within a certain time interval in a video. This inverse querying is very valuable for offline inspection and is most effective when domain-specific activity analysis is of concern.

Query 7. *Which events occurred between frames* 100 *and* 1000 *in video* 1?

```
select     events
from       1
where      inverse(100, 1000)
```

Query 8. *Which objects appear between frames* 100 *and* 1000 *in videos* 3 *and* 4?

```
select     objects
from       3,4
where      inverse(100, 1000)
```

4.6.2. View-based querying

In our inverted-tracking scheme, we divide the scene into 16 cells to enhance the expressive power of our query model. Queries 9 and 10 are examples of the *most popular path* and the *most abnormal region* query types. The results of these queries are based on the row-wise indexing of the 16 cells, where the first cell is in the top-left corner. The most abnormal region is determined by inspecting the high-level events (i.e., deposit, pick-up, crossover, move-together). The cell-id that these events occur most is selected as the most abnormal region. This might be a set of regions since the event occurrences are counted.

The most popular path is determined by inspecting the underlying data model for inverted tracking. The cell through which the objects enter the scene most is identified as the *most frequently entered region*, which is the initial point for the most popular path. The end of this path is traced among the remaining three sides of the scene. If the *most frequently entered region* is a corner cell, the tracing is carried out among the remaining two sides. The tracing operation to find the end of the path includes determining the cell-id that objects leave the scene most. Having determined the start, end, and the direction of the path, the cells between the start and end cells are traversed such that the cells visited by the objects most are selected.

Query 9. *What is the most popular path in video* 1?

```
select     most-popular-path
from       1
```

Query 10. *What is the most abnormal region in videos* 3 *and* 4?

```
select     most-abnormal-region
from       3,4
```

The result for Query 9 is $\{(1, \{5, 6, 10, 11, 12\})\}$. The starting cell is found to be 5 and the end cell is determined as 12. All the in-between cells (i.e., $6, 9, 10, 7, 11, 8$ in traversing order) are inspected and the most popular path is obtained. The result for Query 10 is $\{(3, \{7\}), (4, \{14\})\}$.

5. Visual query specification

We include a visual query interface to provide a mechanism for an intuitive way of VSQL query specification. This interface is easy to use and devised in a flexible manner, which makes it easily adaptable to various domains. The event predicate labels are manageable through XML-based configuration files; hence, domain-specific or user-dependent event predicates can be used to express queries.

The visual query interface provides *object specification*, *event specification*, and *scenario specification* facilities. It also utilizes an XML-based repository to make the specified objects available for later use. The scenarios are expressed as a sequence of events and the order of events can be changed to obtain various scenario combinations. The query results are presented in a separate window where the user can browse the results. The specification of Query 11 through the interface is illustrated in Fig. 8.

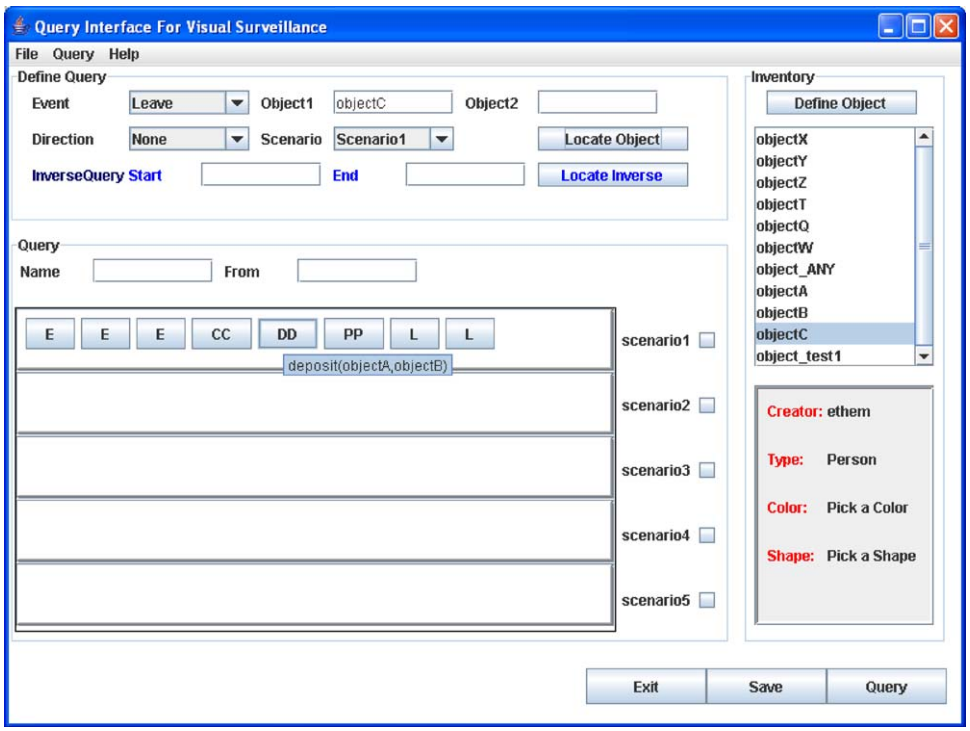

Fig. 8. The specification of Query 11. The event sequence corresponding to the scenario is shown in the first row of the scenario drawing panel. The letters in the boxes on the drawing panel ('E', 'CC', 'DD', 'PP', and 'L') denote enter, crossover, deposit, pickup, and leave event predicates, respectively.

Query 11. *List the segments from video* 1 *where two persons, one with a bag, meet*; *then the person carrying a bag leaves the bag*; *the other person takes the bag*; *and then both persons leave.*

```
select segment
from   1
where  objectA = objdata(class=human),
       objectB = objdata(class=non-human),
       objectC = objdata(class=human) and
       enter(objectA) enter(objectB)
       enter(objectC)
       crossover(objectA,objectC)
       deposit(objectA,objectB)
       pickup(objectC,objectB) leave(objectA)
       leave(objectC)
```

Having specified the events by choosing the objects acting in the event, the scenario is defined based on these events. The scenario drawing panel can be considered a timeline, a widely used query-specification technique for sequence-based data to model temporal relations among events. In this panel, the events can be reordered and time gap between events can be adjusted, which brings more flexibility to scenario-based query specifications.

6. Performance experiments

We tested the retrieval performance of our scenario-based query-processing system by evaluating the methods used in the meta-data extraction process. First, we evaluated the performance of the pixel-level algorithms. Next, we evaluated the object-classification algorithms. The last set of experiments evaluated our semantic annotation process. Since we utilize an SQL-based querying language and use Prolog as our inference engine, the accuracy of our system strictly depends on the peak performance of the above three components.

The evaluation of these algorithms is non-trivial and subjective. In Nascimento and Marques (2006), a discussion on the performance evaluation of object-detection algorithms is given. Among the standard measures, receiver operating characteristics (ROC) analysis (Fawcett, 2006) is used to inspect the effect of a single parameter to the classifier by plotting the true-positive-rate (TPR) and false-positive-rate (FPR) values that are calculated while keeping all the other parameters fixed. Since the algorithms in our keyframe labeling technique yield binary outputs, a set of points is plotted on ROC curves. To elaborate on this, our keyframe labeling algorithm yields exact labels instead of label percentages for keyframes. Hence, a set of points is plotted on ROC curves for the rest of the tests. The points over the $x = y$ line are considered as good classification results, whereas the ones below are bad. The best classifier is considered to be at point $(0, 1)$, which is the farthest point to the $x = y$ line.

The benchmark data sets provided by PETS 2006 and PETS 2007 were used as our ground truth for the performance experiments, along with annotations that we performed manually from a set of indoor monitoring videos captured by a static camera at our university. We employed a fivefold cross-validation method for the experimental evaluation.

6.1. Performance of the pixel-level processing

We employ motion-detection, color-feature, and shape-feature extraction algorithms for the pixel-level processing techniques. We adapted widely used background subtraction and maintenance schemes for motion detection. For motion tracking, we employed our inverted tracking scheme using motion-history images. For color and shape features, we adapted distance-weighted color and shape vectors, which are composed of angular spans and distance span. Regarding representation and classification, these two feature vectors are shown to be effective through a wide range of performance experiments.

Since ROC analysis requires ground truth evaluation for each parameter setting, we focused on two parameters: α (the learning constant, in Eqs. (3) and (5)) and τ (the temporal duration of the movement, in Eq. (6)). Two sets of experiments were carried out for this analysis. In both of the experiments, the learning constant α varies from 0.6 to 0.8 in increments of 0.05, yielding five points in the ROC curve. The temporal duration constant τ is set to 2 and 3. The extracted regions are annotated manually, and correctly detected pixels are considered as true (Nascimento and Marques, 2006). As shown in Fig. 9, the optimal values for the two crucial pixel-level parameters are found to be 2 for τ, and 0.7 for α, since the points corresponding to $\tau = 2$ are closer to the (0, 1) point. The point corresponding to $\tau = 2$, $\alpha = 0.7$ is the farthest to the $x = y$ line.

6.2. Performance of object classification

The object-classification algorithm that we used takes region data as the input and yields classification values as percentages. The maximum of these values is chosen as the class value for the input region. Hence, our classification algorithm outputs three classes: human, non-human, and object-group.

In this evaluation, we computed two types of classification accuracies: *standard classification accuracy* (*SCA*) and *frame-based classification accuracy* (*FCA*). The former is the percentage of correctly classified objects for each class type, and the latter is the frame-weighted percentage of classification accuracy for each object classified correctly. To elaborate on these definitions, Eqs. (9) and (10) are given for the human class type as follows:

Let CC_H denote the set of correctly classified humans, and $|CC_H|$ denote its cardinality. The SCA for the human class type is formulated as

$$SCA_H = \frac{|CC_H|}{|H|}, \tag{9}$$

where H is the set of objects of human class type.

Let F_H denote the set of frames where at least one human is present, and FCC_H denote the set of frames where at least one human is classified correctly. The FCA for the human class type is formulated as

$$FCA_H = \frac{|FCC_H|}{|F_H|}. \tag{10}$$

Fig. 10 presents the results of the object-classification analysis. Since the number of frames (i.e., keyframes) that an object is classified correctly was taken into consideration, the frame-based accuracy analysis gives better results. The lowest accuracy improvement in the frame-based analysis is for the human class type; many people enter and leave the scene, and in most cases, they are classified correctly. Frame-based classification accuracies for the other two object classes improved significantly when compared to standard accuracies.

6.3. Performance of semantic annotation

To evaluate the performance of our semantic annotation process, the detection of each manually annotated event type is inspected. Since both scenario-based and inverse queries are processed by our inference engine, the retrieval accuracy can be evaluated by the performance of the event-detection mechanism. To be more realistic, the percentage of the frames in which the events are correctly identified is used to judge the accuracy of the event annotation instead of just counting the correctly identified event types. This analysis is similar to the frame-weighted classification accuracy above, and the results are presented in Fig. 11.

As shown in Fig. 11, similar event types are grouped together to simplify the analysis. The accuracy is very high for enter/leave, move, and approach/depart events because they can be detected directly from region extraction. Since we incorporate the number of frames in which an event is identified into the accuracy value, multi-object event types have lower accuracy values than the other event types. This is simply because regions detected as

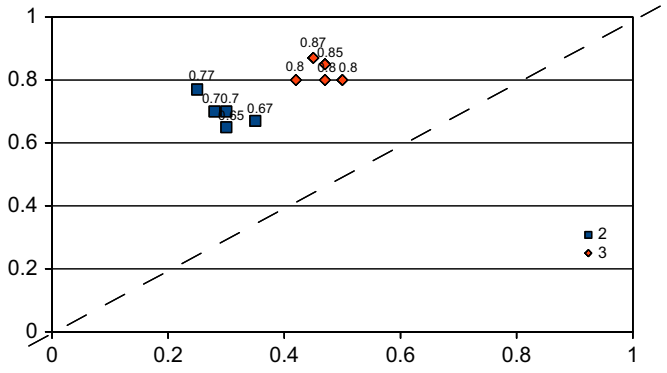

Fig. 9. ROC curve/analysis of pixel-level algorithms. The learning constant α varies from 0.6 to 0.8 in increments of 0.05. ◆ denotes a test configuration when τ is set to 2, whereas ■ denotes a test configuration when τ is set to 3.

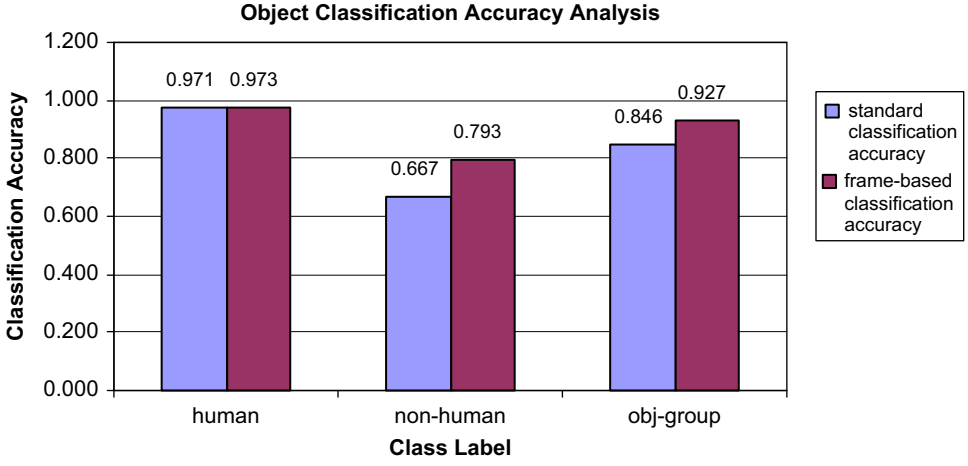

Fig. 10. Object classification accuracy analysis.

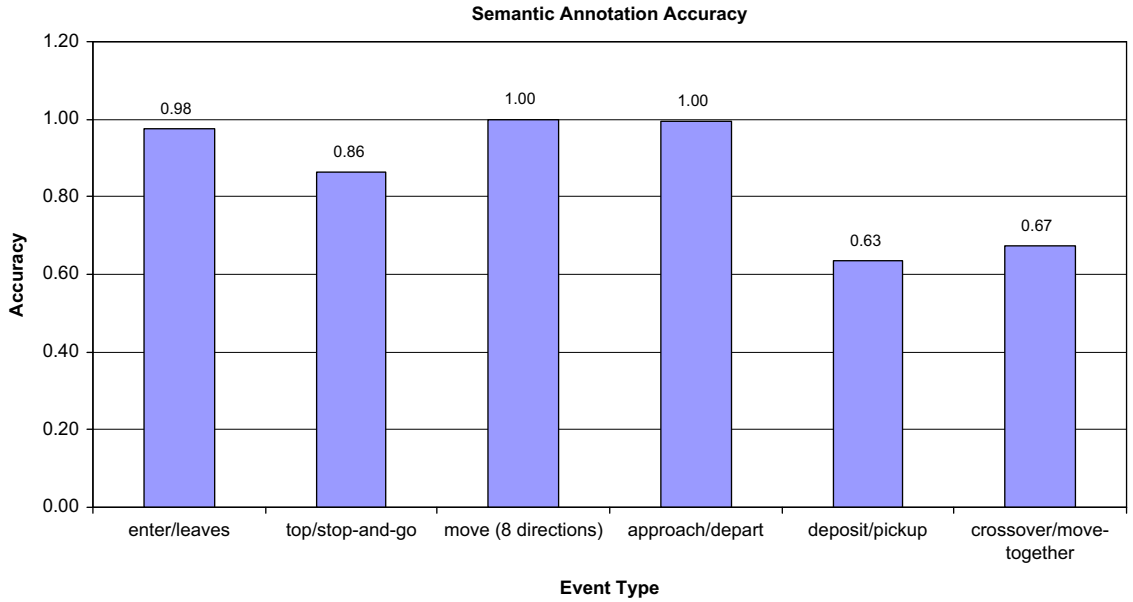

Fig. 11. Semantic annotation accuracy analysis.

object-groups cannot be split into single objects in the frames they are detected.

If an event is detected in at least one frame during its occurrence period, the scenario-based querying and retrieval process is still be effective. This observation is also true for inverse querying because the event is retrieved if it is detected in even a single frame within the querying interval. Since we utilize Prolog as the inference engine in the querying process, the query processor will retrieve what is extracted as meta-data. However, depending on the accuracy of the meta-data extraction process, there might be events that cannot be retrieved since they are not extracted as meta-data either (see Fig. 11). It can be concluded that the performance of scenario-based querying and retrieval is at least as high as the performance of the semantic annotation process.

Our meta-data extraction scheme supports a wide range of event predicates, and provides external predicate definition in terms of the existing predicates. This capability is provided not only to increase the expressive power of the query language but also to make the querying mechanism tailorable for specific video surveillance applications. Undoubtedly, there can be events that cannot be specified using our event predicates, but in our opinion, we have covered an adequate set of events that can be considered as abnormal situations in most of the video surveillance applications.

7. Comparison to related work

Query effectiveness relies on the meta-data extraction process. The background maintenance scheme we employ is similar to that of VSAM (Collins et al., 2000) with an improvement that the extracted background is also used for object tracking. We employ a strategy of moving-object counting similar to the one presented in Kim and Hwang (2002a), extended with a keyframe detection scheme to provide an effective storage for predicates.

The event predicates extracted in our data model are generic in the video-surveillance domain and generally supported by existing systems (e.g., Lyons et al., 2000; Stringa and Regazzoni, 1998; Brodsky et al., 2001). Object-based querying is also implemented in most of the existing systems (e.g., Kim and Hwang, 2002a; Hampapur et al., 2005) to some extent. The queries based on color are considered in some systems (e.g., Stringa and Regazzoni, 2000, 1998; Regazzoni et al., 1998); however, the existing query models are not rich enough to support low-level features about the objects. We extract object-based low-level features and provide a scenario-based querying scheme for complex querying, also including color and shape descriptors of the objects (e.g., Queries 5 and 6).

Our query model supports scenario-based querying, which allows temporal ordering of event predicates, as well as of object information, based on low-level feature descriptors and directional predicates. An inverse-querying scheme is also provided to help the offline inspection process. In addition, view-based querying is available, which augments the query expressiveness of our model. Hence, one of the main differences between our querying mechanism and that of MILS (Hampapur et al., 2005) is that we provide mechanisms for defining specialized queries in a more expressive manner (e.g., Queries 7–10).

Our query-specification scheme is based on VSQL, which provides an intuitive way of expressing logical predicates. In VidMAP (Shet et al., 2005), the authors provide high-level rules for a couple of query types. In contrast, we provide low-level feature descriptions and directional predicates, as well as temporal information about events to enrich the supported query types. Our query-processing system also supports view-based querying. Compared to the existing systems, the proposed scenario-based query-processing approach provides support for a wide range of query types and facilitates after-the-fact analysis.

8. Conclusion

We propose a scenario-based query-processing system for video surveillance archives, which provides a mechanism for effective offline inspection. A scenario-based query is specified as a sequence of event-based subqueries that can be enriched with object-based low-level features and directional predicates. With the help of the proposed inverted tracking technique, the system provides support for view-based query handling using a fixed view-based representation of the content. Our system also provides support for inverse querying as a specialized tool for

after-the-fact activity analysis. We developed Video Surveillance Query Language, which is specialized for scenario-based querying. We present a rule-based query-processing module to provide not only an efficient processing mechanism for scenario-based querying, but also a flexible medium for external predicate definition, which allows the system to be tailored to various domains. We also present a visual query interface to facilitate the query specification process.

The performance of our scenario-based querying system was evaluated through a set of experiments. Since the performance of the overall querying scheme strictly depends on the meta-data extraction process, we also carried out experiments on these pixel-level and region-level methods. It was shown through these experiments that the querying support of our system is highly effective and has expressive power in offline inspection.

To further increase the capabilities and the expressive power of our query-processing system, we are planning to implement the negation operator for scenario-based query results and for variables in the query-specification process. There might be specific uses for this negation operator for the inspectors. In the query-specification interface, we are planning to include a natural language parser that can learn domain-specific keywords *a priori*. This will simplify the query-specification process and enrich the semantic quality of the results.

Appendix A. Grammar for Video Surveillance Querying Language (VSQL)

```
/* main query string */
<query> ::= select <target> from <range> [where <query-condition>] ';'

<target> ::=      segments |  /* retrieve video sequences/intervals */
                  frames |   /* retrieve frames for event queries */
                  events |  /* inverse querying w.r.t. events */
                  objects |   /* inverse querying w.r.t. objects */
                  most-popular-path | most-abnormal-region |  /* view-based queries */
                  <objectlist>  /* retrieve object(s) */
<objectlist> ::= [<objectlist> ','] <objlabel>
<range> ::= all | <videolist>
<videolist> ::= [<videolist> ','] <vid>
<query-condition> ::= [<object-assignment-list> and] <scenario> | <inverse-condition>
<object-assignment-list> ::= [<object-assignment-list> ','] <object-assignment>
<objectassignment> ::= <objlabel> <objoperator> <objcondition>
<scenario> ::= [<scenario> [<timegap>]] <event-condition>
<event-condition> ::= <single-object-event-condition> | <multi-object-event-condition>
<inverse-condition> ::= inverse '(' <intvalue> ',' <intvalue> ')'

/* event query conditions */
<single-object-event-condition> ::= <single-object-event-label> '(' <objlabel> ')'
<single-object-event-label> ::= enter | leave | stop | stop-and-go | move-<direction>
<multi-object-event-condition> ::= <multi-object-event-label>
 '(' <multi-object-condition> ')'
<multi-object-event-label> ::= crossover | deposit | pickup | move-together | approach | depart
<multi-object-condition> ::= <objlabel> ',' <objlabel> [',' <direction>]
<direction> ::= west | east | north | south | northeast | southeast | northwest | southwest

/* object conditions */
<objcondition> ::= objdata '(' <objdesclist> ')'
<objdesclist> ::= [<objdesclist> ','] <objdesc>
<objdesc> ::= <classdesc> | <colordesc> | <shapedesc>
<classdesc> ::= class '=' <classvalue>
<colordesc> ::= color '=' <colorlabel>
<shapedesc> ::= shape '=' <shapelabel>
<colorlabel> ::= red | green | blue | yellow | white | black | orange | violet
<shapelabel> ::= box | cone | cylinder | sphere
<classvalue> ::= human | non-human | object-group

/* primitive types */
<intvalue> ::= (1-9)(0-9)*
<vid> ::= <intvalue>
<timegap> ::= <intvalue>
<objlabel> ::= (a-z)(A-Za-z0-9)*
<objoperator> ::= '=' | ``!=''
```

Appendix B. Conjunction and disjunction examples

Assume the results of two scenario-based queries are as follows:

$R_1 = \{(1, \{[50, 325], [447, 740]\}), (3, \{[25, 285], [780, 940]\})\}$
and

$R_2 = \{(1, \{[200, 475], [520, 700]\}), (2, \{[120, 340]\})\}$.
If we apply the conjunction and disjunction algorithms in Figs. 6 and 7 to obtain $R_C = R_1 \wedge R_2$ and $R_D = R_1 \vee R_2$,

$R_C = \{(1, R_{1,1} \wedge R_{2,1})\}$.
Since $[447, 740] \supset [520, 700]$,

$R_{1,1} \wedge R_{2,1} = \{[447, 740]\}$.
Hence,

$R_C = \{(1, \{[447, 740]\})\}$
and

$R_D = \{(1, \{[50, 325], [447, 740], [200, 475]\}), (3, \{[25, 285], [780, 940]\}), (2, \{[120, 340]\})\}$.

Appendix C. Sample facts-base

A snapshot of the facts-base for a sample video is given to clarify the understanding of the object information and corresponding event information. An example fact denoting our interval extension notation (fact at line 4). This section shows the extraction of a deposit event such that an object enters the scene, then split occurs and one object, classified as human, continued its motion, whereas the other object stopped.

```
          /* object information facts */
1         object-info(1, obj001, human, blue, box, 12, 5).
2         object-info(1, obj001, human, blue, box, 13, 6).
3         object-info(1, obj001, human, blue, box, 14, 6).
4         object-info(1, obj002, non-human, white, box, [14,17], 6).
5         object-info(1, obj001, human, blue, box, 15, 7).
6         object-info(1, obj001, human, blue, box, 16, 8).

          /* event information facts */
7         event-info(1, enter, 12, 5).
8         event-info(1, enter, 14, 6).
9         event-info(1, deposit, 15, 7).
10        event-info(1, stop, 16, 6).
11        event-info(1, leave, 17, 8).

          /* scenario-based meta-data */
12        enter(1, obj001, 12, 5).
13        move-east(1, obj001, 13, 6).
14        enter(1, obj002, 14, 6).
15        move-east(1, obj001, 15, 7).
16        deposit(1, obj001, obj002, west, 15, 6).
17        move-east(1, obj001, 16, 8).
18        stop(1, obj002, 16, 6).
19        leave(1, obj001, 17, 8).
...
```

References

Bobick, A., Davis, J., 2001. The recognition of human movement using temporal templates. IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (3), 257–267.

Bogomolov, Y., Dror, G., Lapchev, S., Rivlin, E., Rudzsky, M., 2003. Classification of moving targets based on motion and appearance. In: Proceedings of the British Machine Vision Conference, vol. 2, pp. 429–438.

Brodsky, T., Cohen, R., Cohen-Solal, E., Gutta, S., Lyons, D., Philomin, V., Trajkovic, M., 2001. Visual surveillance in retail stores and in the home. In: Video-Based Surveillance Systems Computer Vision and Distributed Processing. Kluwer Academic Publishers, Dordrecth, pp. 51–65.

Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., Wixson, L., 2000. A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Carnegie Mellon University, The Robotics Institute, 2000.

Dönderler, M., Ulusoy, Ö., Güdükbay, U., 2004. Rule-based spatio-temporal query processing for video databases. VLDB Journal 13 (3), 86–103.

Duong, T., Bui, H., Phung, D., Venkatesh, S., 2005. Activity recognition and abnormality detection with the switching hidden semi-markov model. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 838–845.

Duque, D., Santos, H., Cortez, P., 2006. The OBSERVER: an intelligent and automated video surveillance system. In: Campilho, A., Kamel, M. (Eds.), Lecture Notes in Computer Science, vol. 4141. Proceedings of the International

Conference on Image Analysis and Recognition (ICIAR). Springer, Berlin, 2006, pp. 898–909.
Durak, N., Yazıcı, A., George, R., 2007. Online surveillance video archive system. In: Cham, T.-J., et al. (Eds.), Lecture Notes in Computer Science (LNCS), vol. 4351. Proceedings of the 13th International Multimedia Modeling Conference (MMM 2007), Part I. Springer, Singapore, pp. 376–385.
Elgammal, A., Harwood, D., Davis, L., 1999. Non-parametric model for background subtraction. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition, Workshop on Motion, Ft. Collins, CO, USA.
Fawcett, T., 2006. An introduction to ROC analysis. Pattern Recognition Letters 27, 861–874.
Gutchess, D., Trajkovic, M., Cohen-Solal, E., Lyons, D., Jain, A., 2001. A background model initialization algorithm for video surveillance. In: Proceedings of the International Conference on Computer Vision, Vancouver, Canada, pp. 733–740.
Hamid, R., Johnson, A., Batta, S., Bobick, A., Isbell, C., Coleman, G., 2005. Detection and explanation of anomalous activities: representing activities as bags of event n-grams. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1031–1038.
Hampapur, A., Brown, L., Connell, J., Lu, M., Merkl, H., Pankanti, S., Senior, A., Shu, C., Tian, Y., 2005. Multi-scale tracking for smart video surveillance. IEEE Signal Processing Magazine 22 (2), 38–51.
Haritaoglu, İ., Harwood, D., Davis, L., 2000. W4: Real-time surveillance of people and their activities. IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8), 809–830.
Hu, W., Tan, T., Wang, L., Maybank, S., 2004. A survey on visual surveillance of object motion and behaviors. IEEE Transactions on Systems, Man, and Cybernetics, Part C—Applications and Reviews 34 (3), 334–352.
Jain, A., Vailaya, A., 1996. Image retrieval using color and shape. Pattern Recognition 29 (8), 1233–1244.
Kim, C., Hwang, J., 2002a. Fast and automatic video object segmentation and tracking for content-based applications. IEEE Transactions on Circuits and Systems for Video Technology 12 (2), 122–129.
Kim, C., Hwang, J., 2002b. Object-based video abstraction for video surveillance systems. IEEE Transactions on Circuits and Systems for Video Technology 12 (12), 1128–1138.
Kim, K., Chalidabhongse, T., Harwood, D., Davis, L., 2005. Real-time foreground/background segmentation using codebook model. Real-time Imaging 11 (3), 172–185.
Li, L., Huang, W., Gu, I., Tian, Q., 2003. Foreground object detection from videos containing complex background. In: Proceedings of the ACM International Conference on Multimedia. ACM Press, Berkeley, CA, USA, pp. 2–10.
Lyons, D., Brodsky, T., Cohen-Solal, E., Elgammal, A., 2000. Video content analysis for surveillance applications. In: Proceedings of the Philips Digital Video Technologies Workshop, 2000.
Nascimento, J., Marques, J., 2006. Performance evaluation of object detection algorithms for video surveillance. IEEE Transactions on Multimedia 8 (4), 761–774.
Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006), New York, June 2006. Benchmark Data 〈http://www.cvg.rdg.ac.uk/PETS2006/data.html〉.
Paschos, G., Valavanis, F.P., 1999. A color texture based visual monitoring system for automated surveillance. IEEE Transactions on Systems, Man, and Cybernetics, Part C—Applications and Reviews 29 (2), 298–307.
Regazzoni, C., Sacchi, C., Stringa, E., 1998. Remote detection of abandoned objects in unattended railway stations by using a DS/CDMA video surveillance system. In: Regazzoni, C., Fabri, G., Vernezza, G. (Eds.), Advanced Video-Based Surveillance System. Kluwer, Boston, MA, pp. 165–178.
Regazzoni, C., Ramesh, V., Foresti, G., 2001. Scanning the issue/technology: special issue on video communications, processing, and understanding third generation surveillance systems. Proceedings of the IEEE 89 (10), 1355–1367.
Rivlin, E., Rudzsky, M., Goldenberg, R., Bogomolov, U., Lepchev, S., 2002. A real-time system for classification of moving objects. In: Proceedings of the International Conference on Pattern Recognition, vol. 3, pp. 688–691.
Şaykol, E., Sinop, A., Güdükbay, U., Ulusoy, Ö, Çetin, E., 2004. Content-based retrieval of historical Ottoman documents stored as textual images. IEEE Transactions on Image Processing 13 (3), 314–325.
Şaykol, E., Güdükbay, U., Ulusoy, Ö., 2005a. A database model for querying visual surveillance by integrating semantic and low-level features. In: Candan, K., Celentano, A. (Eds.), Lecture Notes in Computer Science, vol. 4457. Proceedings of 11th International Workshop on Multimedia Information Systems (MIS'05). Sorrento, Italy, 2005, pp. 163–176.
Şaykol, E., Güdükbay, U., Ulusoy, Ö., 2005b. A histogram-based approach for object-based query-by-shape-and-color in multimedia databases. Image and Vision Computing 23 (13), 1170–1180.
Shet, V., Harwood, D., Davis, L., 2005. Vidmap: video monitoring of activity with prolog. In: Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance, pp. 224–229.
Stringa, E., Regazzoni, C., 1998. Content-based retrieval and real time detection from video sequences acquired by surveillance systems. In: Proceedings of the International Conference on Image Processing ICIP, pp. 138–142.
Stringa, E., Regazzoni, C., 2000. Real-time video-shot detection for scene surveillance applications. IEEE Transactions on Image Processing 9 (1), 69–79.
Swain, M., Ballard, D., 1991. Color indexing. International Journal of Computer Vision 7 (1), 11–32.
Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2007), Rio de Janeiro, Brazil, October 2007 〈http://www.pets2007.net〉.
Thirde, D., Borg, M., Valentin, V., Barthelemy, L., Aguilera, J., Fernandez, G., Ferryman, J., Bremond, F., Thonnat, M., Kampel, M., 2006. People and vehicle tracking for visual surveillance. In: Proceedings of the Sixth IEEE International Workshop on Visual Surveillance, pp. 169–176.
Xiang, T., Gong, S., 2006. Beyond tracking: modelling activity and understanding behaviour. International Journal of Computer Vision 67 (1), 21–51.
Xiang, T., Gong, S., 2008. Incremental and adaptive abnormal behaviour detection. Computer Vision and Image Understanding 111 (1), 59–73.
Zhong, H., Shi, J., Visontai, M., 2004. Detecting unusual activity in video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 819–826.