A Scenario-Based Video Surveillance Data Modeling and Querying System

Ediz Şaykol, Uğur Güdükbay, and Özgür Ulusoy
Department of Computer Engineering, Bilkent University
06800 Bilkent, Ankara, Turkey
{ediz,gudukbay,oulusoy}@cs.bilkent.edu.tr

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, event models and content-based access to surveillance video have significant semantic gaps remaining unfilled. In this paper, we propose a scenario-based querying and retrieval model for video surveillance archives. In our model, a scenario is specified as a sequence of event predicates, that can be enriched with object-based low-level features. Our model also provides support for inverse querying, which can be considered as a tool for after-the-fact type of activity analysis. We have devised a visual query specification interface to ease the scenario-based query specification process. It is shown through performance experiments that our system has an effective expressive power and satisfactory retrieval accuracy in video surveillance.

This work is supported in part by European Commission 6^{th} Framework Program MUSCLE Network of Excellence Project with grant number FP6-507752, and Scientific and Technical Research Council of Turkey (TUBITAK) with grant number EEEAG-105E065.

I. Introduction

In a traditional surveillance system, human operators monitor multiple guarded environments simultaneously to detect, and possibly prevent, a dangerous situation. As a matter of fact, human perception and reasoning are limited to process the amount of spatial data perceived by human senses. These limits may vary depending on the complexity of the events and their time instants. The acceleration in communication capabilities and automatic video processing techniques, and the reasonable cost of the technical devices have increased the interest in video surveillance applications in the recent years. As a consequence, the capabilities of the human operators have been augmented through the use of these applications.

As argued in [1], a complete automated video surveillance system should support both real-time alarm generation and database support for offline inspection. Human-intervention for initialization is acceptable to some extent. For both alarm generation and offline inspection, the input video stream should be processed to analyze the actions. This processing generally starts with a background/foreground subtraction ([2], [3], [4]) or temporal template-based methods ([5], [6]) to detect moving objects, and continues with event (or activity) recognition by tracking these objects. The suspicious ones among the events are generally reported to the operator and/or stored in a database for a later inspection. As far as the database part is concerned, the researchers generally assume simple structures for events and objects. The event descriptors generally contain time information and the salient object labels acting in the event. Object-level indexing is not as frequent as event-level indexing. To the best of our knowledge, no video surveillance system has been introduced in the literature that is embedding object-based low-level features (e.g., color, shape) to the scenario-based querying and retrieval module.

We propose a system that provides an integrated environment for querying video surveillance archives by semantic (event-based) and low-level (object-based) features. The architecture of our surveillance system is shown in Figure 1. Users of the system are able to specify event-based scenario queries. The system returns the segments where the events in the scenarios occurred. The preliminaries of the meta-data extraction process are described in [7]. A real-time alerting module is also available in our system, but in this paper, our main focus is on the querying and retrieval capabilities of the system. The query processing component provides support for scenario-based, event-based, and object-based queries, where the low-level object features can be

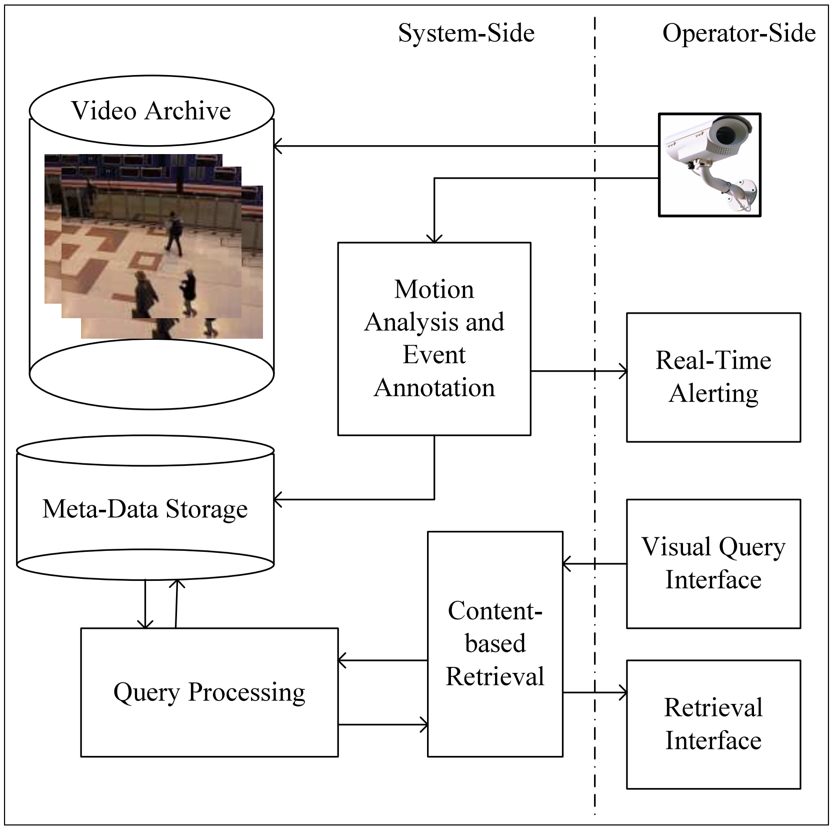


Fig. 1. *The overall architecture of our system. The queries are handled by the query processing module, which communicates with both the meta-data store and the content-based retrieval module. The database contains event and object features extracted by automated tools. These automated tools also trigger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

used to improve querying effectiveness. Our system also supports *inverse querying* that can be used as a tool for activity analysis in various domains, e.g., video forensics. A query language, which we call Video Surveillance Query Language (VSQL) [7], is proposed to express queries in an SQL-based manner. A visual query specification interface has also been developed, which can be considered as the visual counterpart of VSQL.

The rest of the paper is devoted to the detailed description of the approaches we propose for video surveillance querying and retrieval. Section II discusses related studies along with a comparison with our contributions. Section III presents the data model we developed to achieve querying and retrieval of surveillance videos. Query processing capabilities of system are described in Section IV. Section V discusses the retrieval component of our system, along with the interfaces designed for visual query specification and result presentation. Performance experiment results for our techniques are presented in Section VI. Finally, Section VII concludes the paper with some future directions.

II. Related Studies

Most of the video surveillance systems that appear in the literature focus on object detection and tracking. Extracted event information based on object tracking and shot detection is generally stored in a database and exhaustively searched when a query based on event information is submitted (e.g., [8]). From another perspective, retrieving the video sequences related to a generated alarm is the basic way of querying the system. To this end, Stringa and Regazzoni describe complex query types including color and/or shape properties of the salient objects [9].

Video Surveillance and Monitoring (VSAM) system presented in [2] is one of the complete prototypes for object detection, object tracking and classification, as well as 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 be robust for detection of stop-and-go type of motions. The background maintenance scheme we employ is similar to that of VSAM. However, in our framework, the extracted background is used not only for event annotation but also object tracking.

Stringa and Regazzoni ([9], [10], [11]) propose a real-time surveillance system employing semantic video-shot detection and indexing. In that system, lost objects are detected by the help of temporal rank order filtering. The *interesting* video shots are detected by a hybrid approach based on low-level (color) and semantic features. Retrieving all the clips related to an alarm is the basic way of querying the system. The authors also mention about more complex query types including color and/or shape properties of the objects identified as dangerous. In our framework, we extract object-based low-level features, and provide a scenario-based querying scheme for complex querying including color and shape descriptors of the objects.

Haritaoğlu et al. [5] propose a model for real-time analysis of people activities. Their model uses a stationary camera and background subtraction to detect the regions corresponding to person(s). Their system, called W^4, uses shape information to locate people and their body parts (head, hands, feet, and torso). The system operates on monocular gray-scale video data, and no color cues are used. The creation of models of the appearance of person(s) helps the tracking process through people interaction (e.g., occlusions), and simultaneous activities of multiple people. The system uses a statistical background model holding bimodal distribution of intensity at each pixel to locate people. The system is capable of detecting single person, multiple persons, and multiple person groups in various postures.

In [12], [13], an object-based video abstraction model is proposed. In that work, the authors employ a moving-edge detection scheme 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 keyframes. When a change occurs in the number of moving regions, the current frame is declared as a keyframe indicating that an important event has occurred. This scheme also facilitates the detection of important events. If the number of moving objects remains the same for the next frame, then a shape-based change detector is applied to the consecutive frames. A frame-based similarity metric is also defined to detect the distance between two frames. Our framework employs the strategy of moving object counting mentioned in [12] with rule-based extensions to help the event annotation process.

Rivlin et al. [14] propose a real-time system for moving object detection, tracking, and classification where the video stream is coming from a static camera. Effective background initialization and background adaptation techniques are employed for a better change detection. Target detection phase also benefits from a color table representation of object data. The detected moving objects are classified into *human*, *animal*, and *vehicle* classes 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 is combining appearance and motion-based features to increase the accuracy [15].

IBM's MILS (MIddleware for Large Scale Surveillance) [16] 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 also employs relevance feedback and data mining facilities to increase the effectiveness. The main difference between our querying mechanism and that of MILS is the fact that we provide scenario-based querying support for video surveillance systems, which intentionally gives more opportunities for defining specialized queries in a more expressive and effective manner.

Lyons et al. [8] 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 retrieval. They adapt a non-parametric background subtraction approach based on [17]. VCA discriminates people from objects and the main events recognized are *entering scene*, *leaving scene*, *splitting*, *merging*, and *depositing/picking-up*. The retrieval component enables users to retrieve video sequences based on event queries. The event categories are very similar to those we use in our framework. However, as an additional feature, our framework also enables object-based querying that can be refined by providing low-level and/or directional descriptors.

Brodsky et al. [18] designed a system for indoor visual surveillance, especially to be used in the retail stores and in the houses. They assume a stationary camera and use background subtraction. A list of events that the object participates is stored for each object where the events are simply *entering*, *leaving*, *merging*, and *splitting*. One of the distinctions of this system and our framework is the use of color feature. In the system described in [18], the color feature of the objects is mainly used for reassigning labels for moving connected components, whereas we also provide object-based querying based on color and/or shape features.

III. Data Model

Data model of our video surveillance system contains object-level information at the lowest level. The color (average and dominant), velocity, aspect-ratio of the Minimum Bounding Rectangle (MBR), and contour information is used for describing (moving) objects, which are classified as *human* or *non-human*. At a higher level, primitive object actions are annotated, e.g. stop, move, enter, etc. At the highest level, conceptual event predicates are detected to identify activities, e.g. crossover, deposit, approach, etc. This information extraction scheme, shown in Figure 2, is explained in the following parts.

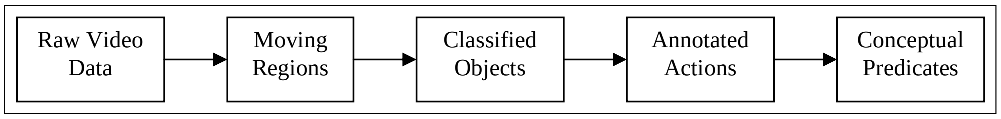


Fig. 2. *The semantic flow in information extraction.*

A. Extraction of Moving Regions

We employ an adaptive background maintenance scheme to extract the moving regions (or moving parts) in a video frame, similar to the one proposed in [2]. The adaptive background subtraction technique is combined with a 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 $M_f(x, y)$ 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 in Eq. 1:

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

The (moving) pixel intensities that are larger than the background intensities ($B_f(x, y)$) are used to fill 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 at the current frame. A statistical background and threshold maintenance scheme is employed as presented in Eqs. 2-5:

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

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

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

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

where α is the learning constant, and the constant k is set to 5 in Eq. 5. As discussed in [2], $B_f(x,y)$ is analogous to 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.

We also employ view-based motion tracking approach similar to Motion History Image (MHI) technique proposed in [6]. MHI provides detecting and tracking motion parameters of the moving regions. These parameters are basically the structure and the orientation of the motion. In an MHI, the pixel intensity is encoded as a function of the temporal history of the motion at that pixel, where more recently moving pixels are brighter. Here, the MHI (*MHI*$_f(x,y)$) of a frame f is constructed by the update rule in Eq. 6:

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

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 [6]).

This motion history tracking technique not only tracks the motion structure of the moving regions but also gives the orientation of them, which leads us to keep track of the trajectories of the regions easily. Moreover, to improve the region tracking capability, we employ a grid-based technique to store the object-presence history, which we call *Inverted Tracking*.

In our Inverted Tracking scheme, we have divided the video frame $I(x,y)$ into 16 cells corresponding to 4 sub-divisions in x and y directions. Figure 3 illustrates the inverted tracking technique on a sample frame. While computing region appearance within a cell, we consider the center-of-mass (c_m) of the region. The $c_m = (x_{c_m}, y_{c_m})$ is computed as in Eq. 7:

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

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

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.

In the inverted tracking scheme, the number 16 is selected not only to increase the computational and storage costs of the system, but also to provide effective positional object-history

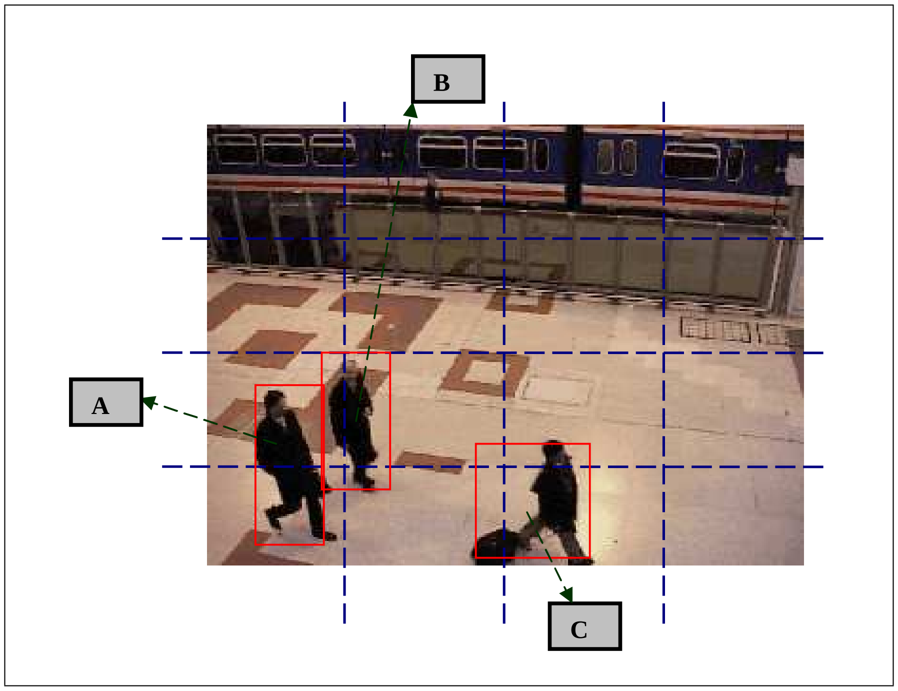


Fig. 3. *The inverted tracking scheme illustrated on a sample video frame [22]. The boxes A, B, and C hold for the structures keeping track of object history lists along with the last frame that the object has appeared within that cell.*

tracking. We have also validated this selection by trying values 4 and 64 instead of 16. As an example of the use of the inverted tracking scheme, in a stop-and-go type of action, while the region is not moving, it becomes the part of the background within a couple of frames, however, when it starts its motion again, we can detect that the previously-stopped region is moving again by tracing back the object history list belonging to that cell of the grid.

B. Classification of Objects

We categorize the (moving) regions into three classes: *human*, *non-human*, and *object-group*. The classification is limited with these three types in order to be generic. The low-level color and shape features, and the aspect-ratio of the MBRs of the regions are used for classification. The low-level features are stored in normalized *color* and *shape vectors* [19]. The color vector stores distance-weighted intensity values to take the local neighborhood of a pixel into account. The shape vector is a composition of *angular* and *distance* spans ([19], [20]). The distance and angular spans of a region are computed with respect to the region's center of mass (c_m). As shown in [19], these encoding schemes are invariant under scale, rotation, and translation, and have an effective expressive power.

We employ a neural network based classification algorithm, the inputs of which are the color and shape feature vectors, and the aspect-ratio of the region MBR. Here, we use the temporal averages of these vectors. The classification algorithm outputs percentages for class values (e.g., 47% human, 34% non-human, 19% object-group), and the moving region is classified by the class with the highest percentage value. When an object-group is detected, we apply k-means clustering to the region, and try to divide the object-group into single objects, which are also classified by the algorithm if splitting an object-group is successful. The initial weights of this classification scheme is fed by a set of manually trained object samples. The weights are updated during the classification process, which improves the accuracy.

C. Annotation of Events

Having classified the moving objects in a video sequence, we annotate the object actions automatically. Basic single object actions detected by our system are *enter*, *leave*, *stop*, *stop-and-go*, and 8 directional forms of *move* coupled with the directional predicates (see `<direction>` in Appendix A). The orientation of a moving region suggested by our tracking algorithm is used to directly annotate move type of actions. When an object stops and moves again later, having detected the object's next move, we annotate the action as stop-and-go type, thanks to our inverted tracking scheme. The other types of the single object actions are rather easier to detect by the tracking algorithm.

Multi-object actions are also annotated in our system, and they are *approach*, *depart*, *deposit*, *pickup*, *crossover*, and *move together*. The first two can be identified by tracking the Euclidean distance between two objects in consecutive frames. If the distance in previous frame is greater/smaller than the distance in the current frame, these two objects are approaching/departing to/from each other. The remaining types are relatively harder to detect. Figure 4 illustrates the methods that we have used to detect these events.

Counting the number of moving objects also gives an important clue for the annotation of events, since at the time of an event, the number of moving objects changes. We utilize the event annotations to hold this number throughout the video processing process. The event annotation algorithm also handles directional relations among the moving objects by the help of the motion

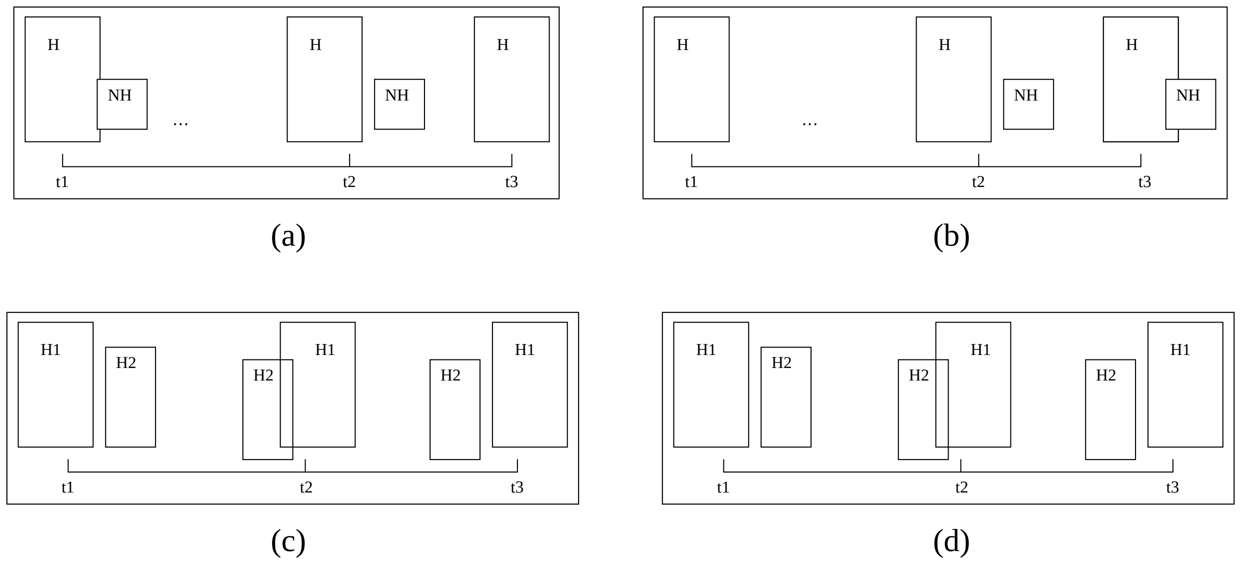


Fig. 4. *Annotation of events. (a) The clustering algorithm detects 2 objects at t1, and they are classified as 1 human (H), and 1 non-human (NH). Then, they are detected as 2 single objects at t2, and when NH stops at t3,* **deposit** *is identified. (b) Similar to (a),* **pick up** *is identified. (c) H1 and H2 are classified as human at t1, and an object group is detected at t2, and cannot be clustered. By tracking the orientations of the objects within t1-t3 time interval,* **crossover** *is identified. (d) Similar to (c),* **move-together** *is identified except that here the humans move in the same directions.*

tracking scheme. To elaborate on our event annotation process, consider the following sequence of operations:

- An object-group OG_1 is identified. The annotation scheme throws *enter(OG_1)*.
- Clustering is applied on OG_1. If object splitting is not successful, *move* predicate is thrown with the valid direction for OG_1.
- If OG_1 can be split into 2 objects O_1 and O_2, these 2 objects are also classified.
  - (a) If O_1 is human and O_2 is non-human, it is a potential for *deposit* event.
  - (b) If both O_1 and O_2 are human, the event could be both *crossover* or *move-together* depending on the orientations of the humans' motion.
  - (c) If both O_1 and O_2 are non-human, the event could be of *move-together* type assuming that these two objects are thrown by a human not in the field-of-view of the camera.
  - Continuing for (a), 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 since the position, velocity and orientation of the moving objects are tracked.

- The directional relations are also identified for multi-object actions to improve the accuracy of retrieval (e.g., *deposit*(O_1, O_2, *south*) is extracted).

The event annotation process for the other types of events are handled in a similar manner. The object-based information (i.e., class value, color and shape information, orientation, average velocity) is stored along with event-based information (i.e., event-label, acting objects, frames) for querying and retrieval purposes. For real-time tracking, alerts for the operator can be triggered when suspicious ones of these types of events are detected.

IV. Query Model

One of the most important tasks in automated video surveillance is query processing. Basically, textual searches for event queries are supported in the existing systems. Some systems support object queries as well to some extent. It is observed that there might be a need for enhancing object queries with color and/or shape descriptions. In addition to this enhancement, allowing directional relation specifications among objects within suspicious events might be helpful in some domains.

The main contribution of our querying approach is providing support to a wide range of event and object queries, which can also be expressed in a scenario-based manner. A classification of the types of queries handled in our system can be listed as follows:

- single object queries,
  - object entering/leaving scene,
  - object stopping/stopping and restarting to move,
  - object moving in 8 basic directions,
- multi-object queries,
  - object depositing/object picking up,
  - objects crossing over/objects moving together,
- inverse queries.

We have devised an SQL-based querying language, which we call Video Surveillance Query Language (VSQL), to provide support for integrated querying video surveillance databases by semantic and low-level features. The grammar for VSQL is given in Appendix A. Semantic sub-queries contain 12 single object event types and 6 multi-object event types, which can be combined to form more complex queries. A *timegap* value can be specified between event conditions. This timegap value has more meaning when specified between the same kind of event types (i.e., a pair of single object events or a pair of multi-object events). Descriptive keywords can be supplied for 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 human operators inspecting (i.e., querying) the videos would like to express these features literally among a set of pre-specified labels. We are also planning to include query-by-example and query-by-sketch type of query specification strategies in the later stages of our work.

Our system provides support for scenario-based queries which are not easy to handle in real-time systems. Scenario-based querying by an effective set of semantic and low-level sub-queries improves the retrieval quality and decreases the time of 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, e.g., after-the-fact analysis in video forensics. VSQL also provides support for inverse querying, which returns the existence of a list of objects or a list of events within a certain time interval to increase the querying effectiveness.

Based on the observation that rule-based modeling is effective for querying video databases [21], a rule-based model has been designed for querying surveillance videos. The submitted queries are sent to our inference engine, Prolog, which processes the meta-data based on a set of rules. This meta-data is the extracted event and object information stored in the knowledge-base. Prior to this rule-based processing, the submitted query string is parsed using a lexical analyzer. Figure 5 shows the flow of execution in the query processing steps.

A. Scenario-Based Querying

A scenario query consists of a sequence of single object and/or multi-object event sub-queries. Satisfying a scenario query means that every event sub-query in the scenario has to exist in a query result. Hence, the results of a scenario is a set of intervals, where each event sub-query exists in every interval element in the result set.

Definition 2 (Query Result of a Scenario): A query result $R_{S,i}$ of a scenario S for a specific

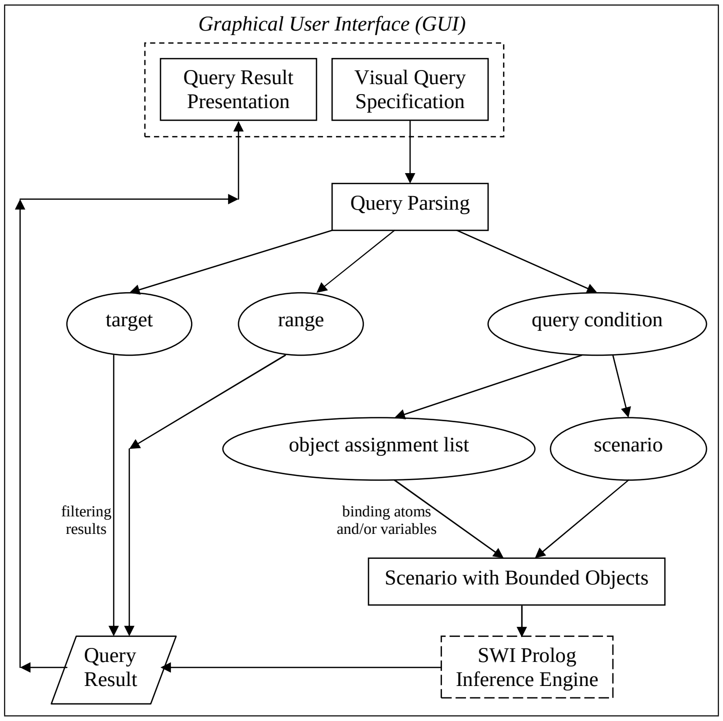

Graphical User Interface (GUI)
Query Result Presentation
Visual Query Specification
Query Parsing
target
range
query condition
object assignment list
scenario
filtering results
binding atoms and/or variables
Scenario with Bounded Objects
Query Result
SWI Prolog Inference Engine

Fig. 5. *The query processing flowchart. A VSQL query submitted to the system using GUI passes through the query parsing, binding, and processing steps, and finally the results are presented to the user.*

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

Corollary 1: $(i, R_{S,i})$-pair denotes the query for a scenario S for a video v_i.

Corollary 2: A query result R_S for a scenario S for all the videos in the archive is a list of pairs where $R_S = \{(i, R_{S,i})|\forall i$, where i denotes the index of videos in the archive$\}$.

Since the query result for a scenario is a list of intervals, any operation on the query results takes the intervals as their operands. The intervals 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. On the other hand, this implication is not correct for atomic intervals. The reason is that the condition associated with an atomic interval does not hold for all its subintervals. As far as this categorization is concerned, the query result intervals of a video surveillance scenario are atomic, hence they cannot be broken into parts. The definition of the logical conjunction and disjunction operations should take this fact into account. The following examples show the formulation of scenario-based queries in VSQL.

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)
```

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)
```

1) Conjunction: The logical conjunction operation is applicable to scenario-based queries. Assume two scenario-based queries and their results R_1 and R_2, that contain atomic intervals. Figure 6 presents the pseudo-code for obtaining the conjunction $R_C = R_1 \wedge R_2$ of the query results. The following example can be given to elaborate on this conjunction algorithm for scenario-based queries.
Assume that,

$$R_1 = \{(1, \{[50, 325], [447, 740]\}), (3, \{[25, 285], [780, 940]\})\}, \tag{8}$$

and

$$R_2 = \{(1, \{[200, 475], [520, 700]\}), (2, \{[120, 340]\})\}. \tag{9}$$

If we apply the algorithm in Figure 6,

$$R_C = \{(1, R_{1,1} \wedge R_{2,1})\},$$

Since $[447, 740] \supset [520, 700]$,

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

Hence,

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

```
$R_C = R_1 \wedge R_2$ 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.*

2) Disjunction: Similar arguments also hold for disjunction. We can assume two scenario-based queries and their results R_1 and R_2, which contain atomic intervals. Figure 7 presents the pseudo-code for obtaining the disjunction $R_D = R_1 \vee R_2$ of the query results. Assuming R_1 as in Eq. 8 and R_2 as in Eq. 9, the disjunction R_D can be determined as follows:

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

B. Event-based Querying

Our system provides support for event-based querying. The users may want to query all the occurrences of a single event of any single object or multi-object type within the archive. In this type of querying, the query results are returned as a set of frames, rather than as frame intervals, where the event specified in the query has happened.

Definition 3 (Query Result of an Event): A query result $R_{E,i}$ for an event-based query E for a specific video v_i is a list of frame numbers, where $R_{E,i} = \{f_E |$ the event E happens at $f_E\}$.

$R_D = R_1 \vee R_2$ 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.*

Corollary 3: $(i, R_{E,i})$-pair denotes the query for an event E for a video v_i.

Corollary 4: A query result R_E for an event E for all the videos in the archive is a list of pairs, where $R_E = \{(i, R_{E,i}) | \forall i$, where i corresponds to the index of videos 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)
```

The logical conjunction and disjunction operations can be applied on the query results directly, since a query result here is a set of frames.

C. Object-based Querying

Our system supports object-based queries in various ways. First, the existence or appearance of the objects can be queried. In this type of querying, the system returns the set of frames where the object has appeared. The low-level features (color, shape) and class values (human, non-human, etc.) 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 a result can be considered a simple set.

Query 4: 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 is a consequence of the *unification*[1] concept in Prolog. The system can return all the objects satisfying some pre-specified conditions, such as color, shape, and class. In this type of querying, the return type of the query is a list of object labels. 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 5: List the names of all the persons with a black coat.

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

D. Inverse Querying

An important type of querying in video surveillance is *inverse querying* especially for after-the-fact type activity analysis. The results of this query type is a list of objects appeared or a list of event labels occurred within an interval. Since the event labels are domain-specific, this type of querying would be more effective when domain-specific activity analysis is of concern.

Query 6: Which events occurred between frames 100 and 1000 in video 1?

```
select events
from   1
```

[1]Unification mechanism is used to bind the variables to atoms. All the atoms satisfying the predicate that includes the variable are returned as the result.

```
where inverse(100, 1000)
```

Query 7: Which objects appear between frames 100 and 1000 in videos 3,4?

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

V. Visual Query Specification

Our system includes a visual query specification interface, which enables users to express VSQL queries in an intuitive manner. This interface is easy-to-use and devised in a flexible manner, which makes it easily tailorable 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.

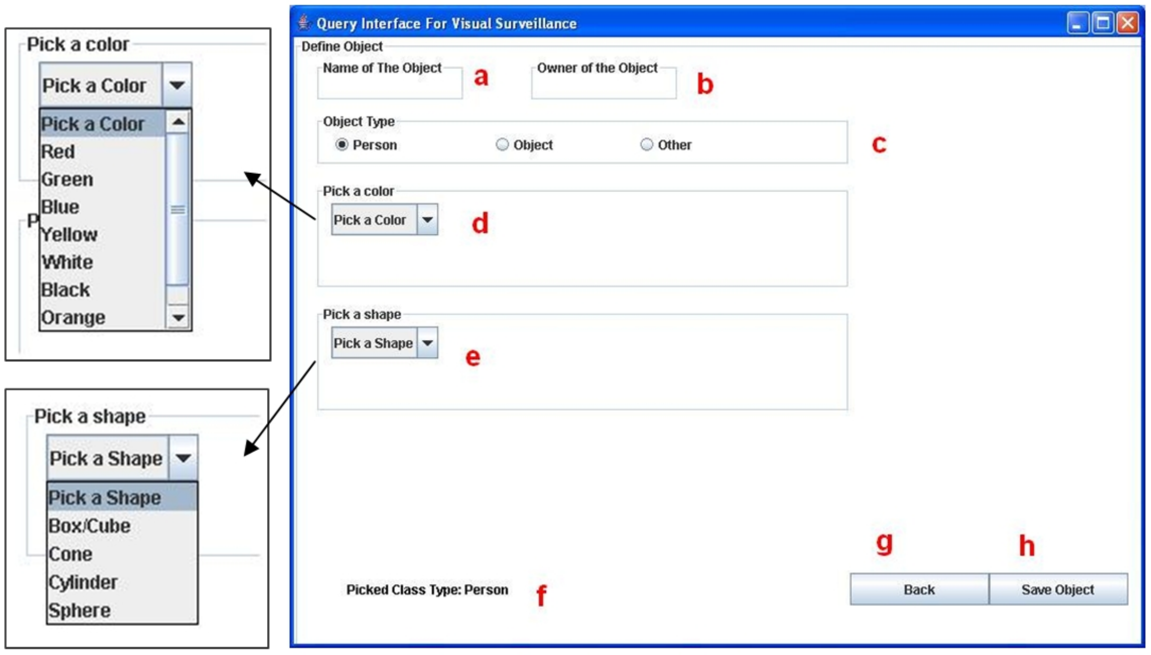
The visual query interface provides *object specification*, *event-specification*, and *scenario specification* facilities. It also utilizes an XML-based object repository to make the specified objects available for later use. The scenarios are expressed as a sequence of events on a drawing panel, where the order of the events can be altered to obtain various scenario combinations. The query results are presented to the user in a separate window where the user can browse the intervals in the result.

The object specification part of the visual query interface is very crucial not only for object-based queries, but also for event-based and scenario-based queries. Since descriptive keywords are used to describe low-level features of the human(s) and/or non-human(s), the interface is devised accordingly (see Figure 8).

The main contribution of our query model is scenario-based query support. Since a scenario is a sequence of events, the events forming the scenario have to be specified first. The specification of Query 8 through the interface is illustrated in Figure 9.

Query 8: List the segments from video 1 where two persons, one with a bag, meet together; then the person carrying a bag leaves the bag; the other person takes the bag; and then both persons leave.

```
select segment
from   1
```



Picked Class Type: Person

Fig. 8. *Object specification: (a) name; (b) name of the creator; (c) type; (d) color; (e) shape of the object; (f), (g), and (h) are the auxiliary interface elements.*

```
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. Scenario drawing panel can be considered as a timeline, which is a widely-used query specification technique for sequence-based data. In this panel, the events can be reordered and time gaps between events can be adjusted, which brings more flexibility in scenario-based query specification.

In the visual query interface, an *inverse query* can be specified by providing the type of the query (i.e., event, object), along with the list of videos, and the starting and ending frames of the interval. Query 6 can be specified by setting the type of the query to *events* and providing

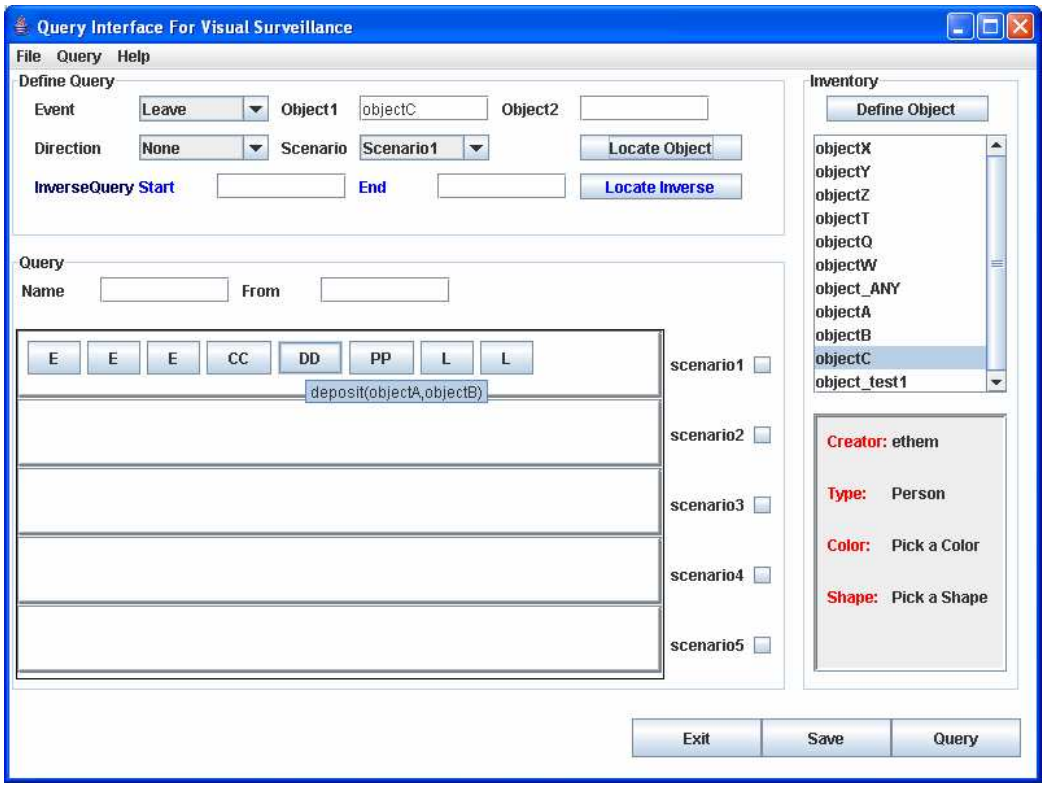


Fig. 9. *The specification of Query 8. The event sequence corresponding to the scenario is shown in the first row of the scenario drawing panel. The letters written on the boxes on the drawing panel, 'E', 'CC', 'DD', 'PP', and 'L' denote enter, crossover, deposit, pickup, and leave event predicates, respectively.*

the values, 1, 100, and 1000, respectively.

VI. Performance Experiments

The retrieval performance of our system is tested by providing experimental evaluations for the techniques used in the system. First, the performance of the pixel-level algorithms is evaluated. This is followed by the evaluation of object classification algorithms. The last set of experiments is conducted to evaluate our semantic annotation process. Since we utilized an SQL-based querying language and Prolog as our inference engine, scenario-based querying accuracy strictly depends on the performance of these three parts.

As our ground truth for the performance experiments, the benchmark data provided by PETS 2006 [22] is used along with manual annotations. We also collected a set of indoor monitoring

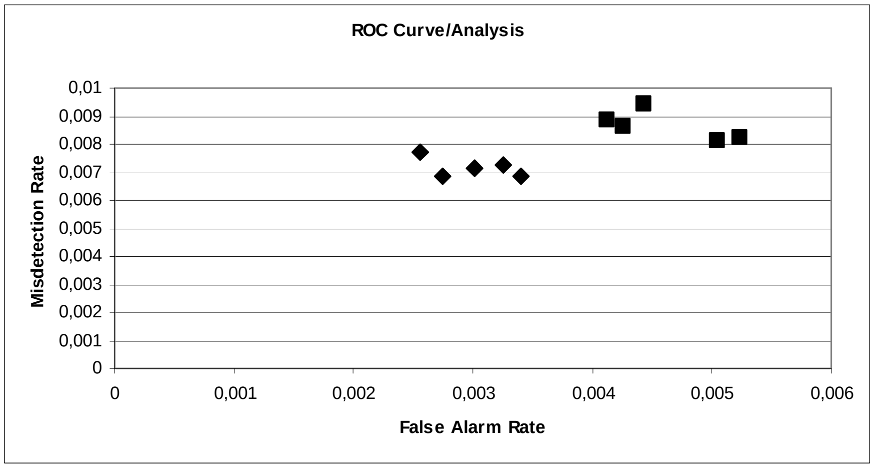


Fig. 10. *ROC curve/analysis of pixel-level algorithms. The learning constant* α *is varied from* 0.6 *and* 0.8 *by incrementing* 0.05 *steps.* ◆ *denotes a test configuration when* τ *is set to 2, whereas* ■ *denotes a test configuration when* τ *is set to 3.*

videos captured by a static camera within our university. We annotated these videos manually to validate the performance of our system. The average number of objects appearing per frame is 3.19 in our testbed.

A. Performance of the Pixel-level Processing

The pixel-level processing techniques that we employed are basically motion detection, color feature and shape feature extraction algorithms. For motion detection, we adapted widely-used background subtraction and maintenance schemes, and for motion tracking we utilized motion history images coupled with our inverted tracking algorithm. For color and shape features, we adapted distance-weighted color vector and shape vector, which is composed of angular span and distance span. These two feature vectors are shown to be effective in [19] through a wide range of performance experiments.

Evaluation of these algorithms is non-trivial and subjective. In [23], a discussion on performance evaluation of object-detection algorithms is given. Some of the standard measures for evaluating pixel-level techniques are *misdetection rate* and *false alarm rate*, and *receiver operating characteristic (ROC) curves* [24]. ROC curves are used to inspect the impact of a single parameter. While keeping all the parameters fixed, the value of the parameter of interest is modified. The misdetection and false alarm rates for the pixel-level algorithms used in our implementation are plotted in a graph for ROC analysis (Figure 10). Since ROC analysis requires ground truth evaluation for each parameter setting, here we focused on two parameters: α (the learning constant) and τ (temporal extent of motion).

Since in this paper our main emphasis is on scenario-based querying and retrieval, we selected the optimal set of system parameters (i.e. τ is set to 2, α is set to 0.7) for the remaining performance experiments.

B. Performance of Object Classification

The object classification algorithm that we used takes region data as the input and yields classification values as a percentage. The maximum of these values is chosen as the class value for the input region. We also utilized k-means clustering algorithm to divide an object group into single objects, if possible. Hence, our classification algorithm outputs three class values, which are human, non-human, and object-group.

In this evaluation, we computed two types of classification accuracies: *standard classification accuracy* and *frame-weighted classification accuracy*. The former is the percentage of correctly classified objects for each class label, whereas in the latter the number of frames that the object is classified correctly is taken into account while computing the percentage. Since we have employed the temporal averages of the feature vectors within the classification scheme, the frame-weighted accuracy analysis gives better results. Figure 11 presents the object classification analysis results. The lowest accuracy improvement in the frame-weighted analysis is for the human class, since many people enter and leave the scene. However, the classification accuracies for the other two classes improve significantly when the accuracies are frame-weighted.

C. Performance of Semantic Annotation

The detection of each manually annotated event type is inspected to evaluate the performance of our semantic annotation process. Since both scenario-based and inverse queries are processed by our inference engine, the retrieval accuracy of our querying performance can be evaluated by the event detection performance. To be more realistic, the percentage of the frames where the events are correctly identified is used as the event annotation accuracy 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 Figure 12.

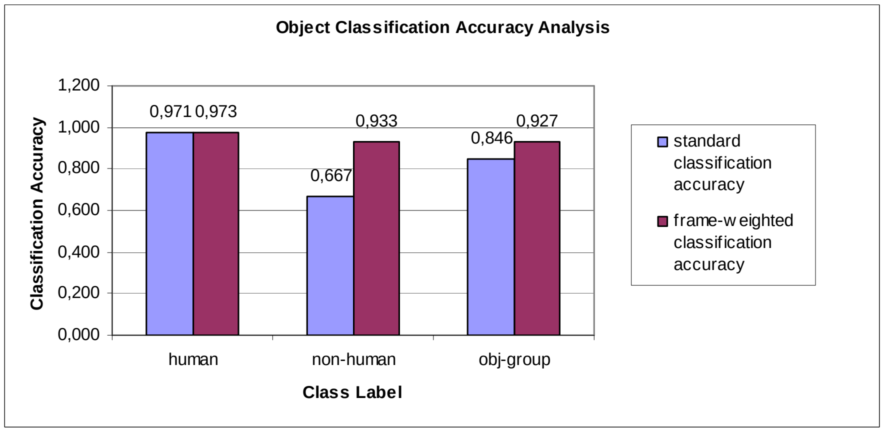


Fig. 11. *Object classification accuracy analysis.*

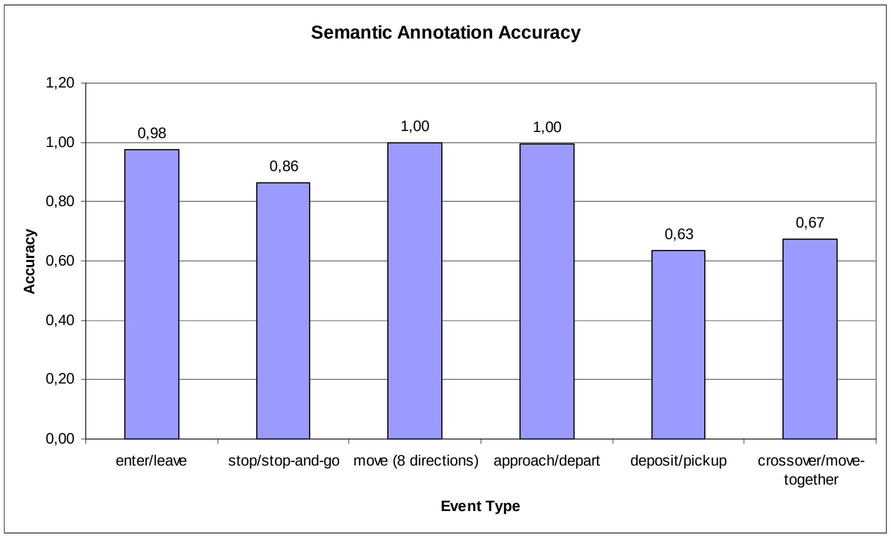


Fig. 12. *Semantic annotation accuracy analysis.*

As shown in Figure 12, the similar event types are grouped together to simplify the analysis. The accuracy is very high for enter/leave, move, and approach/depart type of events since they can be detected directly from region extraction. Since we accumulated the number of frames that an event is identified correctly into the accuracy value, multi-object event types have lower accuracy values than the other event types. This is simply from the fact that regions detected as object-groups cannot be split into single objects at the frame that they are detected.

Even if an event is not identified correctly for every frame that it has occurred, it might not affect the scenario-based querying and retrieval effectiveness, if the event is detected in at least one frame during the occurrence period. This observation is also true for inverse querying since the event is retrieved even it is detected in a single frame within the querying interval. Since we utilized Prolog as the inference engine in the querying process, 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.

VII. Conclusion

We propose a video surveillance system that provides support for event-based scenario queries and object queries. The main focus of the work presented in this paper is the querying and retrieval components of our system. Our query model supports scenario-based, event-based, and object-based queries, where the low-level object features can be used to enrich the queries. Our model also supports inverse querying that can be used as a tool for activity analysis in various domains. A visual query specification interface is also developed as the visual counterpart of our query language.

Performance of our scenario-based querying approach was evaluated through a set of experiments. Since the performance of the overall querying scheme strictly depends on the data extraction and object classification algorithms, we also carried out experiments on these pixel-level and region-level algorithms. It was shown through all these experiments that our system has an effective expressive power and satisfactory retrieval accuracy in video surveillance.

To increase the capabilities and the expressive power of our system, we are planning to implement the negation operator for the scenario-based query results. There might be cases such that the inspectors would like to inspect the data with the help of this negation operator. Moreover, we are planning to include a natural language parser in the query specification user interface, which can learn domain-specific keywords a priori and simplify the query specification process.

References

[1] C. Regazzoni, V. Ramesh, and G. Foresti, "Scanning the issue/technology: Special issue on video communications, processing, and understanding third generation surveillance systems," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1355–1367, 2001.

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

[3] D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons, and A. Jain, "A background model initialization algorithm for video surveillance," in *Int. Conf. on Computer Vision*, Vancouver, Canada, 2001, pp. 733–740.

[4] L. Li, W. Huang, I. Gu, and Q. Tian, "Foreground object detection from videos containing complex background," in *ACM Multimedia 2003*, Berkeley, CA, USA, 2003.

[5] İ. Haritaoğlu, D. Harwood, and L. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, 2000.

[6] A. Bobick and J. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[7] E. Şaykol, U. Güdükbay, and Ö. Ulusoy, "A database model for querying visual surveillance by integrating semantic and low-level features," in *Lecture Notes in Computer Science (LNCS), Proc. of 11th Int. Workshop on Multimedia Information Systems (MIS'05))*, K. Candan and A. Celentano, Eds., Sorrento, Italy, 2005, pp. 163–176.

[8] D. Lyons, T. Brodsky, E. Cohen-Solal, and A. Elgammal, "Video content analysis for surveillance applications," in *Philips Digital Video Technologies Workshop*, 2000.

[9] E. Stringa and C. Regazzoni, "Real-time video-shot detection for scene surveillance applications," *IEEE Trans. on Image Processing*, vol. 9, no. 1, pp. 69–79, 2000.

[10] E. Stringa and C. Regazzoni, "Content-based retrieval and real time detection from video sequences acquired by surveillance systems," in *Int. Conf. on Image Processing*, 1998, pp. 138–142.

[11] C. Regazzoni, C. Sacchi, and E. Stringa, "Remote detection of abandoned objects in unattended railway stations by using a DS/CDMA video surveillance system," in *Advanced Video-Based Surveillance System*, C. Regazzoni, G. Fabri, and G. Vernezza, Eds. Boston, MA: Kluwer, 1998, pp. 165–178.

[12] C. Kim and J. Hwang, "Fast and automatic video object segmentation and tracking for content-based applications," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 2, pp. 122–129, 2002.

[13] C. Kim and J. Hwang, "Object-based video abstraction for video surveillance systems," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1128–1138, 2002.

[14] E. Rivlin, M. Rudzsky, U. B. R. Goldenberg, and S. Lepchev, "A real-time system for classification of moving objects," in *Int. Conf. on Pattern Recognition*, vol. 3, 2002, pp. 688–691.

[15] Y. Bogomolov, G. Dror, S. Lapchev, E. Rivlin, and M. Rudzsky, "Classification of moving targets based on motion and appearance," in *British Machine Vision Conference*, vol. 2, 2003, pp. 429–438.

[16] A. Hampapur, L. Brown, J. Connell, M. Lu, H. Merkl, S. Pankanti, A. Senior, C. Shu, and Y. Tian, "Multi-scale tracking for smart video surveillance," *IEEE Signal Processing Magazine*, vol. 22, no. 2, 2005.

[17] A. Elgammal, D. Harwood, and L. Davis, "Non-parametric model for background subtraction," in *Int. Conf. on Computer Vision and Pattern Recognition, Workshop on Motion*, Ft. Collins, CO, USA, 1999.

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

[19] E. Şaykol, U. Güdükbay, and Ö. Ulusoy, “A histogram-based approach for object-based query-by-shape-and-color in multimedia databases,” *Image and Vision Computing*, vol. 23, no. 13, pp. 1170–1180, 2005.

[20] E. Şaykol, A. Sinop, U. Güdükbay, Ö. Ulusoy, and E. Çetin, “Content-based retrieval of historical Ottoman documents stored as textual images,” *IEEE Trans. on Image Processing*, vol. 13, no. 3, pp. 314–325, 2004.

[21] M. Dönderler, Ö.Ulusoy, and U. Güdükbay, “Rule-based spatio-temporal query processing for video databases,” *The VLDB Journal*, vol. 13, no. 3, pp. 86–103, 2004.

[22] Ninth IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006) Benchmark Data `http://www.cvg.rdg.ac.uk/PETS2006/data.html`, New York, June 2006.

[23] J. Nascimento and J. Marques, “Performance evaluation of object detection algorithms for video surveillance,” *IEEE Trans. on Multimedia*, vol. 8, no. 4, pp. 761–774, 2006.

[24] H. Trees, *Detection, Estimation, and Modulation Theory*. New York: Wiley, 2001.

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 */
            <objectlist>             /* retrieve object(s) */
<objectlist> := [<objectlist> ','] <objlabel>
<range> := all | <videolist>
<videolist> := [<videolist> ','] <vid>
<query-condition> := <object-assignment-list> and <scenario>
<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>

/* 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

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