A Database Model for Querying Visual Surveillance Videos by Integrating Semantic and Low-Level Features[*]

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 visual surveillance has emerged as a trendy application domain in recent years. Many approaches have been developed on video processing and understanding. Content-based access to surveillance video has become a challenging research area. The results of a considerable amount of work dealing with automated access to visual surveillance have appeared in the literature. However, the event models and the content-based querying and retrieval components have significant gaps remaining unfilled. To narrow these gaps, we propose a database model for querying surveillance videos by integrating semantic and low-level features. In this paper, the initial design of the database model, the query types, and the specifications of its query language are presented.

1 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 visual surveillance applications in the recent years. Many approaches related with content-based retrieval and automatic video processing and understanding (e.g., automatic video shot detection, event classification, low-level feature based querying, etc.) have been developed in the mean time with the advances in visual surveillance technology. These advances have led to the integration of automatic video processing and content-based retrieval with

[*] This work is supported in part by Turkish State Planning Organization (DPT) under grant number 2004K120720, and European Commission 6th Framework Program MUSCLE Network of Excellence Project with grant number FP6-507752.

visual surveillance systems. Due to the highly variable nature of visual surveillance videos, a need has arisen for robust scene processing and event recognition.

In our work, the main focus is on *indoor monitoring*, and a framework for querying surveillance videos by integrating semantic and low-level features is developed. Regarding this issue, some powerful systems (e.g., [1–4]) exist in the literature. Besides system-level integration, some methods have been proposed for smaller units (e.g., detecting scene changes [5, 6], moving object detection and tracking [7]). As far as the database indexing and retrieval parts are concerned, the researchers generally designed simple database structures for events. The event descriptors stored in a database generally contain the start and finish times, and the salient object labels. Indexing at the object feature level is not as frequent as the event level. In [8], the authors proposed an approach for traffic surveillance and stored object motion in the database. There are also some approaches that deal with the color information of the video objects (e.g., [2, 4]). However, their use is either to keep track of the video objects or to classify the video objects.

The main contribution of the proposed framework is the querying capability by integrating semantic features (i.e., events, sub-events, and salient objects) and low-level object features (i.e., color, shape, and texture) for surveillance videos. To the best of our knowledge, no systems exist that are embedding object-based low-level features (e.g., color, shape, and texture) to the indexing and retrieval module in visual surveillance domain. Moreover, the framework provides support for effective query specification (e.g., query-by-example, query-by-sketch) and retrieval as opposed to keyword-based database searches. In the following, we describe our motivations and the basic assumptions we made in our work.

- Enriching the querying module with low-level object features (color, shape, texture) is more meaningful for indoor surveillance. We might need to query an event to detect an intruder in a supermarket by specifying the color of his coat, texture on his shirt, etc. This low-level feature enrichment would possibly decrease the rate of false alarms. For the intruder detection example, it is possible that there exist many innocent people making the same type of actions (events) in the supermarket at that time, causing a significant post-processing for the retrieved persons to find the intruder, or generate false alarms.
- Static camera and constant light source are assumed for the sake of simplicity in terms of database modeling. These assumptions fit most to indoor environments than the other categories. Although there are

some recent approaches assuming inputs from two fixed cameras for indoor environments [9], the way of processing object motions is very different and sophisticated (e.g., contour matching and tracking in 3D) for multi-camera surveillance approaches.
- The pre-processing steps for event/object indexing into the database for querying are more straightforward in indoor monitoring.Complex background scenes are rare and the object motions (e.g., motions of humans) are generally expectable. This is basically because of the fact that indoor environments are generally simpler than others.

The rest of this paper is organized as follows: Section 2 summarizes some related studies. The proposed database model is presented in Section 3. Finally, Section 4 concludes the paper.

2 Related Work

Video Surveillance and Monitoring (VSAM) system presented in [3] 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. In that work, a hybrid algorithm was developed, which is based on an adaptive background subtraction by three-frame differencing. The background update 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 object motions. The background maintenance scheme we employ is similar to that of VSAM. However, in our framework, the extracted background is also used for both event annotation and object tracking.

Stringa and Regazzoni [1, 10, 11] proposed a real-time surveillance system employing semantic video-shot detection and indexing. 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. The authors have adapted a change-detection module that processes the background image and the current image to detect the stationary object regions. 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 dangerous object. However, no details are provided for the storage of these low-level object features and their indexing and usage within complex queries. 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.

In [12, 13], an object-based video abstraction model was proposed. The authors employed a moving-edge detection scheme for video frames. The edge map of a frame is extracted first by using Canny edge-detector [14]. The extracted edge map is compared with the background edge map and the moving edges and regions are detected at the end of this process. They employed a semantic shot detection scheme 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.

In [8], Jung et al. proposed a content-based event retrieval framework for traffic surveillance videos using semantic scene interpretation techniques. They employed an adaptive background subtraction and update mechanism, where the background image eventually contains the temporal median values of pixels. One of the important aspects of the work is that they designed the database indexing and retrieval in an object-based manner. However, since their primary concern is traffic surveillance, the object trajectories and motion descriptors are stored in the database. Their query interface supports query-by-example, query-by-sketch, and query-by-weighting on the trajectory descriptors. The database is searched exhaustively to find the best matches for a given query. Our framework also includes object-based querying by providing examples or sketches. However, our querying module also enables specification of more complex queries including low-level and directional descriptors.

Lyons et al. [15] developed video content analyzer (VCA), the main components of which are background subtraction, object tracking, event reasoning, graphical user interface, indexing, and retrieval. They adapted a non-parametric background subtraction approach based on [16]. A finite state model was designed for object tracking. VCA discriminates people from objects and the main events recognized are as follows: *entering scene*, *leaving scene*, *splitting*, *merging*, and *depositing/picking-up*. The retrieval component was designed 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. [4] designed a system for indoor visual surveillance especially in the retail stores and in the houses. They assumed a stationary camera and used background subtraction technique described in [15]. The pixels detected as foreground are grouped into connected components and tracked throughout the scene. Object tracking module handles merging and splitting of moving objects by making use of a color model extracted for each moving object. A list of events that the object is participated are stored for each object, where the events are simply *entering, leaving, merging,* and *splitting.* The system also includes a classification module to distinguish people, pets, and other objects. One of the distinctions of this system and our framework is the use of color feature. In this system, the color feature of the objects is mainly used for reassigning labels for moving connected components, whereas, as an extension, we also provide object-based querying based on color and/or shape features.

Haritaoğlu et al. [17] proposed a 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 the literature, most of the systems have focused on (unattended) object detection and tracking moving object. Extracted event information, based on the object tracking and shot detection modules, is generally stored in a database and exhaustively searched when a query is submitted based on event information (e.g., [15]). From another perspective, retrieving the video sequences related to a generated alarm is the basic way of querying the system (e.g., [1]). In [1], the authors also mentioned more complex query types including color and/or shape properties of the salient objects. The querying module of [8] supports query-by-example, query-by-sketch, and query-by-weighting on the trajectory descriptors of moving objects.

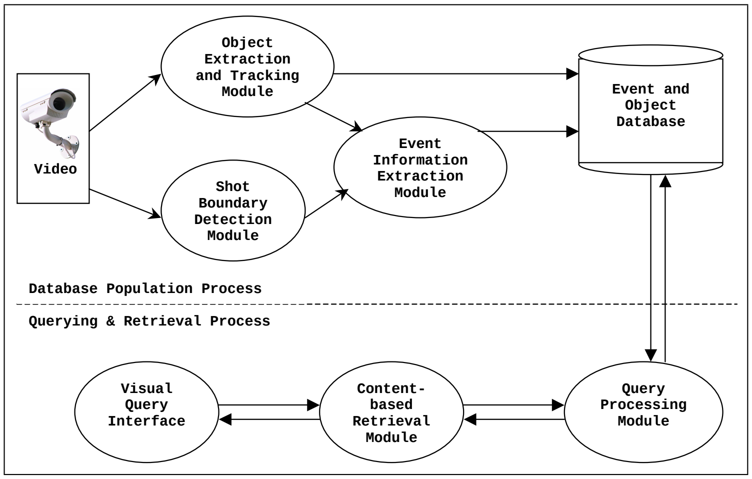

Object Extraction and Tracking Module
Event and Object Database
Video
Event Information Extraction Module
Shot Boundary Detection Module
Database Population Process
Querying & Retrieval Process
Visual Query Interface
Content-based Retrieval Module
Query Processing Module

Fig. 1. *The overall architecture of the framework.*

3 A Database Framework for an Integrated Querying of Visual Surveillance

We propose a framework which provides an integrated environment for querying indoor surveillance videos by semantic (event-based) and low-level (object-based) features. The overall architecture of the framework is shown in Figure 1. The queries are handled by *query processing module*, which communicates with both the *feature database* and the *content-based retrieval module*. The database contains event and object features extracted by automated tools. The *visual query interface* is used to submit queries to the system and to visualize the query results. The users should be able to specify event queries enriched with low-level features for objects and directional descriptors for events.

3.1 Object Extraction and Tracking Module

A fully-automatic object detection and tracking module is designed and implemented, which employs an adaptive background maintenance scheme, similar to the one proposed in [3]. At the pixel-level, the algorithm tries to identify the moving object pixels. The adaptive background subtraction technique is combined with a three-frame differencing technique to determine the moving object pixels as a region and also to detect stop-and-go type of object motions. According to the three-frame differencing algorithm, an object is assumed to be moving if the intensities of the object have changed between current and previous frames, and between current and next-to-previous frames.

Video data can be considered as a sequence of frames. Let $I_f(x, y)$ denote the intensity value of a pixel at (x, y) at frame f. Hence, $M_f(x, y) = 1$ if (x, y) is moving at 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 follows: $M_f(x, y) = 1$ if $(|I_f(x, y) - I_{f-1}(x, y)| > T_f(x, y))$ and $(|I_f(x, y) - I_{f-2}(x, y)| > T_f(x, y))$.

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 to be updated based on the observed moving pixel information at the current frame. A statistical background and threshold maintenance scheme is employed in the module as follows:

$$B_0(x, y) = 0, \tag{1}$$

$$B_f(x, y) = \begin{cases} \alpha B_{f-1}(x, y) + (1 - \alpha) I_{f-1}(x, y), & M_f(x, y) = 0, \\ B_{f-1}(x, y), & M_f(x, y) = 1, \end{cases} \tag{2}$$

$$T_0(x, y) = 1, \tag{3}$$

$$T_f(x, y) = \begin{cases} \alpha T_{f-1}(x, y) + (1 - \alpha)(k \times |I_{f-1}(x, y) - B_{f-1}(x, y)|), & M_f(x, y) = 0, \\ T_{f-1}(x, y), & M_f(x, y) = 1, \end{cases} \tag{4}$$

where k is set to 5 in Eq. 4. As argued in [3], $B_f(x, y)$ 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. A snapshot of object extraction and tracking module is shown in Figure 2.

3.2 Shot Boundary Detection Module

Object tracking and event classification tasks directly benefit from the extracted video *shot*s. Moving objects generally enter or leave the scene at shot boundaries. Hence, the shot boundary detection module is particularly designed for detecting video shot boundaries, hence *cut*s. The shot boundary detection algorithm employed can be summarized as follows: Let f_i and f_{i-1} denote two consecutive frames in a video clip. Let I_f denote the intensity histogram of a frame f. For a consecutive frame pair

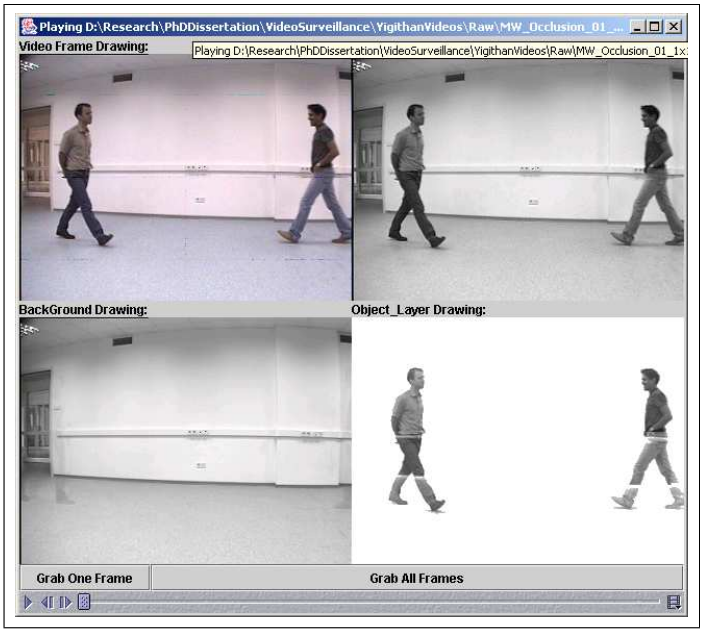

Playing D:\Research\PhDDissertation\VideoSurveillance\YigithanVideos\Raw\MW_Occlusion_01_...
Video Frame Drawing:
Playing D:\Research\PhDDissertation\VideoSurveillance\YigithanVideos\Raw\MW_Occlusion_01_1x
BackGround Drawing:
Object_Layer Drawing:
Grab One Frame
Grab All Frames

Fig. 2. *Object extraction and tracking module user interface. Top-left image is the original image, top-right is the intensity layer image, bottom-left is the background layer image, and bottom-right is the object layer image where moving object pixel region can be seen.*

(f_{i-1}, f_i), let d_i denote the histogram intersection [18] distance between $I_{f_{i-1}}$ and I_{f_i}. By using histogram intersection technique, d_i can be found by

$$d_i = 1 - S_{I_{f_i}, I_{f_{i-1}}} = 1 - \frac{\sum_i^n \min(I_{f_i}[i], I_{f_{i-1}}[i])}{\min(|I_{f_i}|, |I_{f_{i-1}}|)}, \tag{5}$$

where $|I_{f_i}|$ denotes the L_1-norm (i.e., length) of an histogram I_{f_i}. If $d_i \geq t_1$, then i is introduced as a shot boundary, or a cut. The value for t_1 is estimated by trial-error technique and it is found that 0.3763 gives best results for the video clips tested so far. A further improvement is made on the algorithm to increase the success rate. The algorithm starts with a lower t_1 value and extracts shot boundary candidates first. Then, it computes pixel-wise distance d_i^p between f_{i-1} and f_i, where i is a shot boundary candidate. If $d_i^p \geq t_2$, then i is introduced as a shot boundary. The experiments show that $t_1 = 0.3$ and $t_2 = 0.13$ give promising results.

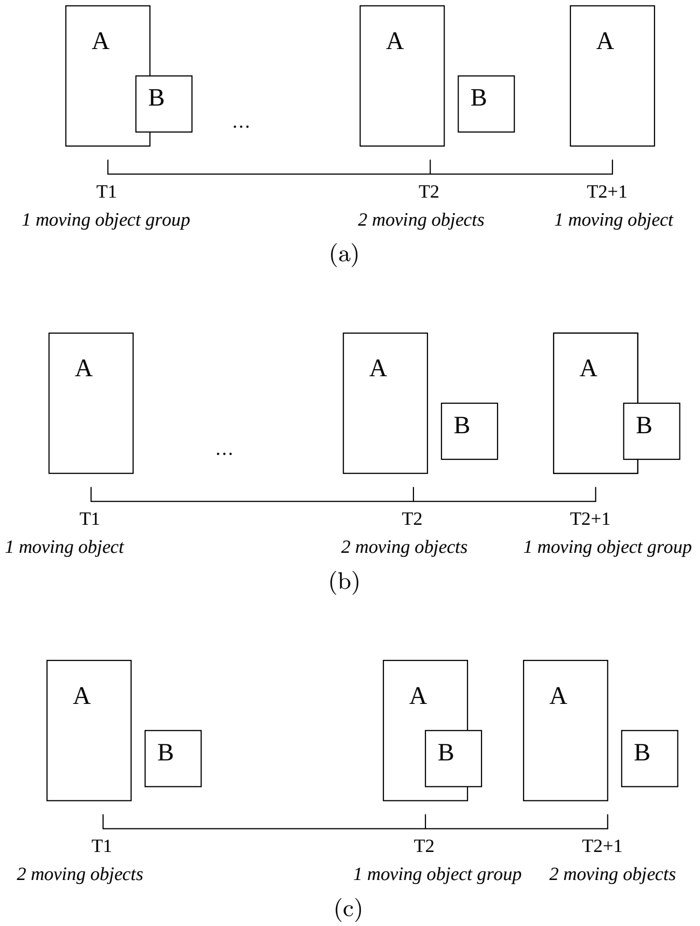
3.3 Event Information Extraction Module

This module is intended to 'annotate' video frame intervals based on event labels, objects appearing within the interval, and low-level feature descriptions for the event objects. There are some specific types of events, which are important for indoor visual surveillance to detect suspicious events. *A person depositing or picking up an object* and *two people crossing over* are examples for two important events. *People/object entering or leaving the scene*, *people/object joining or splitting* can be considered as sub-events to detect suspicious events [15].

The inputs to this module are the moving objects (or moving object groups) at a frame. While detecting moving object groups, our object tracking algorithm benefits from the algorithm discussed in [7]. A moving object group is identified as a moving object until the objects in the group are separated. Counting the number of moving objects also gives an important clue for detecting the events, since the number of moving objects changes at the time of the events. We also keep track of spatial (directional and topological) relations among the moving objects within the event detection algorithm.

Figure 3 illustrates the detecting of events. For example, *deposit* event is detected as follows: The object tracking module detects A and B as a moving object group first at time T1. Then, they are identified as two separate moving objects by the same module through some algorithms used in [7]. Hence, at time T2, the objects A and B are detected as they are moving separately. At time T2+1, the only detected moving object is A. Therefore, a potential *deposit* event is detected at frame T2. The spatial relations among the moving objects are tracked in addition to the outputs of object tracking module, which helped the detection process. The detection process for the other types of events are similar to the deposit event. Refinements on this event detection algorithm have been carried out continuously to end up with a better scheme.

The event information extraction module can be summarized as follows: based on the above depositing event example, the event is detected at time T2. Hence, the event information is stored based on the status at time T2. This type of information storage is reasonable because the status at the time of the suspicious event is of interest while querying. For each frame that an event is identified, the objects appearing on that frame (A and B in this example) are stored in *Event and Object Database* along with their low-level features and the spatial relations among them. The low-level object features stored are *color vector* and *shape vector*, which is a composition of *angular span* and *distance span* [21].



(a)

(b)

(c)

Fig. 3. *Detection of Events. (a) Deposit, (b) Pick Up, and (c) Crossover.*

3.4 Query Processing Module

One of the most important tasks in automated visual surveillance is the query processing module. 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.

Query Types: The main contribution of the querying module in our system is providing support to a wide range of event and object queries. A classification of the types of queries handled can be listed as follows:

- Single Object Queries,
  - Object Entering/Leaving Scene,
- Multi-Object Queries,
  - Object Depositing/Object Picking Up,
  - Objects Crossing Over.

We plan to extend the framework to support queries based on suspicious events in surveillance videos. By adding low-level and/or spatial sub-queries, more complex queries can be submitted to the system. Spatial sub-queries are more meaningful for queries involving multiple objects. However, low-level sub-queries can be supplied for each object. In real-life applications, low-level sub-query specification with high-level descriptors might be sufficient (e.g., a man with a black coat entering scene).

Query Language: An SQL-based querying language, which we call Visual Surveillance Query Language (VSQL), is designed to provide support for integrated querying of indoor visual surveillance videos by spatial, semantic, and low-level features. The query processing module extends the querying capability by allowing low-level object feature specifications as well as spatial relationships among the objects. In short, VSQL provides support for semantic (suspicious event), spatial (relationships among objects), and low-level (object features at a specific suspicious event) sub-queries for visual surveillance domain. Semantic sub-queries can be coupled with either spatial or low-level, or both, to form complex queries. A grammar for VSQL is presented in Appendix A.

VSQL identifies two single-object event conditions (*enter*, *leave*), and three multi-object event conditions (*deposit*, *pickup*, and *crossover*) in a scenario. The moving objects are classified into four categories, namely *person*, *object*, *pet*, *other*, by a classification tool, which is developed outside the scope of the work in this paper [7]. The query processor counts the number of moving objects specified in a query by tracing the *enter* event conditions. A timegap value can be specified between event conditions. The order of the events is considered to be unimportant if a timegap value is not specified for an event condition pair of the same type (i.e., pair of single-object conditions or pair of multi-object event conditions).

Query Processing: The event and object information is stored in the database is based on the frames at which an event is detected. This includes the low-level features of the objects appearing on that frame and the spatial relations among them. By keeping the directional relations among the objects, event querying can be refined by specifying directional predicates in the query. For example, while querying an event of *'a person depositing an object'*, the query can be enriched by directional predicates so that *'a person depositing an object to his west'* can also be queried. Moreover, since the low-level features of moving objects are also stored, more detailed queries can be submitted to the system effectively.

Based on the observation that rule-based model is effective for querying video databases [19, 20], a rule-based model is designed for querying visual surveillance videos. Our framework is to provide support for scenario-based queries which are very difficult to handle in real-time systems. The submitted queries are sent to Prolog, which processes the extracted event and object information is stored in the database (i.e., knowledge-base) based on a set of rules. Prior to the processing by Prolog, the query string is passed through lexical analyzer and a parse tree is created. We are planning to embed a similarity-based metric in this querying process, which will be used as a confidence value for the user.

3.5 Content-Based Retrieval Module

In this section, some real-life querying scenarios are provided. We assumed a lobby of a hotel as our indoor environment for these scenarios. Since our system focuses on database querying and retrieval issues, it is assumed that there is a need to query the suspicious situation given in the scenarios. A real-time visual surveillance system has to detect such cases and inform the human operator for more security. It is also possible that the human operator does not see any danger in the situation at the time of the event. However, he may be informed later on that a dangerous situation has happened in one of the events triggered by the system. Based on this argument, the assumption of the need to query suspicious situations is not superficial.

Scenario 1: *A person with a black coat enters a lobby.*

```
select objectA from 1 where
    objectA = objdata(class = person, color = black) and enter(objectA)
```

The event is a simple object appearance type of query. It is assumed that the dominant color of the person is black, which is the color of his coat.

Scenario 2: *A person enters a lobby with a bag, deposits his bag, and leaves the lobby.*

```
select segment from 1 where
    objectA = objdata(class = person), objectB = objdata(class = object)
    and enter(objectA) enter(objectB) deposit(objectA,objectB) leave(objectA)
```

The event described in this scenario is very crucial because unattended bags are one of the primary sources of suspicious situations in indoor environments. Additional descriptors for both person and bag improve the quality of the retrieval. Directional descriptors can be added as well since the directional relations are stored. Hence, the query phrase 'deposits his bag' can be refined by 'deposits his bag to his west', which might give better results. Then, the deposit condition will be as follows:

```
    deposit(objectA,objectB,west)
```

Scenario 3: *A person enters a lobby with a bag, after 3 seconds another person enters the lobby, two persons meet and exchange the bag, then they leave the lobby.*

```
select segment from 1 where
    objectA = objdata(class = person), objectB = objdata(class =
    object),
    objectC = objdata(class = person) and
    enter(objectA) enter(objectB) 3 enter(objectC)
    crossover(objectA,objectC) deposit(objectA,objectB)
    pickup(objectC,objectB) leave(objectA) leave(objectC)
```

This scenario is an example of a 'cross-over', and generally real-time systems detect such events and inform human operators. If we consider a quite crowded lobby, the number of crossovers is relatively large. Hence, providing additional descriptors for persons and/or bag (both low-level and directional) decreases the number of possible results of querying. This directly shortens the time and helps the security persons to catch the intruders.

Scenario 4: *A person with a black coat enters a lobby with a yellow bag, deposits the bag, another person with a white coat enters the lobby, picks up the bag.*

```
select segment from 1 where
    objectA = objdata(class = person, color = black),
    objectB = objdata(class = object, color = yellow),
    objectC = objdata(class = person, color = white) and
    enter(objectA) enter(objectB) deposit(objectA,objectB)
    enter(objectC) pickup(objectC,objectB)
```

This scenario describes a sequence of deposit and pickup events without crossover. Color descriptors are added for all of the objects acting in the scenario. The leaving times of the persons are not of interest for the querying of the event.

As mentioned in the scenarios, the suspicious events can be queried by adding low-level features to (moving) objects and/or directional relations for the event. This type of querying improves the retrieval quality and decreases the search space. These gains are more meaningful when the number of events to be searched in the database is relatively large.

4 Conclusion

In this paper, we propose a database model for an integrated querying of visual surveillance videos by semantic and low-level features. The application domain chosen is indoor monitoring environment video, which fits most into our assumptions and which is simpler than other complex environments. The system has a database population process that extracts necessary event and object information for effective querying. An SQL-based querying language is designed to express the query types. A wide range of event and object queries including semantic, spatial, and low-level is supported.

As the database modeling process is evolving, improvements will be made both in terms of database population and querying-and-retrieval processes. The design and implementation of the query processing module is an ongoing project, hence the above improvements will be employed accordingly to handle a wide range of query set effectively.

References

1. Stringa, E., Regazzoni, C.: Real-time video-shot detection for scene surveillance applications. IEEE Trans. on Image Processing **9** (2000) 69–79
2. Foresti, G., Marcenaro, L., Regazzoni, C.: Automatic detection and indexing of video-event shots for surveillance applications. IEEE Trans. on Multimedia **4** (2002) 459–471
3. Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O., Burt, P., Wixson, L.: A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12, Carnegie Mellon University, The Robotics Institute (2000)
4. Brodsky, T., Cohen, R., Cohen-Solal, E., Gutta, S., Lyons, D., Philomin, V., Trajkovic, M.: Visual surveillance in retail stores and in the home. In: Video-Based Surveillance Systems: Computer Vision and Distributed Processing, Kluwer Academic Pub. (2001) 51–65

5. Latecki, L., Wen, X., Ghubade, N.: Detection of changes in surveillance videos. In: IEEE Conf. on Adv. Video and Signal Based Surv. (AVSS'03). (2003) 237–242
6. Stefano, L.D., Mattoccia, S., Mola, M.: A change-detection algorithm based on structure and colour. In: IEEE Conf. on Adv. Video and Signal Based Surv. (AVSS'03). (2003) 252–259
7. Töreyin, B., Çetin, A., Aksay, A., Akhan, M.: Moving object detection in wavelet compressed video. Signal Processing: Image Communication **20** (2005) 255–264
8. Jung, Y., Lee, K., Ho, Y.: Content-based event retrieval using semantic scene interpretation for automated traffic surveillance. IEEE Trans. on Intelligent Transportation Systems **2** (2001) 151–163
9. Eaton, R., Scassellati, B.: ViSIT: Visual surveillance and interaction tracking. In: `http://zoo.cs.yale.edu/classes/cs490/02-03a/ross.eaton/`. (Social Robotics Laboratory, Yale University, accessed at February 27, 2005)
10. Stringa, E., Regazzoni, C.: Content-based retrieval and real time detection from video sequences acquired by surveillance systems. In: Int. Conf. on Image Processing. (1998) 138–142
11. Regazzoni, C., Sacchi, C., Stringa, E.: 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, Boston, MA: Kluwer (1998) 165–178
12. Kim, C., Hwang, J.: Fast and automatic video object segmentation and tracking for content-based applications. IEEE Trans. on Circuits and Systems for Video Technology **12** (2002) 122–129
13. Kim, C., Hwang, J.: Object-based video abstraction for video surveillance systems. IEEE Trans. on Circuits and Systems for Video Technology **12** (2002) 1128–1138
14. Canny, J.: A computational approach to edge detection. IEEE Trans. on Pattern Analysis and Machine Intelligence **8** (1986) 679–698
15. Lyons, D., Brodsky, T., Cohen-Solal, E., Elgammal, A.: Video content analysis for surveillance applications. In: Philips Digital Video Technologies Workshop. (2000)
16. Elgammal, A., Harwood, D., Davis, L.: Non-parametric model for background subtraction. In: Int. Conf. on Computer Vision and Pattern Recognition, Workshop on Motion. (1999)
17. Haritaoğlu, İ., Harwood, D., Davis, L.: W4: Real-time surveillance of people and their activities. IEEE Trans. on Pattern Analysis and Machine Intelligence **22** (2000) 809–830
18. Swain, M., Ballard, D.: Color indexing. Int. J. of Comp. Vis. **7** (1991) 11–32
19. Dönderler, M., Şaykol, E., Arslan, U., Ulusoy, Ö., Güdükbay, U.: BilVideo: Design and implementation of a video database management system. Multimedia Tools and Applications (accepted for publication) (2005)
20. Dönderler, M., Ö.Ulusoy, Güdükbay, U.: Rule-based spatio-temporal query processing for video databases. The VLDB Journal **13** (2004) 86–103
21. Şaykol, E., Sinop, A., Güdükbay, U., Ulusoy, Ö., Çetin, E.: Content-based retrieval of historical Ottoman documents stored as textual images. IEEE Trans. on Image Processing **13** (2004) 314–325

A Grammar for Visual Surveillance Querying Language(VSQL)

Visual Surveillance Query Language (VSQL) is designed to provide support for integrated querying of indoor visual surveillance videos by spatial, semantic, and low-level features.

```
/* main query string */
<query> := select <target> from <range> [where <querycondition>] ‘;’

/* main query string components */
<target> := event | <objectlist> | segment
<objlist> := [<objlist> ‘,’] <objlabel>
<range> := all | <videolist>
<videolist> := [<videolist> ‘,’] <vid>
<querycondition> := <objectassignmentlist> and <scenario>
<objectassignmentlist> := [<objectassignmentlist> ‘,’] <objectassignment>
<scenario> := [<scenario> <timegap>] <eventcondition>
<eventcondition> := <entercondition> | <leavecondition>
   | <depositcondition> | <pickupcondition> | <crossovercondition>
<objectassignment> := <objlabel> <objoperator> <objcondition>

/* single object query conditions */
<entercondition> := enter ‘(’ <objlabel> ‘)’
<leavecondition> := leave ‘(’ <objlabel> ‘)’

/* multi object query conditions */
<depositcondition> := deposit ‘(’ <multiobjcondition> ‘)’
<pickupcondition> := deposit ‘(’ <multiobjcondition> ‘)’
<crossovercondition> := crossover ‘(’ <multiobjcondition> ‘)’
<multiobjcondition> := <objlabel> ‘,’ <objlabel> [‘,’ <directional>]
<directional> := west | east | north | south | northeast | southeast
   | northwest | southwest

/* object condition */
<objcondition> := objdata ‘(’ <objdesclist> ‘)’
<objdesclist> := [<objdesclist> ‘,’] <objdesc>
<objdesc> := class ‘=’ <classvalue> | <colordesc> | <shapedesc> | <texturedesc>
<colordesc> := color ‘=’ <colorlabel> | <colorvector>
<shapedesc> := [<shapedesc> ‘,’] <shapepair>
<texturedesc> := texture ‘=’ <textureid>
<colorvector> := [<colorvector> ‘,’] <colorpair>
<colorpair> := ‘(’ <intvalue> ‘,’ <doublevalue> ‘)’
<shapepair> := ‘(’ <intvalue> ‘,’ <doublevalue> ‘)’

/* primitive types */
<vid> := (1-9)(0-9)*
<timegap> := null | <intvalue>
<objlabel> := (a-z)(A-Za-z0-9)*
<objoperator> := ‘=’ | ‘‘!=’’
<classvalue> := person | object | pet | other
<intvalue> := (1-9)(0-9)*
<doublevalue> := <intvalue> ‘.’ <intvalue>
<colorlabel> := red | green | blue | yellow | white | black | orange | violet
<textureid> := (1-9)(0-9)*  /* enumerated set of texture patterns */
```