

# A SCENARIO-BASED QUERY PROCESSING FRAMEWORK FOR VIDEO SURVEILLANCE

A DISSERTATION SUBMITTED TO  
THE DEPARTMENT OF COMPUTER ENGINEERING  
AND THE INSTITUTE OF ENGINEERING AND SCIENCE  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY

By  
Ediz Şaykol  
September, 2009

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

---

Assoc. Prof. Dr. Uğur Güdükbay (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

---

Prof. Dr. Özgür Ulusoy (Co-Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

---

Prof. Dr. M. Volkan Atalay

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

---

Assoc. Prof. Dr. A. Aydın Alatan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

---

Asst. Prof. Dr. Selim Aksoy

Approved for the Institute of Engineering and Science:

---

Prof. Dr. Mehmet B. Baray  
Director of the Institute

## ABSTRACT

# A SCENARIO-BASED QUERY PROCESSING FRAMEWORK FOR VIDEO SURVEILLANCE

Ediz Şaykol

Ph.D. in Computer Engineering

Supervisors: Assoc. Prof. Dr. Uğur Güdükbay and

Prof. Dr. Özgür Ulusoy

September, 2009

Video surveillance has become one of the most interesting and challenging application areas in video processing domain. Automated access to the semantic content of surveillance videos to detect anomalies is among the basic tasks; however due to the high variability of the visual features and large size of the video input, it still remains a challenging issue. A considerable amount of research dealing with automated access to video surveillance has appeared in the literature; however, significant semantic gaps in event models and content-based access still remain. In this thesis, we propose a scenario-based query processing framework for video surveillance archives, especially for indoor environments. A scenario is specified as a sequence of event predicates that can be enriched with object-based low-level features and directional predicates. We also propose a keyframe labeling technique, which assigns labels to keyframes extracted based on keyframe detection algorithm, hence transforms the input video to an event sequence based representation. The keyframe detection scheme relies on an inverted tracking scheme, which is a view-based representation of the actual content by an inverted index. We also devise mechanisms based on finite state automata using this event sequence representation to detect a typical set of anomalous events in the scene, which are also used for meta-data extraction. Our query processing framework also supports inverse querying and view-based querying, for after-the-fact activity analysis, since the inverted tracking scheme effectively tracks the moving objects and enables view-based addressing of the scene. We propose a specific surveillance query language to express the supported query types in a scenario-based manner. We also present a visual query specification interface devised to enhance the query-specification process. It has been shown through performance experiments that the keyframe labeling algorithm significantly reduces the storage requirements and yields a reasonable anomaly detection performance. We

have also conducted performance experiments to show that our query processing technique has a high expressive power and satisfactory retrieval accuracy in video surveillance.

*Keywords:* video surveillance, scenario-based querying, keyframe labeling, inverse querying, view-based querying, anomaly detection, after-the-fact analysis.

## ÖZET

# GÖZETİM VİDEOLARI İÇİN SENARYO TABANLI SORGULAMA ÇATISI

Ediz Şaykol

Bilgisayar Mühendisliği, Doktora

Tez Yöneticileri: Doç. Dr. Uğur Güdükbay ve

Prof. Dr. Özgür Ulusoy

Eylül, 2009

Video gözetim son yıllarda en çok ilgilenilen ve üzerinde çalışılan video işleme uygulama alanlarından biridir. Olağandışı durum yakalamak için anlamsal içeriğe otomatik erişim en temel görevlerdendir; ancak büyük girdi boyutu ve görsel özelliklerdeki değişkenlik nedeniyle problem zorluğunu korumaktadır. Literatürde gözetim videolarına otomatik erişim alanında yeterli çalışma yapılmış olmasına rağmen olay modelleri ve içerik erişimi alanlarında anlambilimsel eksiklikler bulunmaktadır. Bu tez kapsamında özellikle iç mekan videoları için senaryo tabanlı sorgulama çatısı önerilmektedir. Senaryo olay yüklemelerinden oluşan bir dizi olarak belirlenmekte ve nesne tabanlı düşük-düzey özellikleri ve yönsel yüklemeler ile zenginleştirilebilmektedir. Ayrıca, anahtar çerçeve etiketleme tekniği önerilmektedir. Bu teknik anahtar çerçeve yakalama algoritmasından gelen anahtar çerçevelere etiket atayarak videonun olay dizisi olarak ifade edilmesini sağlamaktadır. Anahtar çerçeve yakalama, içeriğin bakış tabanlı gösterilmesini sağlayan ters izleme yöntemine dayanmaktadır. Bu olay dizisi gösterimi kullanılarak olağandışı durum yakalamak amacıyla sonlu durum makinesi tabanlı mekanizmalar geliştirilmiştir. Bu mekanizmalar ayrıca yardımcı veri çıkarma işleminde de kullanılmaktadır. Senaryo tabanlı sorgulama çatısı ters izleme yöntemini sayesinde olay sonu analizi için ters sorgulama ve bakış tabanlı sorgulamayı desteklemektedir. Desteklenen sorgu tiplerini ifade etmek amacıyla özel olarak tasarlanmış bir gözetim sorgu dili önerilmektedir. Sorgu dilinde ifadeyi kolaylaştırmak için ayrıca görsel sorgu belirleme arayüzü geliştirilmiştir. Başarım deneylerinde gösterildiği gibi anahtar çerçeve etiketleme algoritması bellek gereksinimini önemli ölçüde düşürmekte ve yeterli düzeyde olağandışı durum yakalama başarımı göstermektedir. Ayrıca sorgulama işlemi başarımının video gözetim alanında etkili ifade gücü ve yeterli erişim doğruluğuna sahip olduğunu göstermek için deneyler yapılmıştır.

*Anahtar sözcükler:* video gözetim, senaryo tabanlı sorgulama, anahtar çerçeve etiketleme, ters sorgulama, bakış tabanlı sorgulama, olağandışı durum yakalama, olay sonu analizi.

## Acknowledgement

I would like to express my sincere gratitude to my supervisors Assoc. Prof. Dr. Uğur Gdkbay and Prof. Dr. zgr Ulusoy for their instructive comments, suggestions, support, encouragement, and patience throughout the supervision of the thesis.

I am grateful to Prof. Dr. M. Volkan Atalay, Assoc. Prof. Dr. A. Aydın Alatan and Asst. Prof. Dr. Selim Aksoy for reading and reviewing this thesis.

Finally, I would like to express my deepest thanks to my family and to my wife for their morale support.



To my endless love,

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Contributions of the Thesis . . . . .	4
1.2	Organization of the Thesis . . . . .	6
<b>2</b>	<b>Keyframe-Based Data Modeling</b>	<b>8</b>
2.1	Keyframe Labeling . . . . .	9
2.2	Moving Region Extraction . . . . .	11
2.3	Inverted Tracking . . . . .	13
2.4	Keyframe Detection . . . . .	17
<b>3</b>	<b>Histogram-Based Shape and Color</b>	<b>20</b>
3.1	Histogram Construction . . . . .	21
3.2	Similarity Calculation . . . . .	24
<b>4</b>	<b>Meta-Data Extraction</b>	<b>26</b>
4.1	Object-based Meta-Data Extraction . . . . .	28

4.1.1	Color Descriptor . . . . .	28
4.1.2	Shape Descriptor . . . . .	28
4.1.3	Class Descriptor . . . . .	29
4.2	Event-based Meta-Data Extraction . . . . .	30
4.2.1	Single Object Annotation . . . . .	30
4.2.2	Multi-Object Annotation . . . . .	31
4.2.3	Post-Detection Process . . . . .	39
<b>5</b>	<b>Scenario-Based Querying</b>	<b>41</b>
5.1	VSQ: Video Surveillance Query Language . . . . .	42
5.1.1	Scenario-based Query . . . . .	43
5.1.2	Event-based Query . . . . .	44
5.1.3	Object-based Query . . . . .	44
5.1.4	Inverse Query . . . . .	45
5.1.5	View-based Query . . . . .	46
5.1.6	External Predicate Definition . . . . .	46
5.2	Query Processing . . . . .	47
5.2.1	Logical Operators on Query Results . . . . .	50
5.2.2	View-based Query Processing . . . . .	53
5.3	Visual Query Specification . . . . .	53
<b>6</b>	<b>Performance Experiments</b>	<b>57</b>

6.1	Performance of Keyframe Labeling . . . . .	58
6.1.1	Pixel-Level Analysis . . . . .	58
6.1.2	Grid Size Analysis . . . . .	59
6.1.3	Temporal Filtering Analysis . . . . .	60
6.1.4	Storage Gain Analysis . . . . .	63
6.2	Performance of Histogram-Based Approach . . . . .	63
6.3	Performance of Object Classification . . . . .	65
6.4	Performance of Semantic Annotation . . . . .	68
<b>7</b>	<b>Related Studies</b>	<b>70</b>
7.1	Video Surveillance Systems . . . . .	71
7.2	Comparison of Our System to Related Work . . . . .	74
<b>8</b>	<b>Conclusion</b>	<b>76</b>
	<b>Bibliography</b>	<b>78</b>
	<b>Appendix</b>	<b>85</b>
<b>A</b>	<b>Grammar for Scenario-Based Querying</b>	<b>85</b>
A.1	VSQL Grammar Specification . . . . .	85
A.2	Rules for Query Processing . . . . .	88
A.3	Sample Facts-base Snapshot . . . . .	93

# List of Figures

1.1	The architecture of our framework. The queries are handled by the scenario-based query processing module, which communicates with both the meta-data store and the querying interface. The meta-data contains event and object features extracted by automated tools based on keyframe labeling. These automated tools also trigger the real-time alerting module. . . . .	3
2.1	The semantic flow in information extraction. . . . .	9
2.2	The pseudo-code of the keyframe labeling algorithm. . . . .	10
2.3	The noise reduction filters applied on a sample frame. (a) Original frame from PETS 2006 S1-T1-C dataset [2] at $t_1$ , (b) Processed frame at $t_1$ , and (c) Processed frame at $t_2 = t_1 + 0.25$ seconds. . .	14
2.4	Inverted tracking on a sample frame. (a) Original frame from PETS 2006 S1-T1-C dataset [2], (b) Processed frame for $8 \times 8$ grid, and (c) The Motion Appearance Mask (MAM) of the frame. . . . .	15
2.5	The pseudo-code of the keyframe detection algorithm. . . . .	17
2.6	Keyframe filtering applied on PETS 2006 [2] datasets (a) S1-T1-C, originally has 3012 frames, 154 keyframes are extracted on the average; and (b) S2-T3-C, originally has 2551 frames, 138 keyframes are extracted on the average. . . . .	19

3.1	Angle histogram calculations. $\alpha$ is the polar angle for object pixel $p_i$ . $R_1$ region is an equivalence pixel class for angle histogram, and $R_2$ region is an equivalence pixel class for distance histogram. . .	22
4.1	The FSA for recognizing event sequence for crossover and move-together. . . . .	32
4.2	The FSA for recognizing event sequence for deposit. . . . .	34
4.3	The FSA for recognizing event sequence for pickup. . . . .	35
4.4	FSA state transitions for crossover detection for PETS 2006 S1-T1-C dataset [2]. (a) Transition from $S_0$ to $S_1$ , (b) Transition from $S_1$ to $S_2$ , and (c) Transition from $S_2$ to $S_3$ and crossover detection.	36
4.5	FSA state transitions for deposit detection for PETS 2004 leftbag dataset [1]. (a) Transition from $S_0$ to $S_1$ , (b) Transition from $S_1$ to $S_2$ , and (c) Transition from $S_2$ to $S_3$ and deposit detection. . .	37
4.6	FSA state transitions for pickup detection for PETS 2004 leftbag dataset [1]. (a) Transition from $S_0$ to $S_1$ , (b) Transition from $S_1$ to $S_2$ , and (c) Transition from $S_2$ to $S_3$ and pickup detection. . . .	38
4.7	Annotation of events. (a) 2 objects are detected at $t_1$ , and they are classified as one human (H), and one non-human (NH). Then, they are detected as two single objects at $t_2$ , and when NH stops at $t_3$ , <b>deposit</b> is identified. (b) Similar to (a), but <b>pick up</b> is identified. (c) H1 and H2 are classified as human at $t_1$ , and an object group is detected at $t_2$ . By tracking the orientations of the objects within $t_1$ - $t_3$ time interval, <b>crossover</b> is identified. (d) Similar to (c), except that <b>move together</b> is identified because H1 and H2 move in the same direction. . . . .	40

5.1	The query-processing flowchart. A VSQL query submitted to the GUI passes through the parsing, binding, and processing steps; the results are then presented to the user. . . . .	48
5.2	The conjunction operation applied on query results. . . . .	51
5.3	The disjunction operation applied on query results. . . . .	52
5.4	The specification of Query 5.11. The event sequence corresponding to the scenario is shown in the first row of the scenario drawing panel. The letters in the boxes on the drawing panel ('E', 'CC', 'DD', 'PP' and 'L') denote enter, crossover, deposit, pickup and leave event predicates, respectively. . . . .	54
5.5	Object specification: (a) name; (b) name of the owner/creator; (c) type; (d) color; (e) shape of the object; (f), (g) and (h) are the auxiliary interface elements. . . . .	56
6.1	ROC curve/Analysis of pixel-level algorithms. The learning constant $\alpha$ varies from 0.6 to 0.8 in increments of 0.05. The temporal duration constant $\tau$ is set to 2 and 3. . . . .	59
6.2	ROC curve analysis for the grid size parameter. (a) PETS 2006 dataset, and (b) PETS 2004 dataset. . . . .	61
6.3	ROC curve analysis for the temporal detection threshold parameter $t_d$ . (a) PETS 2006 dataset, and (b) PETS 2004 dataset. . . . .	62
6.4	The storage gain and the reduce in the input size for detection for PETS 2004 and PETS 2006 datasets. The keyframe count that is extracted by our technique significantly reduces not only the input size for anomaly detection but also the storage space for after-the-fact analysis. . . . .	64

6.5	Retrieval effectiveness results, (a) with the dataset used in [52, 51], (b) with the dataset used in [40], and (c) with the two datasets mixed. . . . .	66
6.6	Object classification accuracy analysis. . . . .	67
6.7	Semantic annotation accuracy analysis. . . . .	68



# List of Tables

4.1	The state transition function $\delta_C$ for the automaton detecting crossover and move-together. $S_0$ is the initial state, and $S_3$ is the final state accepting event sequence for crossover and move-together. . . . .	33
4.2	The state transition function $\delta_D$ for the automaton detecting deposit. $S_0$ is the initial state, and $S_3$ is the final state accepting event sequence for deposit. . . . .	33
4.3	The state transition function $\delta_P$ for the automaton detecting pickup. $S_0$ is the initial state, and $S_3$ is the final state accepting event sequence for pickup. . . . .	35

# Chapter 1

## Introduction

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

Video surveillance has become an interesting and challenging application domain in video processing. Automated access to the semantic content of surveillance videos is among the interesting areas, basically to detect anomalous situations in the scene. An automated video surveillance system should support both real-time alarm generation and offline inspection components to satisfy the requirements of the operators [32]. In either side, the input video stream should be processed appropriately so that the actions are correctly analyzed. The primary challenges are the large input size and the high variability of the audio-visual features, hence it still remains a challenging issue to access the semantic content of the videos automatically.

The automated video surveillance processing generally starts with the detection of moving regions/objects. Background/foreground subtraction [11, 27, 19, 28, 31, 16] or temporal template-based methods [22, 4] are widely used to detect moving objects. This first step is followed by tracking, classification, and possibly understanding the object behavior [22, 23]. One of the basic aims in understanding the object behavior is detecting the anomalies in the actions of the objects [16, 22, 49, 50, 53, 20, 15]. Abnormal situations and anomalies are reported to the operator and/or stored in a database for later inspection.

Providing a general solution to detect anomalous situations in video surveillance is still an open research area. Reasonable accuracies can be achieved for specific video surveillance applications by restricting the variable nature of the video data. Detecting anomalies requires tracking the actions of moving regions. A typical video stream has too many frames, hence too many moving regions to deal with. Keyframe-based techniques reduce the number of regions to be processed, but effective data models are required for anomaly detection. Pixel-level or region-level detection techniques might have high processing costs or performance limitations for on-the-fly detection due to the large input size.

One of the basic tasks in offline inspection is the content-based retrieval of surveillance videos from the database. In the literature, the researchers generally assume simple data structures for the semantic content: events and objects. The event descriptors contain time information and the objects acting in the event. Object indexing is not as frequent as event indexing and lacks low-level (or object-based) features.

There are specific situations that can be regarded as a sequence of events, and the existence of the whole sequence is of interest. We propose a *scenario-based query processing framework* to detect such sequences in video-surveillance archives, and hence reduce the gap between low-level features and high-level semantic content. Our framework provides support for querying by event-based and object-based features. The supported query types can be ordered to form a scenario-based query where the temporal information among the (sub)queries is also included in the scenario-based query expression.

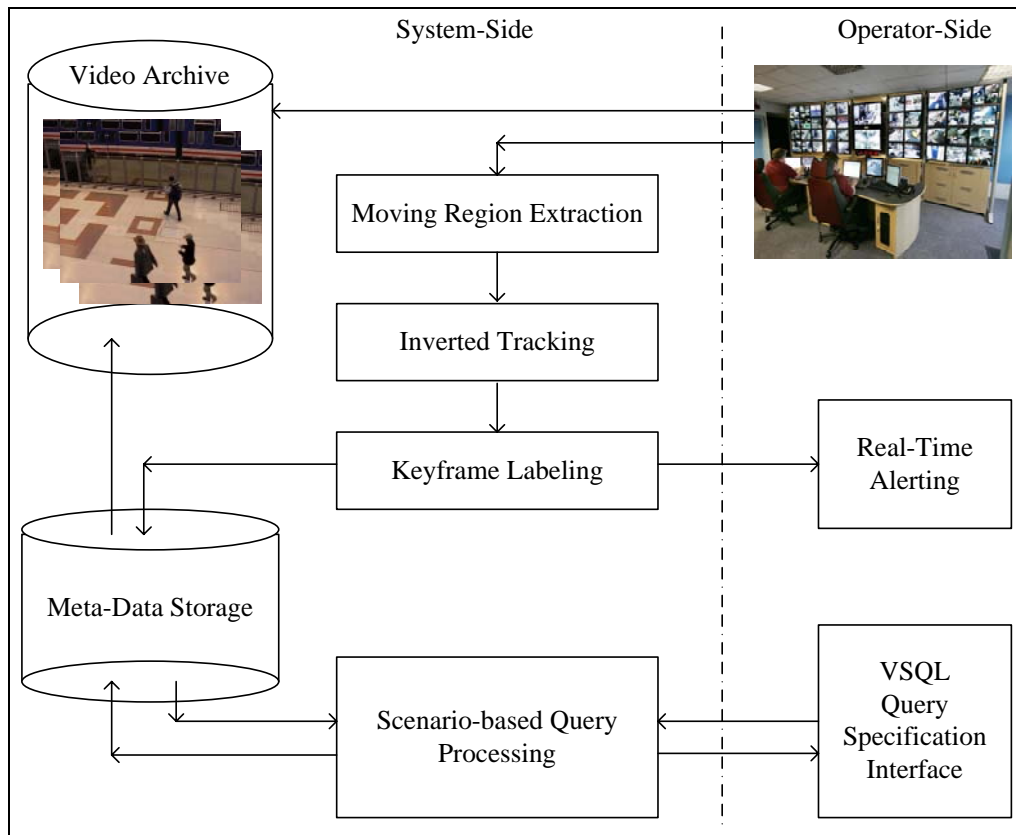


Figure 1.1: The architecture of our framework. The queries are handled by the scenario-based query processing module, which communicates with both the meta-data store and the querying interface. The meta-data contains event and object features extracted by automated tools based on keyframe labeling. These automated tools also trigger the real-time alerting module.

The architecture of our framework [36, 37] for indoor scenes (e.g., lobbies, retail stores, etc.) with a fixed camera is shown in Figure 1.1. The framework presents a keyframe-based data modeling scheme, which utilizes a new data representation for object tracking. A scenario-based query processing component is also presented, which provides support for scenario-based, event-based, and object-based queries, where the low-level object features and directional predicates can be used to improve expressiveness and effectiveness. Our query model also supports *inverse querying*, as well as some statistical view-based query types that can be used as tools for activity analysis in various domains, e.g., video forensics. Our Video Surveillance Query Language (VSQI) has been designed

specifically for scenario-based querying purposes. A query-specification interface has also been developed, which can be considered as the visual counterpart of VSQL. The query-specification interface is as generic and flexible as our query model.

## 1.1 Contributions of the Thesis

The main contributions of the thesis can be listed in two main areas:

### *Scenario-Based Query Processing:*

- The major contribution of the thesis is the supported querying capabilities. To the best of our knowledge, no video surveillance system has been introduced in the literature that supports scenario-based querying enhanced with object-based low-level features and directional predicates, as well as inverse querying and view-based querying. Our observations have shown that scenario-based querying provides an effective medium for after-the-fact activity analysis, since the abnormalities can be expressed in an effective form while preserving the temporal relations among events with a wide set of low-level subqueries. The proposed query-processing framework provides support for a wide range of query types valuable for video surveillance. The supported query types include event and object queries enriched with the low-level feature descriptors (e.g., color, shape) and directional predicates. The query model of the framework allows *scenario-based querying*, where a scenario is a sequence of events ordered temporally. This type of querying increases the retrieval quality of offline inspection.
- We devised a rule-based query-processing module for scenario-based querying. We use Prolog as our inference engine. The extracted meta-data is generic in the sense that the predicates are valid and valuable in almost every type of video-surveillance application. Our rule-based query-processing module provides a flexible mechanism for external predicate definition (i.e., simply by rule injection) so that our framework can be tailored to various

domains.

- We provide a textual query language, which we call Video Surveillance Query Language (VSQL), and its visual counterpart to provide complete querying and retrieval capability. These components are designed in a generic and flexible manner so that they can be used in a variety of video-surveillance domains.

*Keyframe-Based Meta-Data Extraction:*

- In our data model, we introduce a new data representation scheme for region tracking, which we call *Inverted Tracking*. The main benefit of this scheme is that it addresses the scene in a view-based manner with respect to the human operators' points of view. Since the operators inspect the scene to determine abnormalities performed by moving objects, fixed view-based addressing is employed as a part of our data model to enhance its expressive power. For example, the most popular paths of moving objects and the number of objects entering the scene from the left/right side can be queried.
- We propose a keyframe detection and labeling scheme based on inverted tracking. A keyframe is detected if a change occurs in the *Motion Appearance Mask (MAM)* of the frame when compared to that of the previous frame. MAM is a binary representation of the motion, where a 1 in this mask denotes the presence of a motion in that cell. The keyframes are categorized into 4 simple types, namely *join*, *split*, *move*, and *stop* based on the appearances of the identified moving regions. The input stream is represented as a temporally-ordered sequence of keyframe labels, and the anomaly detection is carried out on this sequence. Since an anomaly can be represented by a sub-sequence of keyframe labels, the complexity of the anomaly detection tasks is reduced. Since the anomalies have to be stored in a database for after-the-fact analysis, the required storage space is significantly reduced.
- We also provide mechanisms to detect a typical set of anomalous situations including *crossover*, *deposit*, and *pickup*. To this end, we devise three

separate Finite State Automata to recognize the sequences corresponding to these behaviors. The inputs of these Finite State Automata are the keyframe labels that we assign to the extracted keyframes. Experimental results show that the use of our keyframe labeling technique with these mechanisms yields a reasonable anomaly detection performance. Hence, the mechanisms based on Finite State Automata can be utilized for an on-the-fly anomaly detection scheme together with the keyframe labeling technique.

- A histogram-based approach, originally proposed for content-based retrieval for query-by-color-and-shape [38], is utilized for the object-level meta-data extraction process. This approach is used to obtain high-level descriptors for color and shape information. The shape histograms are also used for obtaining class types (e.g., human, non-human, object group) for moving objects.

## 1.2 Organization of the Thesis

The thesis is organized as follows: Chapter 2 presents the keyframe-based data modeling techniques that form a basis for the meta-data extraction process. Chapter 3 describes the histogram-based approach to encode shape and color features of the objects. In addition to the usage of these features in content-based retrieval, the histogram-based approach is used in several aspects of meta-data extraction process. Chapter 4 provides description of the meta-data extraction process employed in the scenario-based querying framework. Chapter 5 presents the scenario-based query model that we employ. The supported query types, query expression techniques, and the query processing strategies are described based on the extracted meta-data. Chapter 6 presents the experiments evaluating the performance of the data modeling, meta-data extraction, and query processing techniques. Chapter 7 describes some of the existing studies that are related with our work. A comparison of our work to these systems is also provided in this chapter. Chapter 8 concludes the thesis. Appendix A provides the

grammar for video surveillance query language (VSQL) and the rules used by the query processor for scenario-based querying. It also gives a snapshot of a sample fact-base to illustrate how the meta-data is obtained and used to process queries.



## Chapter 2

# Keyframe-Based Data Modeling

In video processing, storage requirement is a very crucial issue due to huge size of the video data. Keyframe-based video processing techniques are popular since they reduce the storage requirements significantly by storing only the data at the keyframes. A keyframe is generally identified when there is a change in the spatio-temporal relations among the salient objects in the scene [13]. In video surveillance, there are abnormal situations to be detected, and hence, there could be other conditions based on the change in the global motion of the scene to detect keyframes. To cover most of the abnormal situations, our keyframe detection algorithm categorizes the keyframes into four primitive types, namely *join*, *split*, *move*, and *stop*, based on the appearances of the extracted moving regions. These four labels are among the primitive event types and it is observed that they can be used to detect typical abnormal situations such as crossover, deposit, and pickup. As a result of this step, a label is assigned for each keyframe and the input video stream is represented as a sequence of events [35].

Our multi-layered data model contains region-level information at the lowest level; e.g., pixel-data for moving regions and motion orientation. At the next level, object-level information is extracted; e.g., class type (e.g., human, non-human, object group) and color and shape feature vectors. At the third level, primitive object actions are annotated; e.g., stop, move, and enter. At the highest level, conceptual event predicates are detected to identify activities, e.g.; crossover,

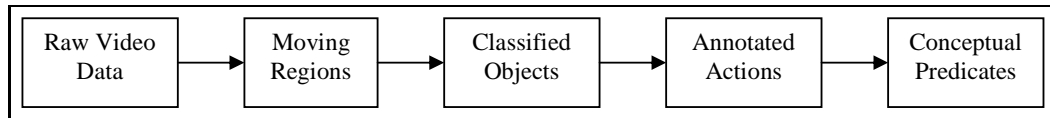


Figure 2.1: The semantic flow in information extraction.

deposit, approach. This information-extraction scheme, shown in Figure 2.1, is explained in the following subsections. *Annotated actions* and *conceptual predicates* are the primary aspects of our meta-data and used directly in the query model.

## 2.1 Keyframe Labeling

The keyframe labeling technique that we employ categorizes the keyframes into four primitive types, namely *join*, *split*, *move*, and *stop*. This keyframe labeling technique relies on moving region extraction and tracking steps where the extracted moving regions are tracked with respect to a grid-based representation, which we call *Inverted Tracking*. The appearances of the identified moving regions are stored in the *Motion Appearance Mask (MAM)* for each frame  $f$ , where a 1 in this mask represents the presence of a motion in that cell. The keyframe labeling computations for  $f$  are performed based on  $MAM_f$  and  $MAM_{f-1}$ . Hence, the keyframe labeling technique produces a temporal ordering of the keyframe labels as an event sequence that can be used to detect basic anomalies, such as crossover, deposit, and pickup. From a different perspective, it is a transformation from an image sequence representation to event sequence representation, where the anomalies can be considered as pre-defined event sequences. Thus, the problem reduces to detecting anomalous sequences in a set of sequences, which is much easier to process.

The pseudo-code of our keyframe labeling algorithm is given in Figure 2.2. At the first step, the moving regions are extracted for each frame. This step includes

```

Inputs:  $V$ , the input video stream;
            $t_s$ , the size threshold;
            $t_o$ , the temporal object appearance threshold;
            $t_d$ , the temporal detection threshold;
Outputs:  $K$ , the list of keyframe labels;

1. for each frame  $f$  in  $V$ 
   /* moving region extraction */
2. extract the set of moving regions  $R_f$ 
3. for each element  $r$  in  $R_f$ 
4.   if  $size(r) < t_s$ 
5.     remove  $r$  from  $R_f$ 
6.   else
7.     increment temporal-appearance( $r$ )
   /* inverted tracking */
8.     if temporal-appearance( $r$ )  $> t_o$ 
9.       set the  $cell(r)$  to 1 in  $MAM_f$ 
   /* keyframe detection */
10. if  $isKeyframe(MAM_f)$ 
11.   compute keyframe label  $l_f$ 
12.   increment temporal-count( $l_f$ )
13.   if temporal-count( $l_f$ )  $> t_d$ 
14.     push  $l_f$  onto  $K$ 
15.      $MAM_{f-1} \leftarrow M_f$ 
16.     clear  $M_f$ 
17. endfor

```

Figure 2.2: The pseudo-code of the keyframe labeling algorithm.

foreground extraction scheme where morphological operations are applied *a priori*. The morphological operations help us group the moving pixels into moving regions by using minimum and maximum object size filters. Through these operations, the salient regions are extracted at the end of the first step. At the next step, the motion appearance mask of the frame is computed and compared with that of the previous frame in order to detect whether the current frame is a keyframe. At the last step, the identified keyframe is labeled. Both in moving region detection and keyframe detection steps, temporal filtering is applied to minimize the effect of temporal noise. The temporal filtering is applied by holding the number of frames that the extracted region has appeared and the number

of frames that a keyframe is labeled, which are followed by a thresholding scheme.

## 2.2 Moving Region Extraction

The moving region extraction is one of the crucial parts of the keyframe labeling algorithm, since all of the computations are performed based on the extracted moving regions. We employ an adaptive background maintenance scheme to extract the moving regions, similar to the one proposed in [11]. We combine the scheme with three-frame differencing to detect the moving pixels. Then, we apply region grouping methods and morphological operations to these pixels to identify the moving regions.

This technique can be described 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 Equation 2.1:

$$M_f(x, y) = \begin{cases} 1, & \text{if } (|I_f(x, y) - I_{f-1}(x, y)| > T_f(x, y)) \text{ and} \\ & (|I_f(x, y) - I_{f-2}(x, y)| > T_f(x, y)) \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

The (moving) pixel intensities that are larger than the background intensities ( $B_f(x, y)$ ) are used to fill in the region of a moving object. This step requires a background maintenance task based on the previous intensity values of the pixels. Similarly, the threshold is updated based on the observed moving-pixel information in the current frame. A statistical background and threshold maintenance scheme is employed, as given in Equations 2.2-2.5:

$$B_0(x, y) = 0, \quad (2.2)$$

$$B_f(x, y) = \begin{cases} \alpha B_{f-1}(x, y) + (1 - \alpha) I_{f-1}(x, y), & M_f(x, y) = 0, \\ B_{f-1}(x, y), & M_f(x, y) = 1, \end{cases} \quad (2.3)$$

$$T_0(x, y) = 1, \quad (2.4)$$

$$T_f(x, y) = \begin{cases} \alpha T_{f-1}(x, y) + (1 - \alpha)(k \times |I_{f-1}(x, y) - B_{f-1}(x, y)|), & M_f(x, y) = 0, \\ T_{f-1}(x, y), & M_f(x, y) = 1, \end{cases} \quad (2.5)$$

where  $\alpha$  is the learning constant and the constant  $k$  is set to 5 in Equation 2.5 [11].

We employ a view-based motion tracking approach similar to the Motion History Image (MHI) technique proposed in [4]. MHI detects and tracks the parameters (i.e., structure and orientation) of the moving regions. In an MHI, the pixel intensity is encoded as a function of the temporal history of the motion at that pixel, where the pixels that moved more recently are brighter.  $MHI_f(x, y)$  of  $f$  is constructed by the update rule in Equation 2.6:

$$MHI_f(x, y) = \begin{cases} \tau, & M_f(x, y) = 1, \\ \max(0, MHI_{f-1}(x, y) - 1), & M_f(x, y) = 0, \end{cases} \quad (2.6)$$

where  $\tau$  denotes the temporal extent of a motion.

A set of filters is applied to reduce the effect of noise in moving region detection. First of all, a distance filter is applied to the extracted moving regions such that the closer regions are joined. The distance threshold for this operation is adjusted with respect to the perimeter of the smaller region to be joined. As the next step, a size filtering is applied to the moving regions, which filters out the regions below the size threshold  $t_s$ . The size filtering that we apply is based on both the area and perimeter of a region.

The last filtering scheme that we employ is temporal filtering. The temporal appearance of a moving region is counted and the region is filtered out if it fails to be present in a pre-defined number of frames. This temporal threshold  $t_o$  is computed with respect to the frame rate of the input stream in order to be consistent in terms of seconds. For example, for a temporal threshold duration

of 0.25 seconds,  $t_o$  will be 6 if the input video stream is 24 frames-per-second. To elaborate further, the noise reduction filters that we applied are shown in Figure 2.3 on a sample frame. In Figure 2.3 (a) and (b), the original frame and its corresponding processed version at time  $t_1$  are shown, respectively. The object in the bottom-right corner in Figure 2.3 (b) is failed to pass the temporal filter since it has not appeared in sufficient number of frames. However, as shown in Figure 2.3 (c), the same object is extracted since it passed the temporal filter  $t_o$ . On the other hand, the smallest rectangle without an object label in the mid-left part of the scene in Figure 2.3 (c) is failed to pass the size threshold.

## 2.3 Inverted Tracking

The *inverted tracking* scheme provides a way to keep track of the object appearances on a video. The term “inverted” is generally used to imply that a mapping or an index from the content is stored along with the actual content. Here, the actual content is the extracted moving regions and a view-based mapping is held along with the content. In this scheme, instead of processing moving regions, the corresponding inverted index is processed for semantic analysis of the input stream. In other words, the inverted tracking scheme constitutes a data model for our keyframe labeling algorithm.

Figure 2.4 illustrates the inverted-tracking technique that we employ on a sample video frame. The video frame  $I(x, y)$  is divided into a pre-defined number of cells corresponding to the subdivisions in  $x$  and  $y$  directions. In Figure 2.4 (b), an  $8 \times 8$  grid is used to hold the moving object appearances. The moving object appearance data is represented in MAM as shown in Figure 2.4 (c).

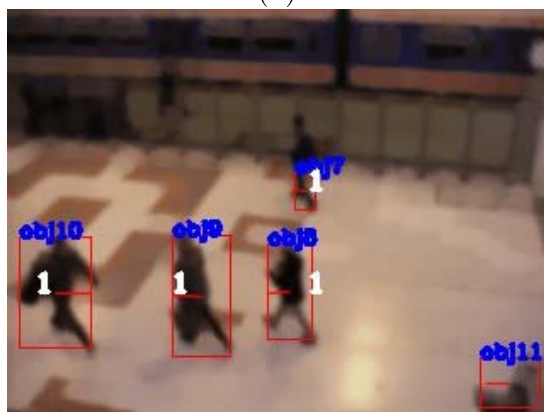
While computing the moving object appearance within a cell, we consider the center-of-mass ( $c_m$ ) of the region corresponding to the moving object.  $c_m = (x_{c_m}, y_{c_m})$  is computed as in Equation 2.7:

$$x_{c_m} = \frac{\sum_i^n x_i}{n}, y_{c_m} = \frac{\sum_i^n y_i}{n}, \quad (2.7)$$

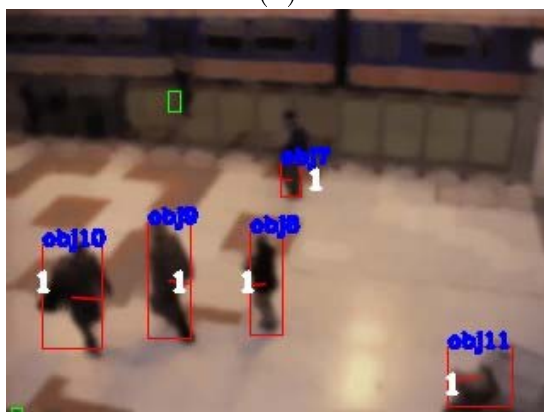
where  $n$  is the total number of pixels in a region.



(a)



(b)



(c)

Figure 2.3: The noise reduction filters applied on a sample frame. (a) Original frame from PETS 2006 S1-T1-C dataset [2] at  $t_1$ , (b) Processed frame at  $t_1$ , and (c) Processed frame at  $t_2 = t_1 + 0.25$  seconds.



(a)



(b)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(c)

Figure 2.4: Inverted tracking on a sample frame. (a) Original frame from PETS 2006 S1-T1-C dataset [2], (b) Processed frame for  $8 \times 8$  grid, and (c) The Motion Appearance Mask (MAM) of the frame.



**Definition 2.1 (Appearance within a Cell)** *The region  $r$  has appeared in a cell  $i$  if  $c_m$  of  $r$  is inside the boundaries of  $i$ , inclusively. To break ties, the boundaries are assumed to belong to the cell on the left and down.*

Since more than one object might appear in a cell, the technique holds object lists for each cell. If an object is moving in a cell  $i$ , it is kept in the object-appearance list of the cell  $i$  until it leaves the boundaries of the cell. If the object stops, it is not dropped from the list until a certain amount of time has passed. A specific type of motion where an object enters the scene, stops for a certain time, and then leaves the scene is called *loitering* [3]; this activity is considered a potential abnormal situation. Hence, our inverted tracking scheme does not drop objects until the maximum amount of time (generally taken as 60 seconds) has passed to detect loitering, simply by tracing the object appearance lists. However, if the object stops and then starts moving before loitering happened, we can detect that the previously stopped region is moving again by tracing back the object list belonging to that cell. Based on the assumption that the objects will be seen after occlusion before loitering, this delayed dropping of objects from appearance lists also helps handle the occlusion problem of tracking and cope with object tracking errors.

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

The most challenging issue in this scheme is the selection of the grid size. Object-based heuristics could be applied to select the grid size as a function of the smallest or the largest object/region size. These heuristics might work, but in our opinion, the grid size has to be selected in a way that is independent of the pixel-level parameters (e.g., size, perimeter). The main reason is that the inverted tracking scheme is a view-based data model on top of the pixel-level, and hence, the view-based scheme has more adaptive processing capabilities on various

datasets when loosely coupled with the pixel-level representation. Another reason is the fact that using a variable grid size based on the pixel-level parameters of the objects makes on-the-fly processing complex. Hence, we used fixed grid sizes in our experiments.

## 2.4 Keyframe Detection

The keyframe detection scheme operates only on the motion appearance masks of the current frame and the previous frame, which makes the processing easier. The pseudo-code of the keyframe detection algorithm is shown in Figure 2.5. This algorithm corresponds to the steps 10 and 11 of the keyframe labeling algorithm shown in Figure 2.2, where a temporal filtering is applied after the keyframe detection to reduce the mis-detection rate based on sudden changes in  $MAM_f$ .

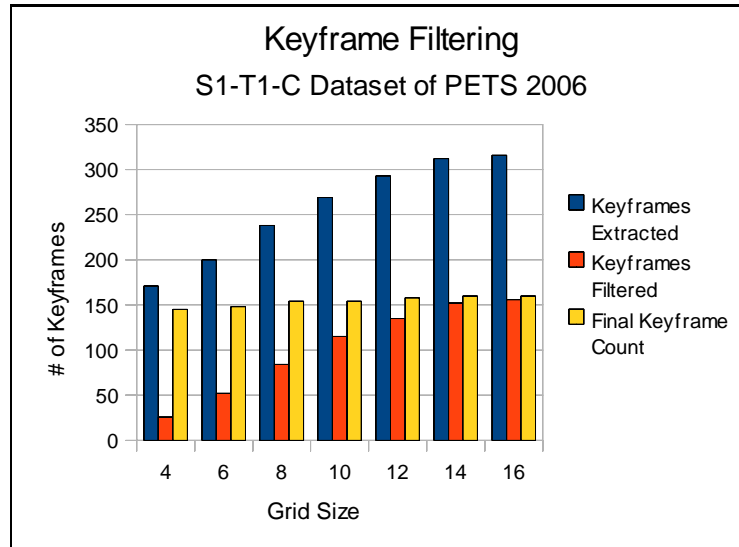
<p><b>Inputs:</b> <math>f</math>, the frame of the input video stream;  <math>MAM_f</math>, the motion appearance mask of <math>f</math>;  <math>MAM_{f-1}</math>, the motion appearance mask of <math>f-1</math>;  <math>t_{stop}</math>, the temporal threshold for detecting stop;  <math> MAM_f </math> denotes the total number of 1s in <math>MAM_f</math>;</p> <p><b>Outputs:</b> <math>l_f</math>, the label of the keyframe;</p> <ol style="list-style-type: none"> <li>1.       if <math> MAM_f  &gt;  MAM_{f-1} </math></li> <li>2.         <math>l_f = \text{SPLIT}</math></li> <li>3.       else if <math> MAM_f  &lt;  MAM_{f-1} </math></li> <li>4.         <math>l_f = \text{JOIN}</math></li> <li>5.       else if <math>MAM_f \wedge MAM_{f-1} \neq 0</math></li> <li>6.         <math>l_f = \text{MOVE}</math></li> <li>7.       else /* <math>MAM_f = MAM_{f-1}</math> */</li> <li>8.         stop-count <math>\leftarrow</math> stop-count +1</li> <li>9.         if stop-count <math>&gt; t_{stop}</math></li> <li>10.        <math>l_f = \text{STOP}</math></li> </ol>
---

Figure 2.5: The pseudo-code of the keyframe detection algorithm.

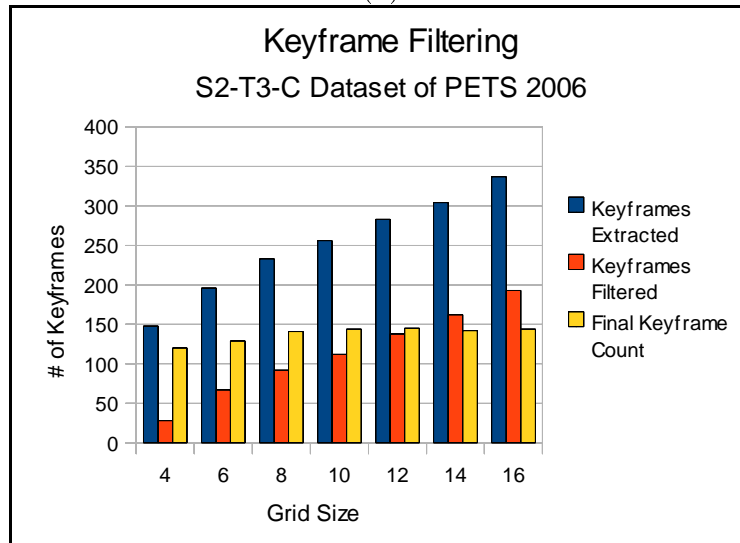
**Definition 2.2 (Keyframe)** *Frame  $f$  is a keyframe if  $(MAM_f \neq MAM_{f-1}) \vee (MAM_f = MAM_{f-1} = \dots = MAM_{f-k} \text{ if } k \geq f_s)$ , where  $MAM_f$  denotes the*

*Motion Appearance Mask of  $f$ , and  $f_s$  denotes the maximum allowed number of frames without motion.*

We employed a filtering mechanism to reduce the number of keyframes further in such a way that if the current and previous frames are labeled as `MOVE`, the current frame is not labeled as a keyframe. Figure 2.6 shows the effect of this keyframe filtering step on a sample video. As expected, the number of extracted frames without filtering increases in parallel with the grid size due to the number of frames with `MOVE` label. However, the final keyframe count is almost constant for various grid sizes when this filtering is applied. Since the labels assigned to the keyframes do not change with this filter, the impact of the grid size to the keyframe labeling algorithm is minimized without degrading the detection performance.



(a)



(b)

Figure 2.6: Keyframe filtering applied on PETS 2006 [2] datasets (a) S1-T1-C, originally has 3012 frames, 154 keyframes are extracted on the average; and (b) S2-T3-C, originally has 2551 frames, 138 keyframes are extracted on the average.

## Chapter 3

# Histogram-Based Shape and Color

Histogram-Based Approach (HBA) is proposed to encode the shape and color low-level features of the moving objects identified by the preprocessing steps discussed in Chapter 2. HBA is basically motivated with some of the basic principles of human visual system. The human visual system identifies objects with the edges they contain, both on the boundary and in the interior, based on the intensity differences among pixels [8]. These intensity differences are captured with respect to the center of mass of these pixels.

HBA processes the information based on the interrelation among pixels to encode shape and color content. This processing is similar to the human visual system in the sense that it considers the pixels in the interior and on the boundary of the objects with respect to the center of mass. In this scheme, each pixel provides a piece of information about the shape and color content of the objects [38].

The low-level feature vectors generated by HBA can be used in many ways. First of all, HBA can be used for content-based querying and retrieval of image and video data according to the shape and color content [14, 39]. In this thesis, this scheme is also used for the object-level meta-data extraction process of

the scenario-based querying framework. The shape histograms are also used for obtaining class values for moving objects.

### 3.1 Histogram Construction

There are three histograms for encoding shape and color information in HBA, where two of them are used to store the shape content of the objects, and one of them is used for the color content. The three histograms can be described as follows:

**Distance Histogram** stores the Euclidean distance between the center of mass of an object ( $c_m$ ) and all of the pixels within the object. The distance between a pixel  $p$  and  $c_m$  is re-scaled with respect to the maximum distance  $d_{max}$  among the pixels (i.e., the distance of the farthest pair of pixels). The scaled distance accumulates into the corresponding bin in the distance histogram. In other words, instead of a constant scale for the objects, an object-specific scale metric is employed, which eases to satisfy the scale invariance by keeping the total number of bins in the distance histogram constant for all of the objects. We set the dimensionality  $\lambda$  of the distance histogram to 10 for storing pixel-wise distance information with respect to 10 levels.

**Angle Histogram** stores the counter-clockwise angle between pixel vectors and the unit vector on the  $x$ -axis ( $e_x$ ). The pixel vector  $v_p$  for a pixel  $p$  is a vector directed from  $c_m$  to  $p$ . The unit vector  $e_x$  is translated to  $c_m$ . As illustrated in Figure 3.1,  $\alpha$  is the counter-clockwise angle, polar angle, for  $p_i$  and increments the corresponding bin in the angle histogram. This type of information storage is novel since it provides an easy and intuitive way to capture angular distribution of the pixels around a fixed object point ( $c_m$ ). We set the dimensionality  $\lambda$  of the angle histogram to 36 with a swapping angle of  $10^\circ$ .

**Color Histogram** stores the three-dimensional vector  $c_i = (h_i, s_i, v_i)$  in HSV

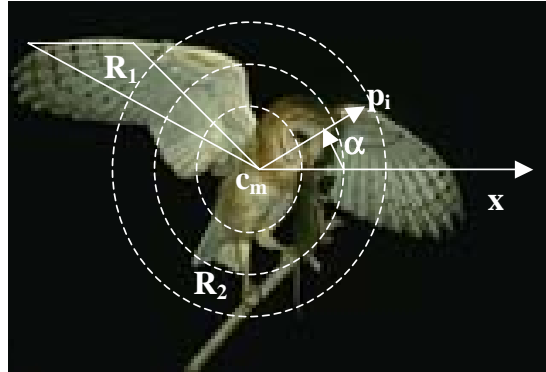


Figure 3.1: Angle histogram calculations.  $\alpha$  is the polar angle for object pixel  $p_i$ .  $R_1$  region is an equivalence pixel class for angle histogram, and  $R_2$  region is an equivalence pixel class for distance histogram.

color space for all of the pixels  $p_i$  belonging to an object. We have quantized the HSV color space as follows: A circular quantization of  $20^\circ$  steps sufficiently separates the hues such that the six primary colors are represented with three sub-divisions. Saturation and intensity are quantized to 3 levels leading to a proper perceptual tolerance along these dimensions. Hence, 18 hues, 3 saturations, 3 intensities, and 4 gray levels are encoded yielding 166 unique colors; i.e., a color feature vector of  $\lambda = 166$  dimensions (see [43] for details). To improve the effectiveness of the histogram, the contributions of the pixels are weighted according to a probabilistic measure in a way similar to the one proposed in [6]. A distance weighting-scheme is employed in the calculation of the pixel weights from the neighborhood, in which the effects of the closer pixels are increased. The probability of the occurrence of a color within its some neighborhood,  $P$ , is a good estimate for the local color activity within some neighborhood. If the neighborhood of a color is uniform, then  $P$  is higher, and vice versa. In this approach, the contribution of uniform regions is decreased and that of singular points is increased. The computations are based on the following equation:

$$w_p = \frac{w_c}{N_p(c)}, \quad (3.1)$$

where  $w_p$  is the weight of pixel  $p$ , and  $N_p(c)$  is the number of pixels having color  $c$  as  $p$  within the neighborhood of  $p$ .  $w_c$  is the weight assigned to the unique color  $c$ . The color weights can be set to 1, but to embed global color information to the equation,  $w_c$  can be computed as:

$$w_c = \frac{N_c}{N_T}, \quad (3.2)$$

where  $N_c$  is the number of pixels having color  $c$ , and  $N_T$  is the area of the object.

In HBA, the pixel weights are also scaled with respect to the distances of the pixels within the neighborhood of a pixel. Rectangular or octagonal  $n \times n$  neighborhood settings are valid for our approach. Having performed a reasonable amount of experiments, the neighborhood of the pixels is chosen as  $5 \times 5$  octagonal windows, since the use of octagonal neighborhoods is more intuitive from the distance point of view.

The two histograms that store shape information are constructed with respect to the center of mass ( $c_m$ ) of the object. The reasons for selecting  $c_m$  are given in [38].

The three histograms are scale, rotation and translation invariant. The scale invariance is provided by re-scaling with respect to the maximum object distance (i.e., the distance of the farthest pair of pixels in the object) for distance histogram, by dividing each bin by the total number of pixels for angle histogram. The weighted storage of the pixels (see Equation 3.1) provides scale invariance for color histogram.

The translation invariance of the histograms is quite obvious and the rotation invariance is an issue for the angle histogram only. For rotation normalization, the major axis of the object is found by traversing the boundary of the object. The major axis is the line segment connecting the farthest pair of pixels [52]. Then, the angle  $\alpha_{MA}$  between the major axis and the x-axis is computed (for the object in Figure 3.1,  $\alpha_{MA} = 150^\circ$ ). It can be observed that  $0^\circ \leq \alpha_{MA} \leq 180^\circ$ . The object is then rotated clockwise so that the major axis becomes horizontal, i.e.,



the object is rotated  $-\alpha_{MA}$  degrees. Since the angle histogram is filled according to this rotated form of the object, the angle histogram is rotation invariant.

## 3.2 Similarity Calculation

We used the *histogram intersection* technique [47] for the similarity calculations between test objects and trained objects. In this technique, two normalized histograms (i.e., combined feature vectors) are intersected as a whole to end up with a distance value. Both of the histograms must be of the same size, and the distance between the histograms is a floating point number between 0 and 1. Equivalence is designated with similarity value 1, and the similarity between two histograms decreases in parallel with the similarity value approaches to 0.

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

$$S_{H_1, H_2} = \frac{\sum_i^n \min(H_1[i], H_2[i])}{\min(|H_1|, |H_2|)}, \quad (3.3)$$

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

In HBA, since the extracted low-level object content is composed of three separate histograms, the total similarity between the query object and a database object is determined as follows:

$$S_T = \frac{S_C \times w_C + S_D \times w_D + S_A \times w_A}{w_C + w_D + w_A}, \quad (3.4)$$

where  $S_C$ ,  $S_D$ , and  $S_A$  denote similarities in color, distance, and shape histograms, computed by histogram intersection, respectively. In this equation, color, distance, and angle histograms are linearly combined with pre-specified weights  $w_C$ ,  $w_D$ , and  $w_A$ , respectively.

The color and shape features, as being the most frequent descriptors, are used in most of the existing systems in a combined way. The main reason for this combined usage is to improve the retrieval effectiveness. A global distance

value can be obtained by linear combination of three partial distances with appropriate weights [24] during the classification process. A possible set of weights can be determined by performing similarity calculations for each of the two feature vectors separately. The average-case retrieval accuracies as a result of these two similarity calculations are normalized and used as feature weights in the linear combination. This pre-computed weight assignment provides more effective results since this approach reflects the characteristics of the datasets.

# Chapter 4

## Meta-Data Extraction

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

**Object-based meta-data:** The object-based facts contain the class value of the object as well as high-level color and shape descriptors (see `<classdesc>`, `<colordesc>`, and `<shapedesc>` in Appendix A.1). The object class type is determined by the classification algorithm. High-level feature descriptors are determined by performing similarity calculations between the feature vectors of the objects and the pre-defined vectors corresponding to a pre-defined set of colors and shape primitives. Since the color and shape feature vectors of the objects may change in time (especially color), this object information is stored when a keyframe is obtained. Hence, keyframe number

and cell id are stored along with the fact. The fact about object information is as follows:

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

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

```
event-info(video-id, event-label, keyframe-number, cell-id).
```

```
single-object-event-label(video-id, object-id,
                          keyframe-number, cell-id).
```

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

We used an interval-based extension scheme to utilize the fact storage mechanism. If the same event is triggered for at least two consecutive keyframes, a keyframe interval is stored with the event fact instead of separate facts with consecutive keyframe numbers. This interval extension is applied for all facts where it is applicable. This type of fact storage reduces the storage costs of the system significantly. Query processing mechanism is also capable of processing interval based fact storage. A sample listing for the facts-base to clarify the meta-data storage mechanism that we use is given in Appendix A.3.

## 4.1 Object-based Meta-Data Extraction

The object-based meta-data contains high-level color and shape descriptors and a class value for the objects in our scenario-based querying framework. Due to the nature of video surveillance, the specification of complex queries on the shape and color content within scenarios is quite uncommon. An operator might simply query the abnormalities with a few keywords describing the color and shape information of the object; e.g., *man with a black coat, box-shaped blue bag*. However, the proposed Histogram-Based Approach (HBA) can be utilized for searches based on the color and/or shape content of the objects. HBA is used to obtain high-level descriptors for color and shape information, as well as utilized in the classification scheme to obtain a class label.

### 4.1.1 Color Descriptor

A set of primary colors is chosen to obtain a high-level color descriptor. The primary colors are encoded in a color histogram, and the histogram intersection technique is used to obtain the similarities between the primary color histograms and the color histogram of the object. The most similar primary color label is chosen as the high-level color descriptor for the object. The label corresponding to the primary color of the object is used as the color meta-data.

### 4.1.2 Shape Descriptor

High-level shape descriptors are obtained in a similar way as color descriptors. A set of primitive shapes that an operator may use to label an object is chosen to obtain a high-level shape descriptor. The shape histograms are constructed for these shape elements. Since our feature vectors are scale and rotation invariant, sample figures for *box*, *cone*, *cylinder*, and *sphere* are used to encode the shape information. The most similar pre-defined shape element is chosen as the high-level shape descriptor for the object. A more standard way of specifying high-level

shape descriptors might be used to increase the expressive power of the queries.

### 4.1.3 Class Descriptor

The objects are categorized into only three classes: *human*, *non-human*, and *object-group*. The low-level color and shape features extracted are used to obtain a class type. The color vector stores the distance-weighted intensity values in order to take the color distribution around a pixel into account. This type of color-feature encoding is different from traditional color vectors, and as shown in [38], aids the object-classification process. The shape vector is a composition of a region's *angular* and *distance* spans ([38, 39]), which are computed based on the region's center of mass ( $c_m$ ). These encoding schemes are invariant under scale, rotation, and translation, and have an effective expressive power [39].

In our  $k$ -nearest neighbors based classification scheme, the color and shape information can be linearly combined with proper weights and we use the temporal averages of these vectors computed during tracking. A global distance value can be obtained by linear combination of three partial distances with appropriate weights during the classification process [24]. A possible set of weights can be determined by performing similarity calculations for each of the two feature vectors separately. The average-case retrieval accuracies as a result of these two similarity calculations are normalized and used as feature weights in the linear combination. This pre-computed weight assignment provides more effective results since this approach reflects the characteristics of the datasets.

We have used the PETS 2004 [1] and PETS 2006 [2] benchmark datasets for the training phases. 918 *human*, 94 *non-human*, and 133 *object-group* samples from PETS 2004 dataset; and 1674 *human*, 313 *non-human*, and 492 *object-group* samples from PETS 2006 dataset are selected. The number of nearest neighbors ( $k$ ) is chosen as 10, and a fivefold cross validation method is employed, where the 20% of the samples are used each time for testing. The preprocessing steps include the extraction of the moving regions and their corresponding color and shape vectors. The weights of the color feature vectors are found to be lower than

the other two feature vectors (i.e., distance and angular spans), as expected, since the color information is less discriminative in video surveillance domain. The classification algorithm outputs percentages for the class types (e.g., 47% human, 34% non-human, 19% object-group), and the moving region is classified as the type with the highest percentage value. Effective clustering techniques to further split the object groups can be embedded to improve the classification scheme.

## 4.2 Event-based Meta-Data Extraction

Having classified the moving objects in a video sequence, we annotated the actions of objects automatically. Counting the number of moving objects gives an important clue about the annotation of an event, since the number of moving objects changes at the time of an event. Hence, we utilized a keyframe-based annotation process, where a keyframe is identified when the total number of moving objects changes or the cell-id of an object changes. This keyframe-based processing provides an easier way for detecting conceptual abnormalities. As shown in [13], this type of annotation and meta-data usage reduces the search space and enables efficient querying when the video archives are large.

### 4.2.1 Single Object Annotation

Our system detects the following basic single-object actions: *enter*, *leave*, *stop*, *stop-and-go*, and eight directional forms of *move* coupled with the directional predicates (see `<direction>` in Appendix A.1). The orientation of a moving region, as suggested by our tracking algorithm, is used to directly annotate move actions. When an object stops and moves again later, we annotate the action as a stop-and-go type after detecting the object's next move within a certain time interval. If this type of action takes more time than allowed, then we annotate the action as loitering. The other types of single-object actions are rather easier to detect using the inverted tracking algorithm.

### 4.2.2 Multi-Object Annotation

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

The keyframe labeling algorithm (see Figure 2.2) that we employ transforms the input video stream to a textual representation of sequence of events in the video. The steps dealing with moving regions are performed once and a textual representation of the video with a relatively smaller size is achieved. The anomaly detection, then, becomes detecting a sequence of event labels in this input sequence. We devise three finite state automata for *crossover*, *move-together*, *deposit*, and *pick up*. Formally, a deterministic Finite State Automaton (FSA) is denoted as a quintuple  $(\Sigma, S, S_0, \delta, F)$ , where  $\Sigma$  is the input alphabet (a finite, non-empty set of symbols);  $S$  is a finite, non-empty set of states;  $S_0$  is an initial state where  $S_0 \in S$ ;  $\delta$  is the state transition function such that  $\delta : S \times \Sigma \rightarrow S$ ; and  $F$  is the set of final states, where  $F \subset S$ . The finite state automata that we devised for *crossover*, *move-together*, *deposit*, and *pickup* will be discussed based on this notation. The input to these finite state automata is the sequence of keyframe labels representing the input video stream. Reasonable detection accuracies are achieved in our experiments, which is important for using keyframe labeling technique in anomaly detection. Having detected the anomalies, the facts-base is populated accordingly and a real-time alerting scheme is triggered.



### 4.2.2.1 Crossover and Move-Together

A *crossover* situation occurs when at least two moving objects move in opposite directions and cross each other, whereas a *move-together* situation occurs when the objects move in the same direction and the faster object passes the slower object. These situations can be extended for more than two objects in a similar manner. In both situations, tracking the moving objects based on the locations to detect crossover situation requires high processing cost. The FSA-based approach operates in a more effective way since the input size is reduced. When detection is of concern, crossover and move-together can be detected by a single FSA-based mechanism, with the only difference that the motion direction of the objects at the time of occlusion has to be taken into consideration.

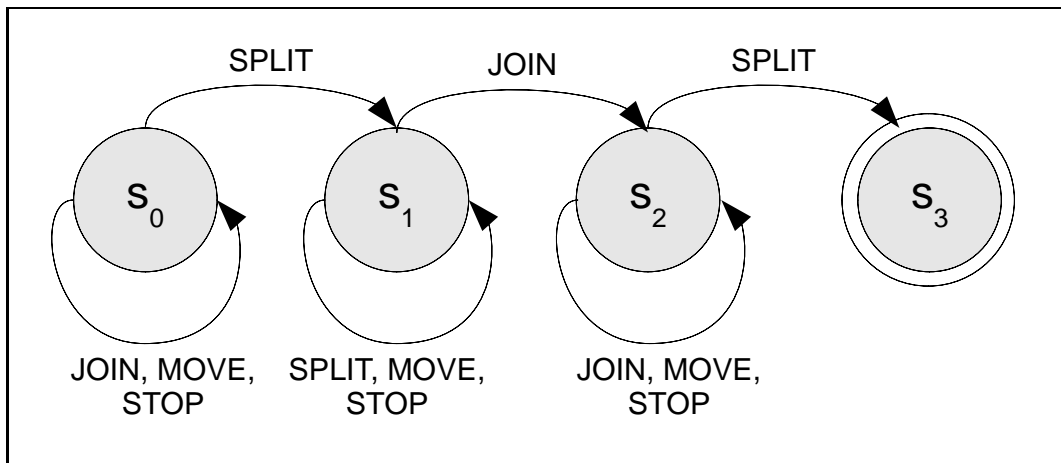


Figure 4.1: The FSA for recognizing event sequence for crossover and move-together.

Let  $FSA_C = (\Sigma, S_C, S_0, \delta_C, F_C)$  represent the finite state automaton detecting crossover and move-together event occurrences  $S_C = \{S_0, S_1, S_2, S_3\}$ ,  $\Sigma = \{JOIN, SPLIT, MOVE, STOP\}$ , and  $F_C = \{S_3\}$ . Figure 4.1 presents  $FSA_C$  and Table 4.1 gives the state transition function  $\delta_C$ .

A sample crossover detection by  $FSA_C$  on a sample video of PETS 2006 dataset [2] is shown in Figure 4.4. At the beginning, the active state  $s_c$  of  $FSA_C$

	SPLIT	JOIN	MOVE	STOP
$S_0$	$S_1$	$S_0$	$S_0$	$S_0$
$S_1$	$S_1$	$S_2$	$S_1$	$S_1$
$S_2$	$S_3$	$S_2$	$S_2$	$S_2$
$S_3$	$S_0$	$S_0$	$S_0$	$S_0$

Table 4.1: The state transition function  $\delta_C$  for the automaton detecting crossover and move-together.  $S_0$  is the initial state, and  $S_3$  is the final state accepting event sequence for crossover and move-together.

is at  $S_0$ . When the objects shown in Figure 4.4 (a) enters the scene,  $s_c$  becomes  $S_1$ , and eventually  $|MAM_f| = 2$ , where  $|MAM_f|$  denotes the total number of 1s in  $MAM_f$ . When the execution reaches at Figure 4.4 (b),  $|MAM_f| = 1$  and  $|MAM_{f-1}| = 2$ , which signals JOIN, hence  $s_c$  reaches at  $S_2$ . Finally in Figure 4.4 (c),  $|MAM_f| = 2$  again and  $|MAM_{f-1}| = 1$ , which signals SPLIT and makes  $s_c$  reach the final state  $S_3$ .

#### 4.2.2.2 Deposit

A *deposit* situation occurs when a moving object leaves a relatively smaller object (e.g., suitcase, bag) in the video scene. Effective motion models are required for detecting the moving object's action.

	SPLIT	JOIN	MOVE	STOP
$S_0$	$S_0$	$S_0$	$S_1$	$S_0$
$S_1$	$S_2$	$S_0$	$S_1$	$S_1$
$S_2$	$S_0$	$S_0$	$S_3$	$S_3$
$S_3$	$S_0$	$S_0$	$S_0$	$S_0$

Table 4.2: The state transition function  $\delta_D$  for the automaton detecting deposit.  $S_0$  is the initial state, and  $S_3$  is the final state accepting event sequence for deposit.

Let  $FSA_D(\Sigma, S_D, S_0, \delta_D, F_D)$  represent the finite state automaton detecting deposit event occurrences  $S_D = \{S_0, S_1, S_2, S_3\}$ ,  $\Sigma = \{JOIN, SPLIT, MOVE, STOP\}$ , and  $F_D = \{S_3\}$ . Figure 4.2 presents  $FSA_D$  and Table 4.2 gives the state transition function  $\delta_D$ .

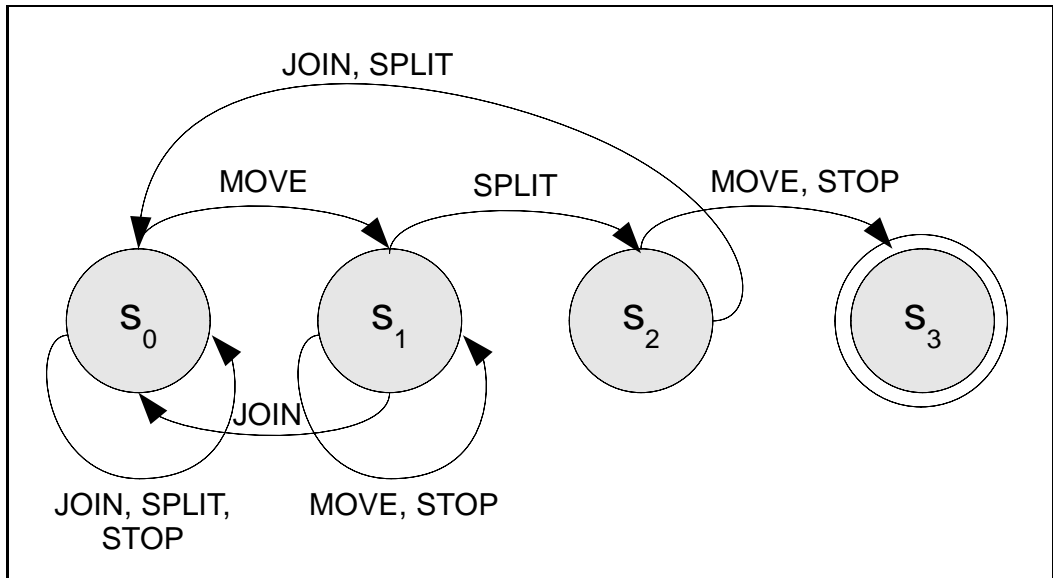


Figure 4.2: The FSA for recognizing event sequence for deposit.

A sample deposit detection by  $FSA_D$  on a sample video of PETS 2004 dataset [1] is shown in Figure 4.5. At the beginning, the active state  $s_d$  of  $FSA_D$  is at  $S_0$ . When the object shown in Figure 4.5 (a) enters the scene,  $s_d$  becomes  $S_1$ , and eventually  $|MAM_f| = 1$ . When the execution reaches at Figure 4.5 (b),  $|MAM_f| = 2$  and  $|MAM_{f-1}| = 1$ , which signals **SPLIT**, hence  $s_d$  reaches at  $S_2$ . Finally in Figure 4.5 (c), **MOVE** is detected when  $|MAM_f| = 2$  and  $|MAM_{f-1}| = 2$ , which makes  $s_d$  reach the final state  $S_3$ .

#### 4.2.2.3 Pickup

*Pickup* can be considered as the dual of deposit; thus a pickup situation occurs when a moving object picks up a relatively smaller object (e.g., suitcase, bag) in the video scene. Similarly, effective motion models are required for detecting the moving object's action.

Let  $FSA_P = (\Sigma, S_P, S_0, \delta_P, F_P)$  represent the finite state automaton detecting pickup event occurrences  $S_P = \{S_0, S_1, S_2, S_3\}$ ,  $\Sigma = \{JOIN, SPLIT, MOVE, STOP\}$ , and  $F_P = \{S_3\}$ . Figure 4.3 presents  $FSA_P$  and Table 4.3 gives the state

	SPLIT	JOIN	MOVE	STOP
$S_0$	$S_0$	$S_0$	$S_1$	$S_0$
$S_1$	$S_0$	$S_0$	$S_1$	$S_2$
$S_2$	$S_2$	$S_2$	$S_3$	$S_2$
$S_3$	$S_0$	$S_0$	$S_0$	$S_0$

Table 4.3: The state transition function  $\delta_P$  for the automaton detecting pickup.  $S_0$  is the initial state, and  $S_3$  is the final state accepting event sequence for pickup.

transition function  $\delta_P$ .

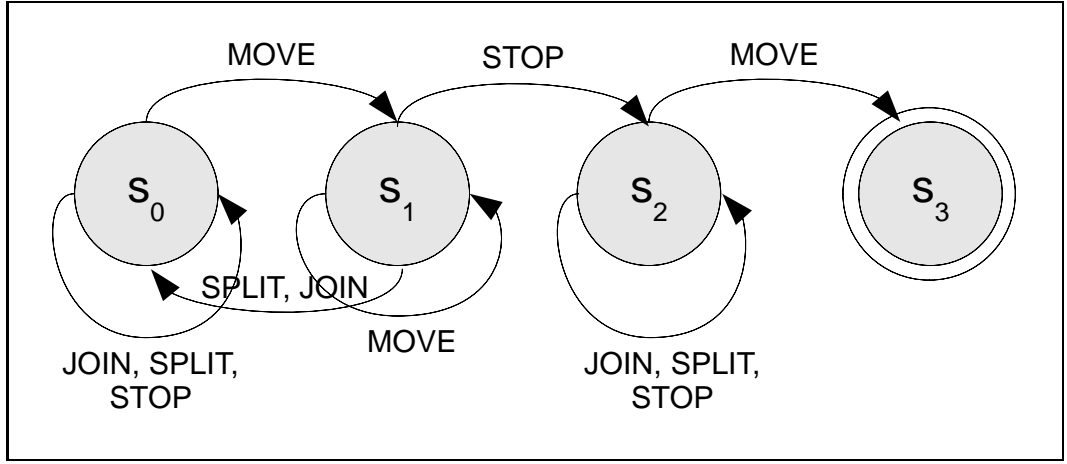
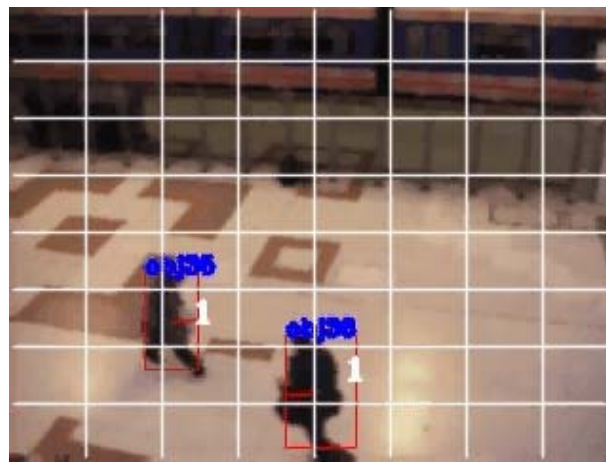


Figure 4.3: The FSA for recognizing event sequence for pickup.

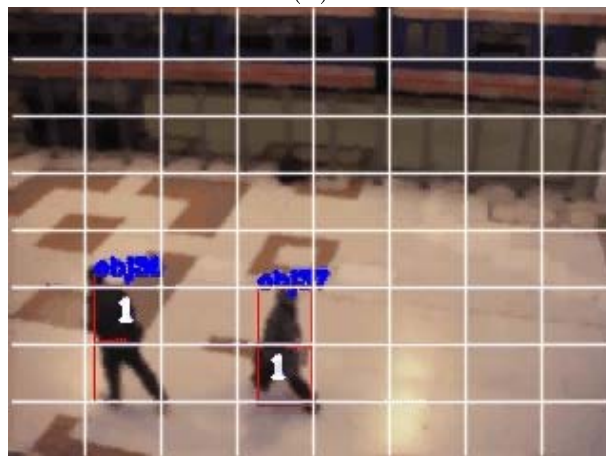
A sample pickup detection by  $FSA_P$  on a sample video of PETS 2004 dataset [1] is shown in Figure 4.6. At the beginning, the active state  $s_p$  of  $FSA_P$  is at  $S_0$ . When the object shown in Figure 4.6 (a) enters the scene,  $s_d$  becomes  $S_1$ , and eventually  $|MAM_f| = 1$ . When the execution reaches at Figure 4.6 (b),  $|MAM_f| = 1$  but sufficiently many frames have passed for  $t_{stop}$ , which signals STOP, hence  $s_d$  reaches at  $S_2$ . Finally in Figure 4.6 (c), MOVE is detected when  $|MAM_f| = 1$  and  $|MAM_{f-1}| = 1$ , which makes  $s_d$  reach the final state  $S_3$ .



(a)



(b)

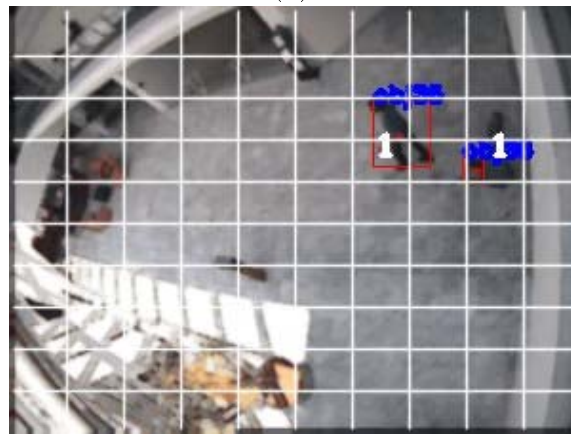


(c)

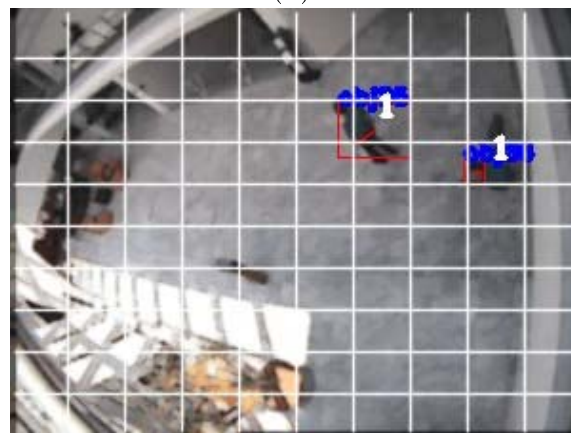
Figure 4.4: FSA state transitions for crossover detection for PETS 2006 S1-T1-C dataset [2]. (a) Transition from  $S_0$  to  $S_1$ , (b) Transition from  $S_1$  to  $S_2$ , and (c) Transition from  $S_2$  to  $S_3$  and crossover detection.



(a)



(b)

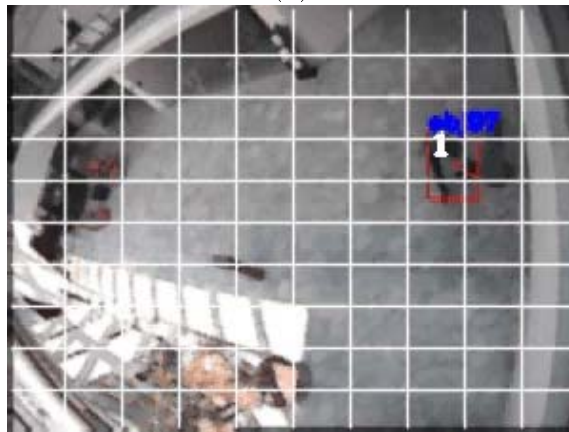


(c)

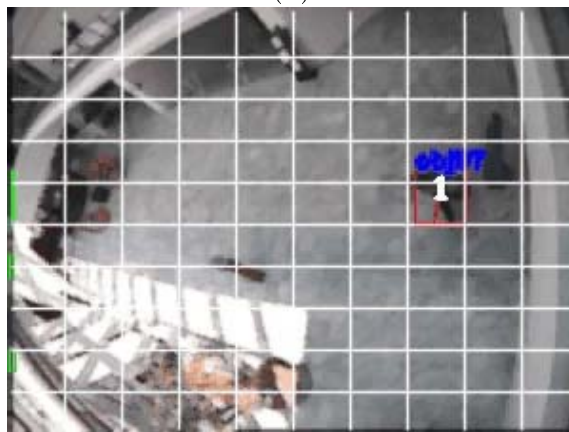
Figure 4.5: FSA state transitions for deposit detection for PETS 2004 leftbag dataset [1]. (a) Transition from  $S_0$  to  $S_1$ , (b) Transition from  $S_1$  to  $S_2$ , and (c) Transition from  $S_2$  to  $S_3$  and deposit detection.



(a)



(b)



(c)

Figure 4.6: FSA state transitions for pickup detection for PETS 2004 leftbag dataset [1]. (a) Transition from  $S_0$  to  $S_1$ , (b) Transition from  $S_1$  to  $S_2$ , and (c) Transition from  $S_2$  to  $S_3$  and pickup detection.

### 4.2.3 Post-Detection Process

Post-processing is applied after the detection process in event-based meta-data extraction. This step populates the fact-base by determining the objects participating in an abnormal event, motion directions of the objects, etc. The directional relations between the moving objects are handled by the help of the inverted tracking scheme. Single object facts are added directly and a post-processing step is required for multi-object events. If more than two objects are detected in a keyframe, the multi-object actions (predicates) are extracted for all of the object pairs. Single-object predicates are also extracted throughout the appearance of the objects in the scene. The following sequence of operations for two moving objects illustrates the post-detection process for event-based meta-data extraction:

- Two objects  $O_1$  and  $O_2$  are identified. The annotation scheme throws  $enter(O_1)$  and  $enter(O_2)$ .
- On the next keyframe, *move* predicates are thrown with the valid directions for  $O_1$  and  $O_2$ . Based on the Euclidean distance between  $O_1$  and  $O_2$ , *approach* or *depart* predicate is thrown.
- If  $FSA_D$  or  $FSA_P$  accepted a sequence of keyframe labels a priori, and  $O_1$  and  $O_2$  are classified as human and non-human, respectively, then a *deposit* or *pick-up* predicate is stored (see Figure 4.7 (a) and (b)).
- If  $FSA_C$  accepted a sequence of keyframe labels a priori, and both  $O_1$  and  $O_2$  are human, then *crossover* and/or *move-together* fact is extracted depending on the humans' orientations (see Figure 4.7 (c) and (d)).
- If  $FSA_C$  accepted a sequence of keyframe labels a priori, and both  $O_1$  and  $O_2$  are non-human, the event could be *move-together*, assuming that the two objects are thrown by a human outside the camera's field of view.
- To improve the accuracy of retrieval, the directional relations for multi-object actions are also identified (e.g.,  $deposit(O_1, O_2, south)$  is extracted).



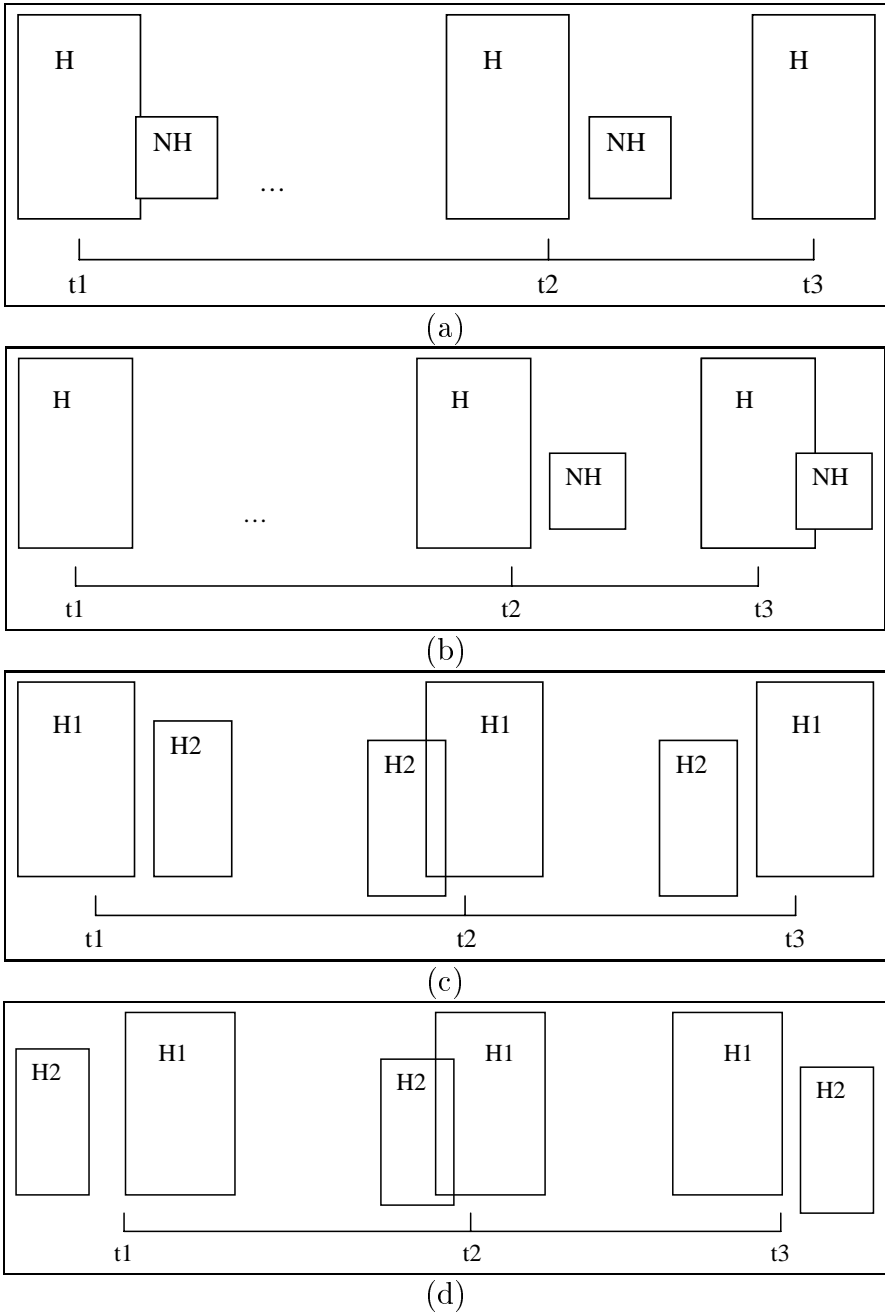


Figure 4.7: Annotation of events. (a) 2 objects are detected at  $t_1$ , and they are classified as one human (H), and one non-human (NH). Then, they are detected as two single objects at  $t_2$ , and when NH stops at  $t_3$ , **deposit** is identified. (b) Similar to (a), but **pick up** is identified. (c) H1 and H2 are classified as human at  $t_1$ , and an object group is detected at  $t_2$ . By tracking the orientations of the objects within  $t_1$ - $t_3$  time interval, **crossover** is identified. (d) Similar to (c), except that **move together** is identified because H1 and H2 move in the same direction.

# Chapter 5

## Scenario-Based Querying

One of the most important tasks in automated video surveillance is query processing. Existing systems generally support textual searches for event queries [41, 46]. Some systems also support object queries to some extent [21]. However, not every abnormal situation can be queried by keywords, predicates, etc. Some situations can be treated as a sequence of events and the whole sequence is of interest. One of the main contributions of our work is the *scenario-based querying* capability to detect such sequences, which is not easy to handle in real time. By ordering the events in a scenario, the temporal information about the events is included in the query specification.

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

We observe that there is a need for enhancing object queries with low-level feature descriptors. When suspicious events occur, directional specifications about objects give valuable information. Our approach provides support for a wide range of event and object queries, including low-level features and directional

predicates, to be posed as a part of the scenario. We also include some specialized query types to provide coherent support for after-the-fact activity analysis. *Inverse querying*, *most popular path*, and *most abnormal region* query types are currently supported. Due to the flexible nature of our data and query models, more complex queries can be formulated.

## 5.1 VSQL: Video Surveillance Query Language

Our Video Surveillance Query Language (VSQL) provides support for integrated querying of video surveillance archives by semantic and low-level features. Semantic subqueries contain 12 single-object event types and 6 multi-object event types, which can be combined to form more complex queries. The timegap value has more meaning when specified between the same kind of event types (e.g., a pair of single-object events or a pair of multi-object events). Descriptive keywords can be supplied for the color and shape features of objects. Instead of a detailed expression of these low-level features, an intuitive way of query specification is chosen in our model. It is more realistic that the human operators inspecting (i.e., querying) the videos would choose these features themselves from a set of pre-specified labels corresponding to the primary colors and a set of primitive shapes. VSQL also provides support for inverse querying, which returns a list of objects or events that occurred within a certain time interval. The grammar specification for VSQL is given in Appendix A.1.

Our query model can be easily tailored to various video-surveillance domains. The extracted semantic predicates are the basic ones that can happen in almost every video-surveillance application. Since a rule-based model is chosen for query processing, different context models for different domains can be applied by extending the predicate specifications.

### 5.1.1 Scenario-based Query

A scenario-based query consists of a scenario part and an object information part. The scenario part is a semantic sequence of single-object and/or multi-object event subqueries ordered temporally. Single-object queries are *enter/leave scene*, *stop/stop-and-go*, and *move* in 8 directions. Multi-object queries include *approach/depart*, *deposit/pick-up*, and *crossover/move-together*. Directional predicates can be used to enrich the multi-object queries. The object information part is a list of object-based information predicates including low-level features and/or class labels for objects. The following examples illustrate the scenario-based queries in VSQL.

**Query 5.1** *A person enters a lobby with a bag, deposits his bag, and leaves the lobby.*

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

The query strings are formed by object and event conditions. Object conditions are expressed by `<objcondition>` and event conditions are expressed by `<event-condition>`, as stated in the grammar for VSQL in Appendix A.1. Based on the fact-base snapshot in Appendix A.3, the result of the scenario-based query is the segment [12, 17].

**Query 5.2** *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)
```

### 5.1.2 Event-based Query

Users may want to query all occurrences of a single object or multi-object event in the archive. Query 5.3 is an example of event-based querying.

**Query 5.3** *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)
```

### 5.1.3 Object-based Query

The existence or appearance of an object can be queried. The low-level features (color, shape) and class values (human, non-human, object-group) of the objects can be used to enrich the query. Query 5.4 is an example of object-based querying.

**Query 5.4** *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 uses the *unification* concept in Prolog, a mechanism which binds variables to atoms. The query processor returns all the objects satisfying some pre-specified conditions, such as color, shape and class type. The result of this type of querying is a list of object labels bound to the variables in the query. This type of querying is more meaningful when the video archive is well-annotated, which means that the objects in videos have been assigned labels (e.g., domain-specific names) *a priori*. Query 5.5 is an example of object-based querying with unification.

**Query 5.5** *List the names of all the persons with a black coat.*

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

#### 5.1.4 Inverse Query

Inverse querying means retrieving the list of events or objects appearing within a certain time interval in a video. This inverse querying is very valuable for offline inspection and is most effective when domain-specific activity analysis is of concern. Queries 5.6 and 5.7 are examples of inverse querying.

**Query 5.6** *Which events occurred between keyframes 5 and 25 in video 1?*

```
select events
from 1
where inverse(5, 25)
```

**Query 5.7** *Which objects appear between keyframes 5 and 10 in videos 3 and 4?*

```
select objects
from 3,4
where inverse(5, 10)
```

### 5.1.5 View-based Query

Since operators look at a fixed scene and inspect moving objects while trying to figure out the abnormal situations, a fixed view-based addressing provides a medium for view-based querying of the scene content, which is provided by the inverted tracking scheme. Examples of supported queries include the most popular path of the moving objects, the number of objects entering the scene from the left/right side, and the cell in which a specific event happened most frequently. Queries 5.8 and 5.9 are examples of the *most popular path* and the *most abnormal region* query types.

**Query 5.8** *What is the most popular path in video 1?*

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

**Query 5.9** *What is the most abnormal region in videos 3 and 4?*

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

### 5.1.6 External Predicate Definition

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

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

Once the enter-left predicate has been specified, it can be used as the existing predicate in scenario-based and event-based querying, as shown in Query 5.10. Note that a timegap value of 60 seconds is used for querying loitering. Note also that *move-any* is another external predicate defined to query the move predicate in any of the eight directions.

**Query 5.10** *List all the loitering intervals caused by the entrances from the left in video 3.*

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

## 5.2 Query Processing

Rule-based modeling is an effective way of querying video databases [13]. Hence, a rule-based model has been designed for querying video surveillance archives. Figure 5.1 shows the flow of execution in our query processing scheme. A VSQL query is sent to our inference engine, Prolog, which processes the meta-data (i.e., fact-base) by using a set of rules (i.e., rule-base). Our rule-base is customizable to specific applications since external predicates can be defined in terms of the existing events.

Prior to this rule-based processing, the submitted query string is parsed using a lexical analyzer. Variables (objects unbounded to a value prior to querying) can be specified as part of the query that is to be bound to meta-data after query processing. *Scenario with bounded atoms/variables* is processed by the inference engine to produce the result set. Logical operators can be applied for the result sets of the query types, which enables conjunction and disjunction on more than one queries.



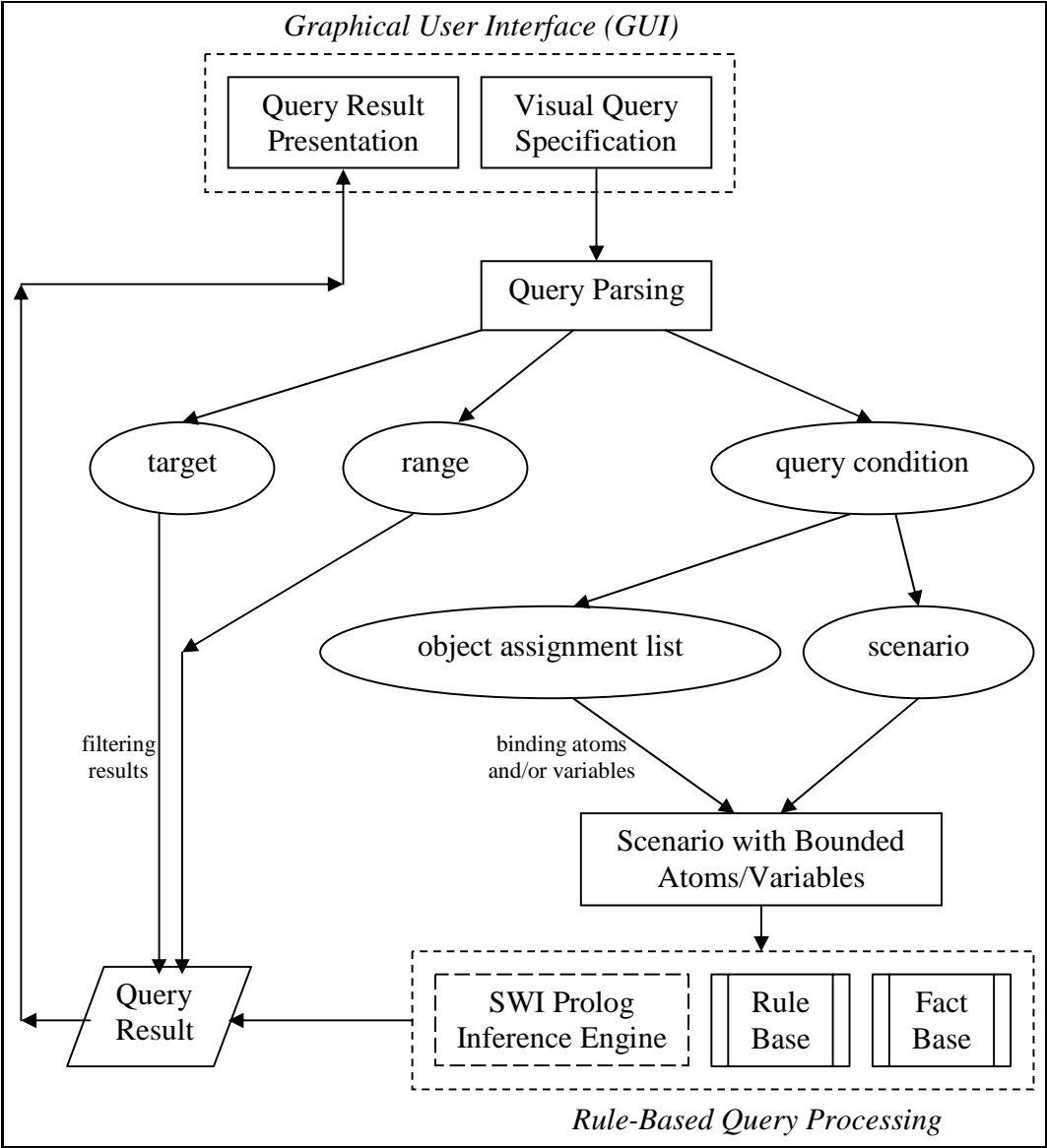


Figure 5.1: The query-processing flowchart. A VSQL query submitted to the GUI passes through the parsing, binding, and processing steps; the results are then presented to the user.

The result of a scenario-based query is a set of intervals, where each subquery is satisfied in an interval element in the result set. Satisfying a scenario-based query means that all subqueries in the scenario have to occur in a specific temporal interval.

**Definition 5.1 (Result of a Scenario-based Query in a Video)** *The result  $R_{S,i}$  of a scenario-based query  $S$  in a video  $v_i$  is a set of intervals specified as follows:*

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

**Definition 5.2 (Result of a Scenario-based Query)** *The result  $R_S$  of a scenario-based query  $S$  for all the videos in the archive is a set of pairs specified as follows:*

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

The inference engine returns a list of keyframe numbers as the result of an event-based query, where the event specified in the query has occurred based on the facts. This result is not a set of intervals because an event occurs at a single keyframe.

**Definition 5.3 (Result of an Event-based Query in a Video)** *The result  $R_{E,i}$  of an event-based query  $E$  in a video  $v_i$  is a set of frame numbers, specified as follows:*

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

**Definition 5.4 (Result of an Event-based Query)** *The result  $R_E$  of an event-based query  $E$  for all the videos in the archive is a set of pairs, specified as follows:*

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

The inference engine returns a list of keyframes as the result of an object-based query, where the object has appeared. If unification is used in the object-based query, then the result is a list of object labels bounded to the unbounded variable in the query string.

A list of event labels or object labels is the result set for inverse querying. Since there might be so many event labels during the time interval specified in the query string, only the important ones that are valuable for after-the-fact analysis are selected for the result set. These are *crossover*, *move-together*, *deposit*, *pickup*, *enter*, *leave*, *stop*, and *stop-and-go*. The result for Query 5.6 is  $\{(1, \{enter, deposit, stop, leave\})\}$  for the time interval between 5<sup>th</sup> and 25<sup>th</sup> keyframes.

### 5.2.1 Logical Operators on Query Results

The intervals in the result sets can be categorized into two types: *non-atomic* and *atomic*. Non-atomicity implies that the condition holds for every frame within the interval. Thus, the condition holds for any sub-interval of a non-atomic interval. Conversely, the condition associated with an atomic interval does not hold for all of its sub-intervals. The intervals in the results of scenario-based queries are atomic, hence they cannot be broken into parts. With this fact in mind, logical conjunction and disjunction operations can be applied to the scenario-based queries. Since the result of an event-based query is a set of frames, the logical operations can be applied directly on the results. Similarly, these logical operations can be applied directly to object-based queries and inverse queries, since the results are simple sets.

#### 5.2.1.1 Conjunction

Assume two scenario-based queries and their results  $R_1$  and  $R_2$ , that contain atomic intervals. Figure 5.2 presents the pseudo-code for obtaining the conjunction  $R_C = R_1 \wedge R_2$  of the query results.

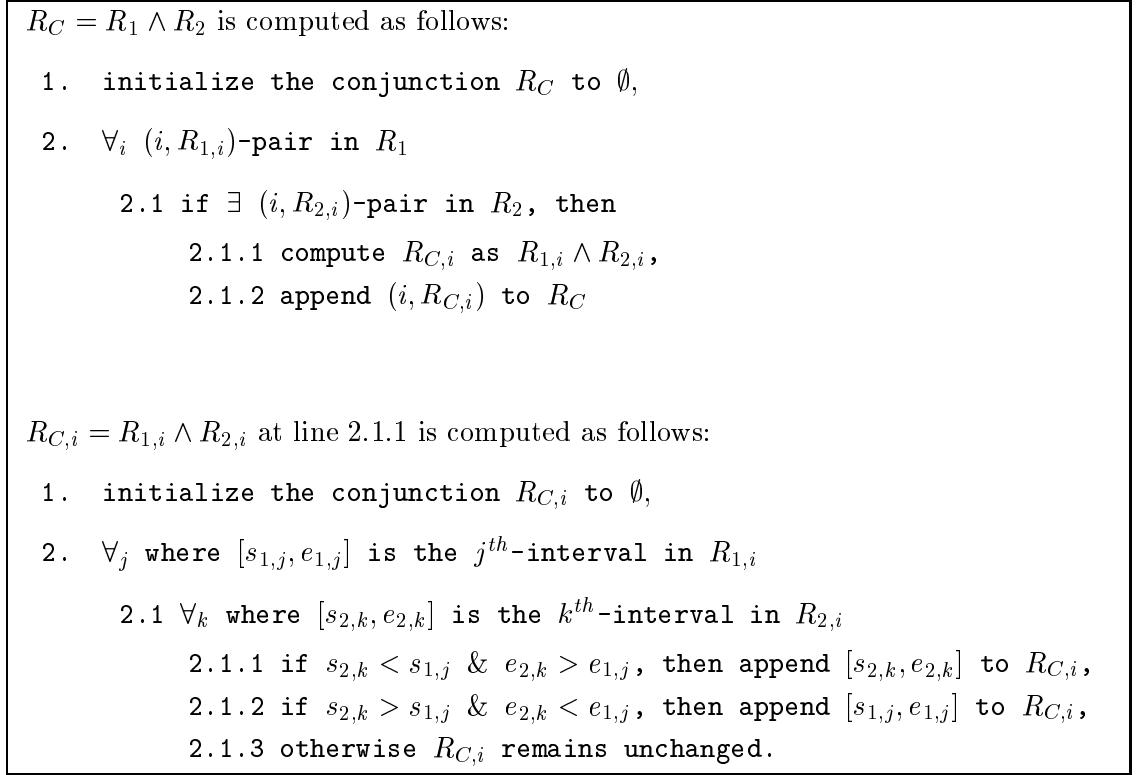


Figure 5.2: The conjunction operation applied on query results.

### 5.2.1.2 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 5.3 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. 5.1 and  $R_2$  as in Eq. 5.2, the disjunction  $R_D$  can be determined as follows:

Assume two scenario-based queries and their results

$$R_1 = \{(1, \{[50, 325], [447, 740]\}), (3, \{[25, 285], [780, 940]\})\}, \quad (5.1)$$

and

$$R_2 = \{(1, \{[200, 475], [520, 700]\}), (2, \{[120, 340]\})\}. \quad (5.2)$$

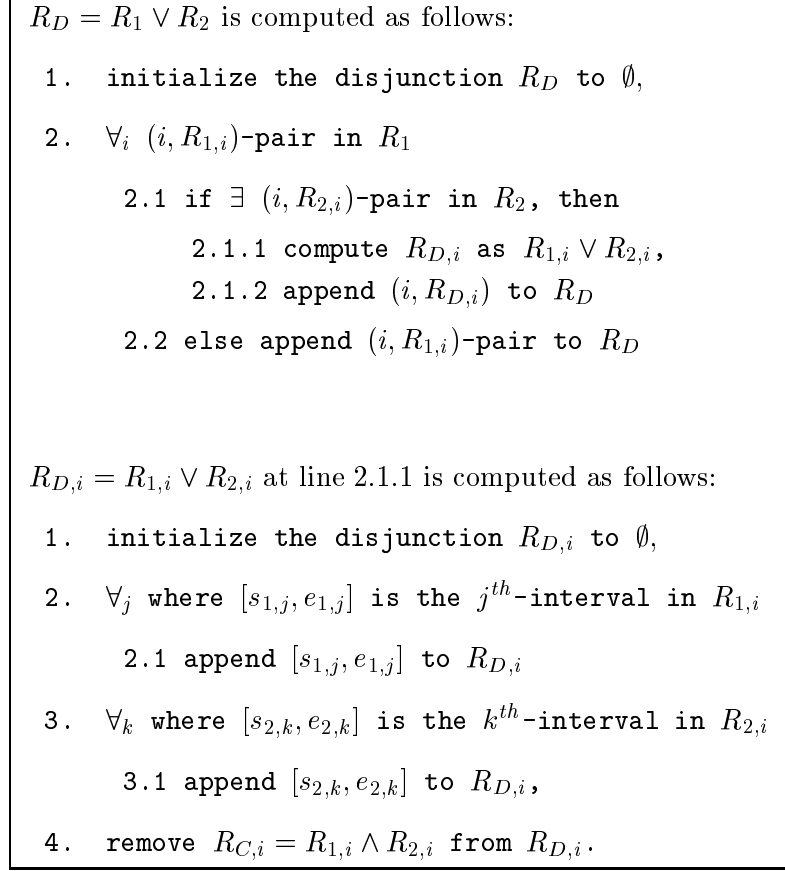


Figure 5.3: The disjunction operation applied on query results.

If we apply the conjunction and disjunction algorithms in Figure 5.2 and Figure 5.3 to obtain  $R_C = R_1 \wedge R_2$  and  $R_D = R_1 \vee R_2$ ,

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

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

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

Hence,

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

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

### 5.2.2 View-based Query Processing

The results of these queries are based on the row-wise indexing of the cells in inverted tracking, where the first cell is in the top-left corner. The most abnormal region is determined by inspecting the high-level events (i.e., deposit, pick-up, crossover, move-together). The cell-id that these events occur most is selected as the most abnormal region. This might be a set of regions since the event occurrences are counted.

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

The result for Query 5.8 is  $\{(1, \{5, 6, 10, 11, 12\})\}$ . The starting cell is found to be 5 and the end cell is determined as 12. All the in-between cells (i.e., 6, 9, 10, 7, 11, 8 in traversal order) are inspected and the most popular path is obtained. The result for Query 5.9 is  $\{(3, \{7\}), (4, \{14\})\}$ .

## 5.3 Visual Query Specification

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

The visual query interface provides *object specification*, *event specification*,

and *scenario specification* facilities. It also utilizes an XML-based repository to make the specified objects available for later usage. The scenarios are expressed as a sequence of events on a drawing panel and the order can be modified to obtain various scenario combinations. Since a scenario is a sequence of events, the events forming the scenario have to be specified first. The specification of Query 5.11 through the interface is illustrated in Figure 5.4.

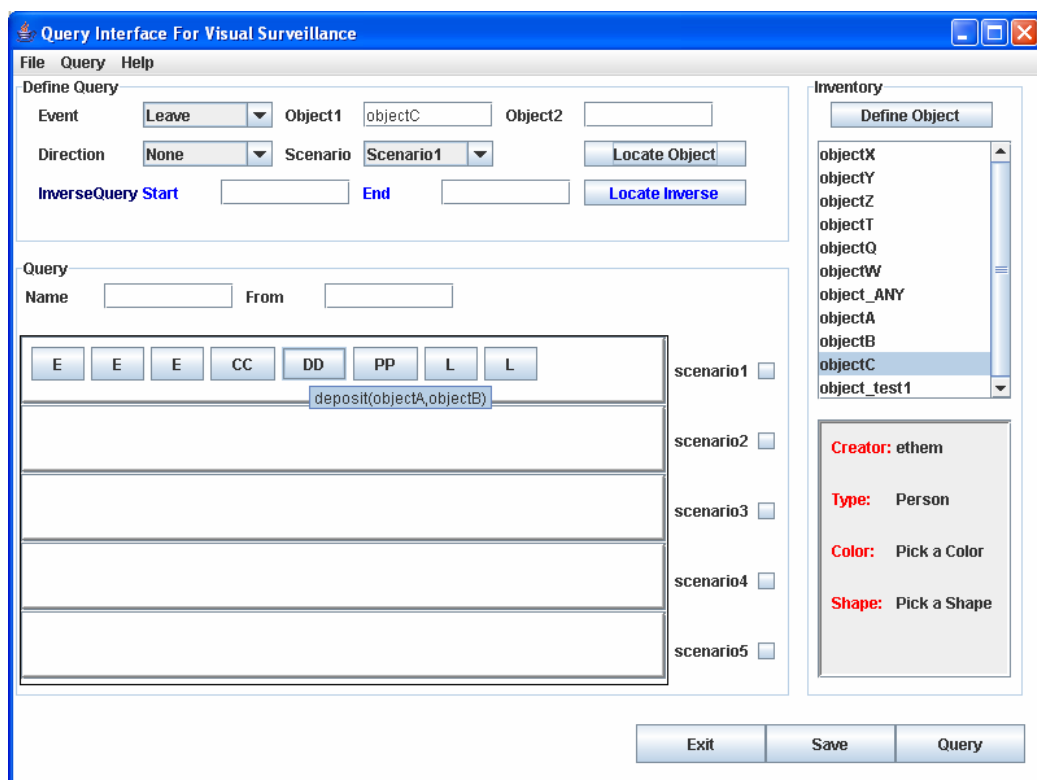


Figure 5.4: The specification of Query 5.11. The event sequence corresponding to the scenario is shown in the first row of the scenario drawing panel. The letters in the boxes on the drawing panel ('E', 'CC', 'DD', 'PP' and 'L') denote enter, crossover, deposit, pickup and leave event predicates, respectively.

**Query 5.11** *List the segments from video one where two persons, one with a bag, meet; then the person carrying a bag leaves the bag; the other person takes the bag; and then both persons leave.*

```
select segment
```

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

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

The object specification in the visual query interface is crucial not only for object-based queries, but also for event-based and scenario-based queries. Since descriptive keywords are used for the low-level features of the human(s) and/or non-human(s), the interface is devised accordingly (see Figure 5.5).



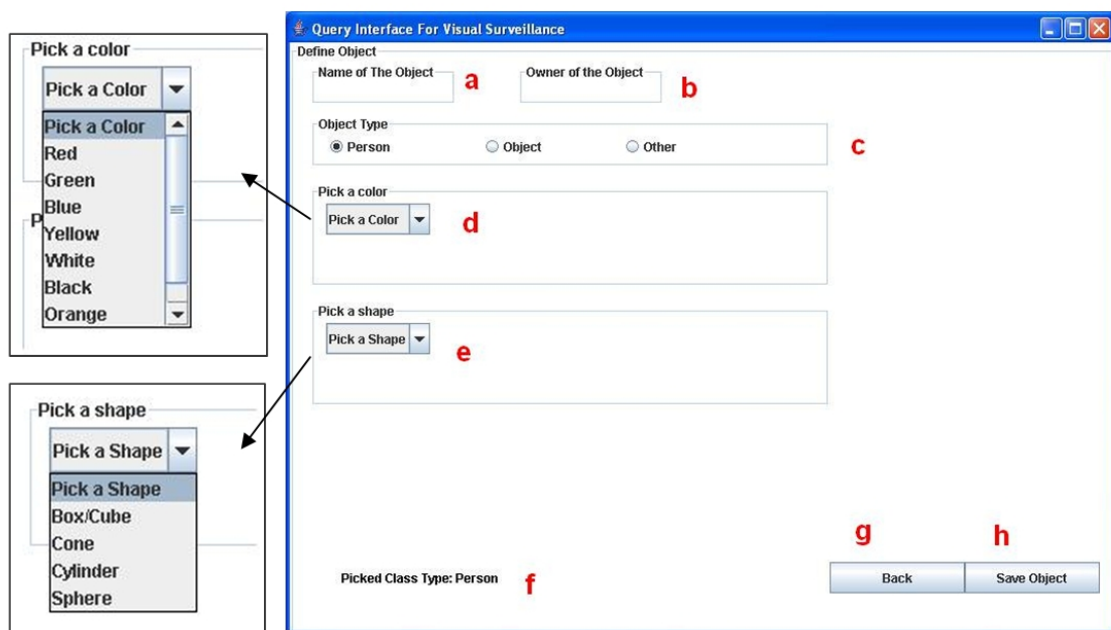


Figure 5.5: Object specification: (a) name; (b) name of the owner/creator; (c) type; (d) color; (e) shape of the object; (f), (g) and (h) are the auxiliary interface elements.

## Chapter 6

# Performance Experiments

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

The evaluation of these algorithms is non-trivial and subjective. In [30], a discussion on the performance evaluation of object-detection algorithms is given. Among the standard measures, Receiver Operating Characteristics (ROC) analysis [18] is used to inspect the effect of a single parameter to the classifier by plotting the true-positive-rate (TPR) and false-positive-rate (FPR) values that are calculated while keeping all the other parameters fixed. Since the algorithms in our keyframe labeling technique yield binary outputs, a set of points is plotted on ROC curves. To elaborate on this, our keyframe labeling algorithm yields exact labels instead of label percentages for keyframes. Hence, a set of points is plotted on ROC curves for the rest of the tests. The points over the  $x = y$  line are considered as good classification results, whereas the ones below are bad.

The best classifier is considered to be at point  $(0, 1)$ , which is the farthest point to the  $x = y$  line.

The benchmark data sets provided by PETS 2004 [1] and PETS 2006 [2] were used as our ground truth for the performance experiments. We manually annotated two sample video from PETS2004, namely *leftbag* and *meetsplit*, having 426 and 543 frames, respectively. Two sample video from PETS2006 are also annotated, namely *S1T1C3* and *S2T3C3*, having 2526 and 2763 frames, respectively. We employed a fivefold cross-validation method for the experimental evaluation.

The major drawback of our keyframe labeling algorithm is the fact that it might not be suitable for crowded scenes, such as video streams of PETS 2007 [3] dataset. The ROC analysis gives poor results for this dataset and the detection accuracy is low. The main reason is that it is very hard to identify the keyframe with a single label in a crowded scene. Too many split/join/move events occur simultaneously. For such datasets, more sophisticated keyframe labeling structures have to be designed. Another drawback is when the object size is very large (e.g., an object occupies nearly one fourth of the frame size or more). In such cases, forming a grid for inverted tracking does not bring significant improvement when compared to the typical detection techniques. However, since the scenario-based query processing component of the framework is decoupled with the moving region extraction parts, the performance of the query processing capabilities is not directly affected with this drawback.

## 6.1 Performance of Keyframe Labeling

### 6.1.1 Pixel-Level Analysis

The preprocessing phase of keyframe labeling includes moving region extraction and tracking, where we adapted widely used background subtraction and maintenance schemes, and the inverted tracking scheme that uses motion-history images. Since ROC analysis requires ground truth evaluation for each parameter setting,

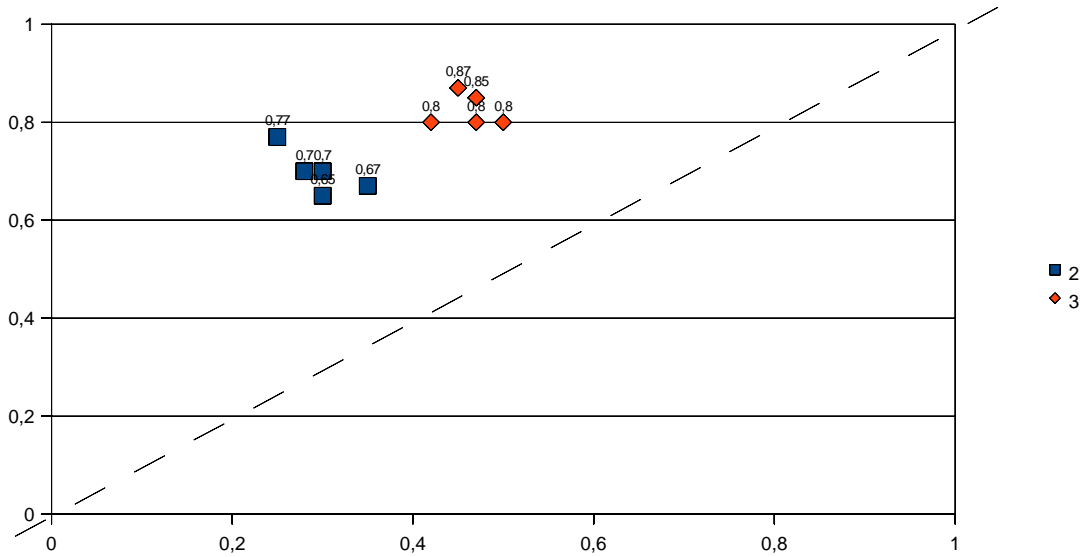


Figure 6.1: ROC curve/Analysis of pixel-level algorithms. The learning constant  $\alpha$  varies from 0.6 to 0.8 in increments of 0.05. The temporal duration constant  $\tau$  is set to 2 and 3.

we focused on two parameters:  $\alpha$  (the learning constant, in Equations 2.3 and 2.5) and  $\tau$  (the temporal duration of the movement, in Equation 2.6). Two sets of experiments were carried out for this analysis. In both of the experiments, the learning constant  $\alpha$  varies from 0.6 to 0.8 in increments of 0.05, yielding five points in the ROC curve. The temporal duration constant  $\tau$  is set to 2 and 3. The extracted regions are annotated manually, and correctly detected pixels are considered as true [30]. As shown in Figure 6.1, the optimal values for the two crucial pixel-level parameters are found to be 2 for  $\tau$ , and 0.7 for  $\alpha$ , since the points corresponding to  $\tau = 2$  are closer to the  $(0, 1)$  point. The point corresponding to  $\tau = 2$ ,  $\alpha = 0.7$  is the farthest to the  $x = y$  line.

### 6.1.2 Grid Size Analysis

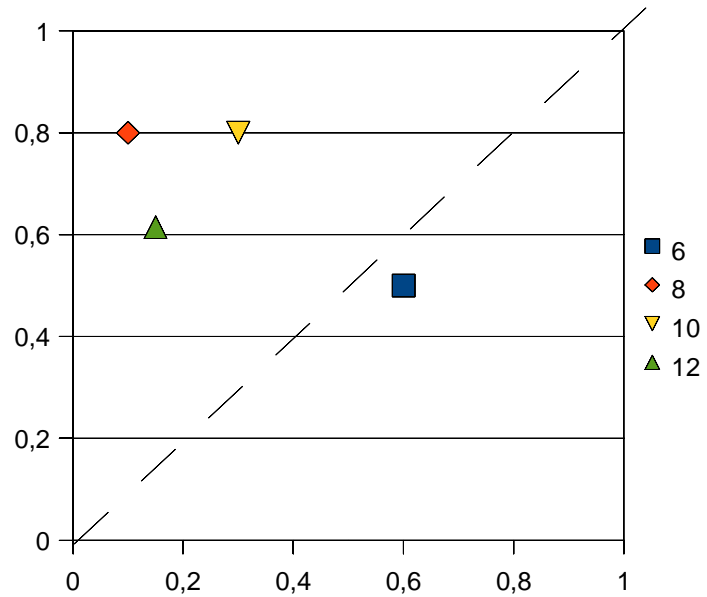
Figure 6.2 shows the ROC analysis results for inspecting the effect of the grid size parameter to anomaly detection. TPR and FPR values are computed using

the outputs of the FSA-based anomaly detection algorithms. The positive output frames for each of the classes are annotated manually, and a similar set of negative output frames is annotated for the analysis. For this experimental setup, the temporal detection threshold  $t_d$  is set to 3 frames for PETS 2006 and 2 frames for PETS 2004 dataset.

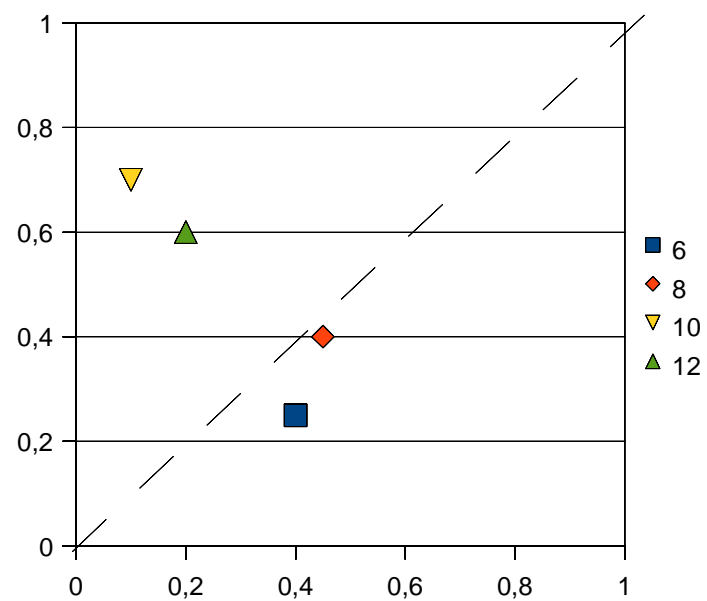
In Figure 6.2 (a), the only bad classification occurs when the grid size is  $6 \times 6$ . The detection algorithm gives best results for the grid size  $8 \times 8$ . In Figure 6.2 (b), the classification for the grid sizes of  $10 \times 10$  and  $12 \times 12$  are among the good ones and  $10 \times 10$  gives better results. The main reason behind this difference among different datasets is the variance in the pixel-level properties, such as object sizes, average velocities, etc. We can conclude that the grid sizes for various datasets should be determined based on the dataset characteristics. The calculation of the appropriate grid size in terms of the pixel-level parameters is very hard. Hence, ROC analysis can be performed, as above, to determine effective grid sizes for the datasets.

### 6.1.3 Temporal Filtering Analysis

Figure 6.3 shows the ROC analysis results for inspecting the effect of the temporal detection threshold parameter  $t_d$  (see Figure 2.2) on anomaly detection. TPR and FPR values are based on the outputs of the FSA-based anomaly detection algorithms. The positive output frames for each class are annotated manually, as well as a similar set of negative output frames for the analysis. For this experimental setup, the grid size is set to  $8 \times 8$  for PETS 2006 and  $10 \times 10$  for PETS 2004 dataset. In Figure 6.3 (a),  $t_d = 3$  gives the best results, whereas in Figure 6.3 (b),  $t_d = 2$  detects the anomalies better than the other values. As expected, increasing the temporal detection threshold lowers the detection accuracy significantly.

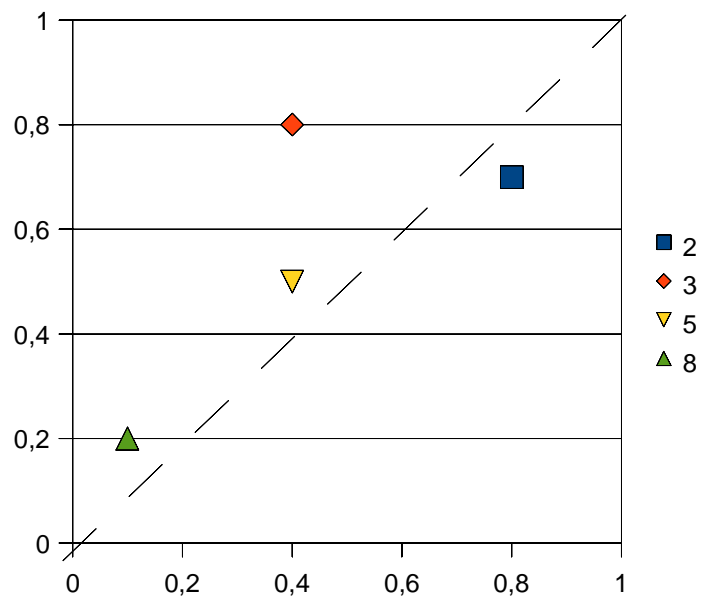


(a)

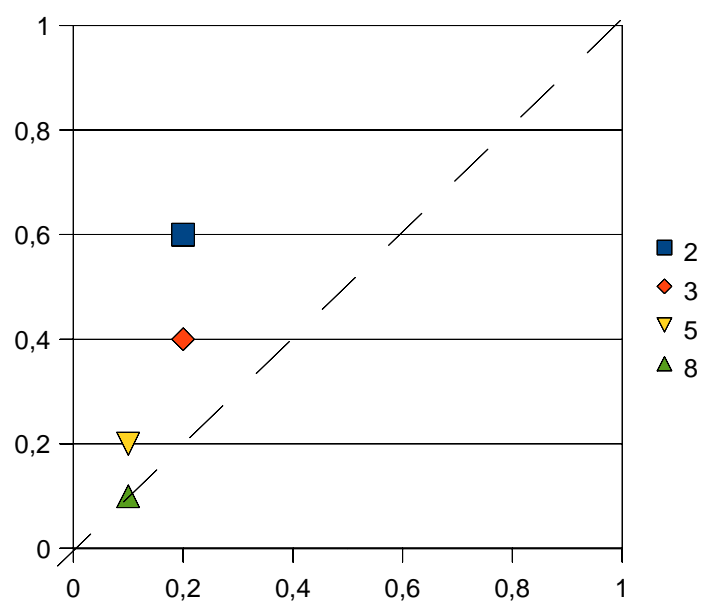


(b)

Figure 6.2: ROC curve analysis for the grid size parameter. (a) PETS 2006 dataset, and (b) PETS 2004 dataset.



(a)



(b)

Figure 6.3: ROC curve analysis for the temporal detection threshold parameter  $t_d$ . (a) PETS 2006 dataset, and (b) PETS 2004 dataset.

### 6.1.4 Storage Gain Analysis

The keyframe-based techniques are used to reduce the storage requirements in video processing. The storage gain can be expressed by the ratio of the number of keyframes extracted and the total number of frames. However, a better storage gain analysis can be performed in terms of the total number of objects, since there might be empty frames in the input stream, which lead to unrealistic results. Hence, the following performance analysis is also performed to inspect the required storage space in terms of the number of objects that are processed. To be fair, the number of objects is computed after completing all of the filtering steps. The main reason behind this comparison is that all objects have to be processed in a typical detector to detect anomalies, whereas the sequence of labels for all keyframes have to be processed in our approach. Figure 6.4 presents the results of this analysis. As expected, the keyframe-based approach has a significantly lower storage cost and input size.

## 6.2 Performance of Histogram-Based Approach

The histogram-based approach (HBA) is used to encode the shape and color content of the objects. The content-based retrieval performance of HBA is shown through a set of experiments in [38, 39]. In addition to its usage in content-based retrieval, object-level meta-data extraction and object classification are performed based on HBA in the scenario-based querying framework. Due to the nature of the video surveillance domain, shape feature is more descriptive than color.

We compare the effectiveness of the shape histogram in HBA is compared with the Generic Fourier Descriptors (GFD), which is derived by applying a modified polar Fourier transform on the shape of an object [51, 52]. In the experiments, 5 radial frequencies and 12 angular frequencies are selected to obtain 60 GFD features. The radial and angular frequencies are similar to our distance and angle histograms, respectively. The following datasets are used:



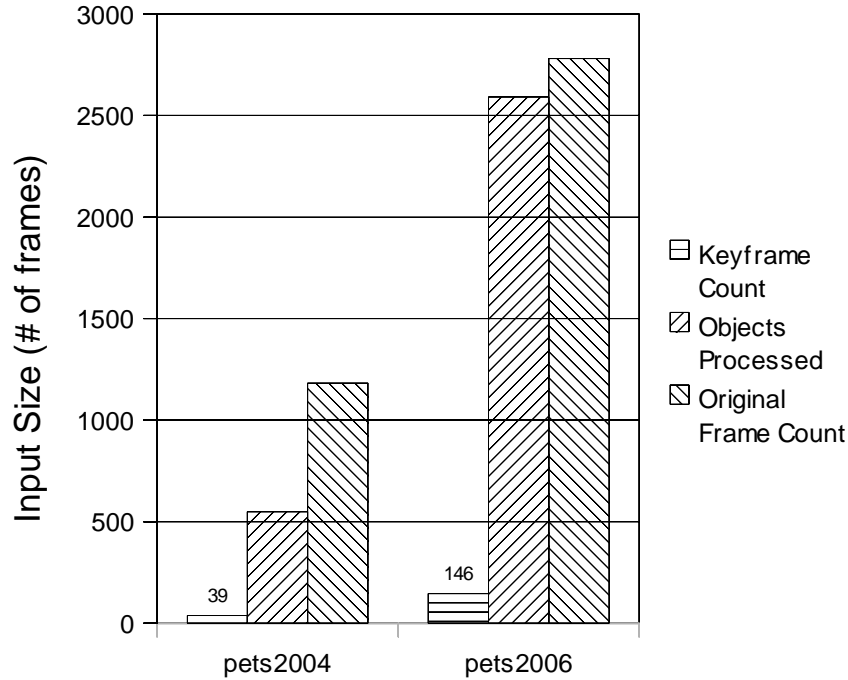


Figure 6.4: The storage gain and the reduce in the input size for detection for PETS 2004 and PETS 2006 datasets. The keyframe count that is extracted by our technique significantly reduces not only the input size for anomaly detection but also the storage space for after-the-fact analysis.

- MPEG-7 region shape database (CE-2) is used to test the robustness of the methods including perspective transformation, scaling, and rotation invariance. The dataset contains 3621 shapes, 651 of which are organized into 31 categories. Each category consists of 21 similar shapes generated by rotating, scaling, and perspectively transforming the shapes. This dataset is also used in [52, 51] to test the retrieval effectiveness of the GFDs.
- In [40], 216 shapes are selected from MPEG-7 set. The shape database can be grouped into 18 categories, each of which has 12 similar shapes.

The shapes are extracted automatically from images, since the images are binary. The retrieval is also performed automatically since the categorization (i.e.,

relevance) is known in advance. The experiment is performed for randomly picked 100 query objects and the results are averaged. Figure 6.5 presents the retrieval effectiveness results of this experiment. The retrieval analysis is performed for the first 20 retrievals and the weights are 0.0, 0.5, and 0.5 for color, distance, and angle histograms, respectively in order to be fair since GFD operates on the shape content only. The results in Figure 6.5 (a) demonstrate that the performance of GFD is generally better than HBA. The main reason behind this is the fact that the dataset in Figure 6.5 (a) has perspective transformed objects. However in Figure 6.5 (b), the performance of HBA is better for most of the recall levels where there are not too many perspective transformed objects.

### 6.3 Performance of Object Classification

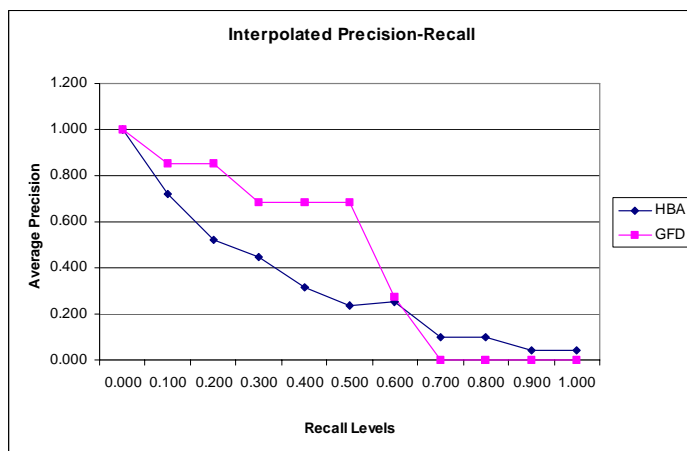
The object-classification algorithm that we used takes region data as the input and yields classification values as percentages. The maximum of these values is chosen as the class value for the input region. Hence, our classification algorithm outputs three classes: human, non-human, and object-group.

In this evaluation, we computed two types of classification accuracies: *Standard Classification Accuracy (SCA)* and *Frame-based Classification Accuracy (FCA)*. The former is the percentage of correctly-classified objects for each class type, and the latter is the frame-weighted percentage of classification accuracy for each object classified correctly. To elaborate on these definitions, Equations 6.1 and 6.2 are given for the human class type as follows:

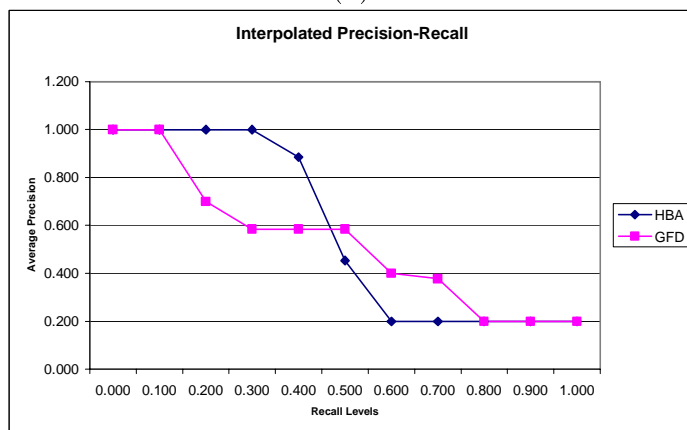
Let  $CC_H$  denote the set of correctly classified humans, and  $|CC_H|$  denote its cardinality. The SCA for the human class type is formulated as:

$$SCA_H = \frac{|CC_H|}{|H|}, \quad (6.1)$$

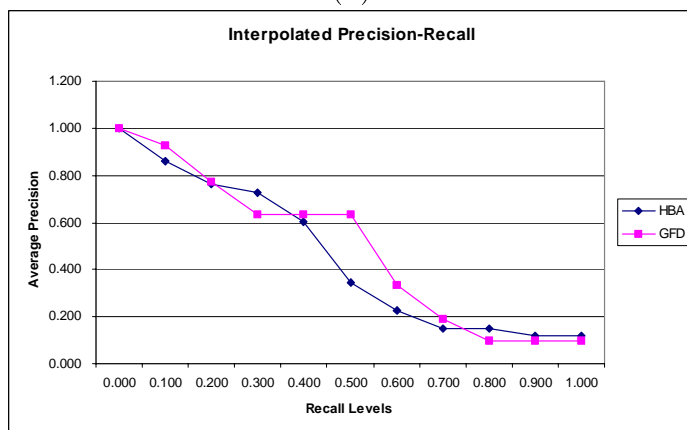
where  $H$  is the set of objects of human class type.



(a)



(b)



(c)

Figure 6.5: Retrieval effectiveness results, (a) with the dataset used in [52, 51], (b) with the dataset used in [40], and (c) with the two datasets mixed.

Let  $F_H$  denote the set of frames where at least 1 human is present, and  $FCC_H$  denote the set of frames where at least 1 human is classified correctly. The FCA for the human class type is formulated as:

$$FCA_H = \frac{|FCC_H|}{|F_H|}. \quad (6.2)$$

PETS 2004 [1] and PETS 2006 [2] benchmark datasets are used in this analysis. We selected frame samples corresponding to 918 *human*, 94 *non-human*, and 133 *object-group* in PETS 2004 dataset; and 1674 *human*, 313 *non-human*, and 492 *object-group* in PETS 2006 dataset. A fivefold cross validation method is employed, where the 20% of the samples are used each time for testing. Figure 6.6 presents the results of the object-classification analysis. Since the number of frames (i.e., keyframes) that an object is classified correctly was taken into consideration, the frame-based accuracy analysis gives better results. The lowest accuracy improvement in the frame-based analysis is for the human class type; many people enter and leave the scene, and in most cases, they are classified correctly. Frame-based classification accuracies for the other two object classes improved significantly when compared to standard accuracies.

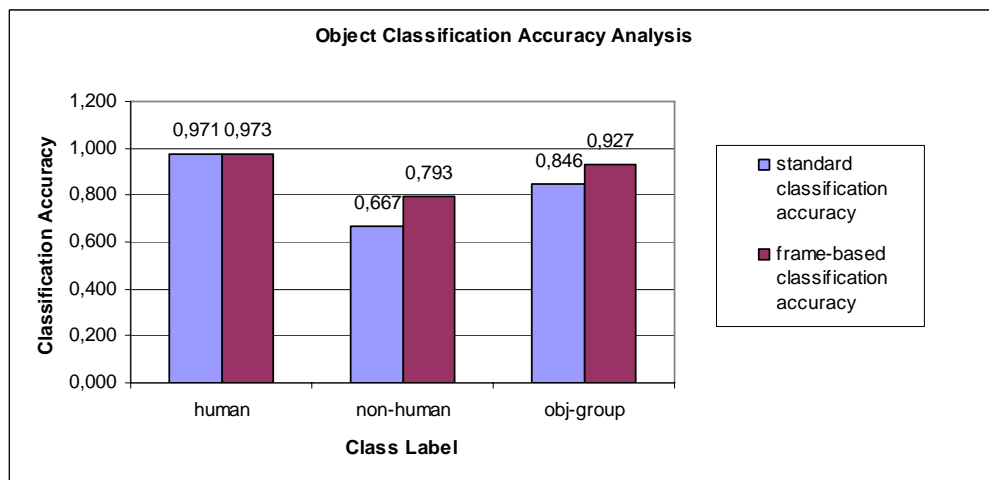


Figure 6.6: Object classification accuracy analysis.

## 6.4 Performance of Semantic Annotation

To evaluate the performance of our semantic annotation process, the detection of each manually annotated event type is inspected. Since both scenario-based and inverse queries are processed by our inference engine, the retrieval accuracy can be evaluated by the performance of the event-detection mechanism. To be more realistic, the percentage of the frames in which the events are correctly identified is used to judge the accuracy of the event annotation instead of just counting the correctly identified event types. This analysis is similar to the frame-weighted classification accuracy above and the results are presented in Figure 6.7.

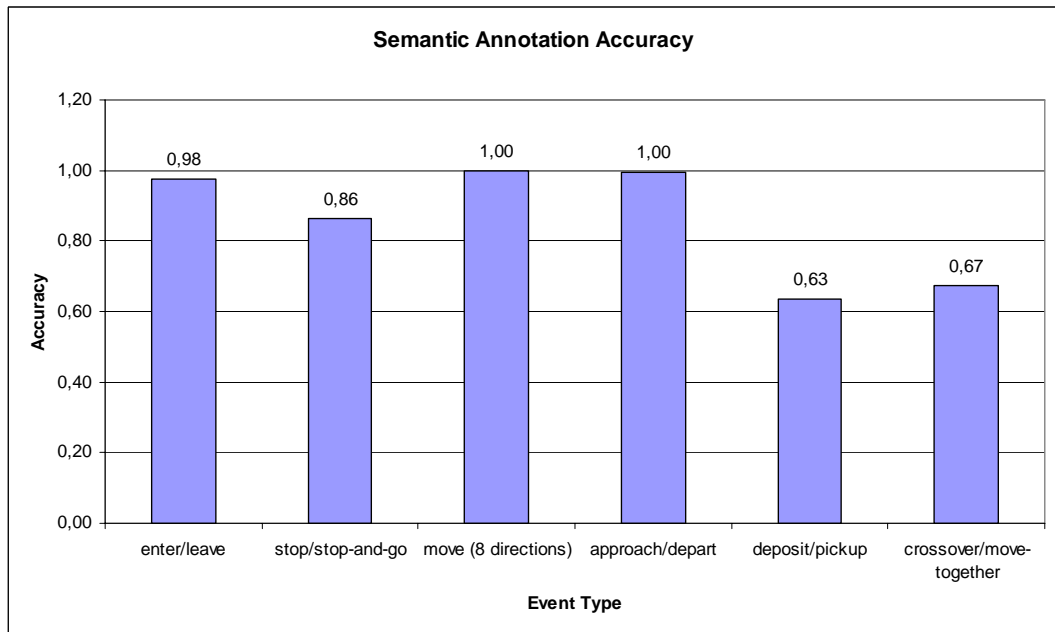


Figure 6.7: Semantic annotation accuracy analysis.

PETS 2004 [1] and PETS 2006 [2] benchmark datasets are used in this analysis. We annotated 600 *enter/leave*, 120 *stop/stop-and-go*, 3600 *move*, 2400 *approach/depart*, 120 *deposit/pick-up*, and 240 *crossover/move-together* frame samples corresponding to the events from the datasets. A fivefold cross validation method is employed, and the results are shown in Figure 6.7, where similar event types are grouped together to simplify the analysis. The accuracy is very high

for *enter/leave*, *move*, and *approach/depart* events because they can be detected directly using the extracted regions. Since we incorporate the number of frames in which an event is identified into the accuracy, multi-object event types have lower accuracies than the other event types. This is simply because regions detected as object-groups cannot be split into single objects in the frames they are detected.

The query processor regards what is extracted as meta-data, since we utilized Prolog as the inference engine in the querying process. However, based on the meta-data extraction accuracies, there could be events that cannot be retrieved since they are not extracted as meta-data (see Figure 6.7). It can be concluded that the performance of scenario-based querying and retrieval depends on the performance of the semantic annotation process.

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

# Chapter 7

## Related Studies

There exist quite a number of approaches in the literature dealing with monitoring and anomalous behavior detection. Many techniques are proposed for dynamic scene segmentation [11, 27, 19, 28, 31, 22, 4, 48, 34, 44] as the pre-processing phase of anomaly detection. One of the primary tasks in monitoring is the abnormal action detection [16, 49, 50, 53, 20, 15, 12, 42] caused by moving objects in the scene. The video surveillance data has both spatial and temporal characteristics and the anomalies are caused by motion or insertion of foreign object(s) to the scene [23]. Each data point has a few continuous attributes, such as color, lightness, texture, and the anomalies to be detected are either anomalous points or regions in the scene [9]. Online anomaly detection techniques, as well as offline processing support [32], are required and one of the key challenges is the large input size.

We provide a brief description of the existing video-surveillance systems in this chapter. We also compare our system to the others in terms of offline query-processing capabilities, especially regarding the semantic content.

## 7.1 Video Surveillance Systems

Stringa and Regazzoni ([46, 45, 33]) propose a real-time surveillance system employing semantic video-shot detection and indexing. In that system, lost objects are detected with 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 comment on more-complex query types, including color and/or shape properties to some extent. We extract object-based low-level features and provide a scenario-based querying scheme for complex querying, also including color and shape descriptors of the objects.

Video Surveillance and Monitoring (VSAM) system proposed by Collins et al. is one of the complete prototypes for object detection, tracking and classification [11]. The hybrid algorithm developed in that work is based on adaptive background subtraction by three-frame differencing. The background maintenance scheme is based on a classification of pixels (either moving or non-moving) performed by a simple threshold test. A model is provided on temporal layers for pixels and pixel regions in order to better detect stop-and-go movements. The background maintenance scheme we employ is similar to that of VSAM's; however, we also use the extracted background for object tracking.

IBM's MILS (MIddleware for Large Scale Surveillance) [21] system provides a complete solution for video surveillance, including data-management services that can be used for building large-scale systems. MILS also provides query services for surveillance data, including *time*, *object size*, *object class*, *object motion*, *context-based object content similarity* queries, and any combination of these. The system employs relevance feedback and data-mining facilities to increase its effectiveness. The main difference between our querying mechanism and that of MILS' is that ours provides opportunities for defining specialized queries in a more expressive manner.

Rivlin et al. [34] propose a real-time system for moving object detection,



tracking, and classification where the video stream originates from a static camera. Effective background initialization and background adaptation techniques are employed for better change detection. The target detection phase also benefits from a color table representing object data. The detected moving objects are classified as *human*, *animal*, and *vehicle* with the help of an expressive set of feature vectors. The authors initiate their feature-vector selection process with a wide set of object-appearance and temporal features. A reduced set, which leads to the best classification accuracy in their experiments, is used for classification. The authors also present a classification approach that combines appearance and motion-based features to increase the accuracy [5].

Haritaoglu et al. [22] propose a model for real-time analysis of people's activities. Their model uses a stationary camera and background subtraction to detect the regions corresponding to a specific 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 by monocular gray-scale video data, and no color cues are used. Creating models of people's appearances helps track interactions (e.g., occlusions) and simultaneous activities. The system uses a statistical background model holding a bimodal distribution of intensity at each pixel to locate people. The system is capable of detecting a single person, multiple persons and multiple-person groups, in various postures.

Lyons et al. [29] developed a system called Video Content Analyzer (VCA), the main components of which are background subtraction, object tracking, event reasoning, graphical user interface, indexing, and retrieving. They adapt a non-parametric background-subtraction approach based on [17]. VCA differentiates between people and objects and the main events it recognizes are *entering scene*, *leaving scene*, *splitting*, *merging*, and *depositing/picking-up*. Users are able to retrieve video sequences based on event queries whose categories are similar to those we use. However, we also provide object-based querying that can be refined by low-level and/or directional descriptors.

Brodsky et al. [7] describe a system for indoor visual surveillance, specifically for use in retail stores and homes. They assume a stationary camera and

use background subtraction. A list of events that the object participates in is stored for each object, simply, *entering*, *leaving*, *merging*, and *splitting*. Both of these techniques operate at the pixel-level and region-level, whereas we provide techniques to transform the input stream into an event sequence representation, which is easier to process and has lower storage costs.

Kim and Hwang present an object-based video abstraction model, where a moving-edge detection scheme is used for video frames [25, 26]. A semantic shot-detection scheme is employed to select object-based key-frames. When a change occurs in the number of moving regions, the current frame is declared as a 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 in the next frame, a shape-based change detector is applied to the remaining frames. The use of keyframes in this approach is very similar to our keyframe detection scheme; however, we utilize the keyframe detection scheme with inverted tracking data model and extend it further by assigning descriptive labels to the keyframes.

Shet et al. [41] present a visual surveillance system, VidMAP, that combines real-time video-processing algorithms with logic programming to represent and recognize activities. The authors used Prolog for the high-level rules that correspond to their supported query set. *Entry violation*, *theft*, and *possess* are examples of rules they use to answer specific queries. Our query-expression scheme is based on VSQL, which is more intuitive and flexible than Prolog. We provide low-level feature descriptions and directional predicates as well as temporal information about events to enrich the query set. Our query-processing system also supports inverse querying and view-based querying, and hence provides a wider range of query types.

In [10], a view-based multiple objects tracking system is proposed including human action recognition scheme. The basic aim in that work is to recognize human actions in an interactive virtual environment even the actions are not abnormal. The blob tracking phase that they have developed assigns labels to each blob based on their previous motion and current motion. The labels that they

use are *continue*, *merge*, *split*, *appear*, and *disappear*, and they hold an inference graph to track multiple objects simultaneously. The labeling mechanism of their scheme and our approach is similar; however, we assign labels to the frames as a global representation of the event occurred at the frame. Moreover, we apply a keyframe-based technique to narrow the storage and processing requirements.

## 7.2 Comparison of Our System to Related Work

A comparison of our work to the related studies is presented to elaborate on our querying capabilities. The query effectiveness relies on the effectiveness of the meta-data extraction process. The background maintenance scheme we employ is similar to that of VSAM's [11] with an improvement that the extracted background is also used for object tracking. We employ a similar strategy of moving-object counting mentioned in [25] extended with a keyframe detection scheme, to provide an effective storage for predicates.

The event predicates extracted in our data model are generic in the video-surveillance domain and generally supported by existing systems (e.g., [29, 45, 7]). Object-based querying is also implemented in most of the existing systems (e.g., [25, 21]) to some extent. The query types based on color is argued in some systems (e.g., [46, 45, 33]), however, the existing query models are not rich enough to support low-level feature information about the objects. We extract object-based low-level features and provide a scenario-based querying scheme for complex querying, also including color and shape descriptors of the objects (e.g., Queries 5.4 and 5.5).

Our query model provides scenario-based querying, which, based on low-level feature descriptors and directional predicates, allows temporal ordering of the event predicates as well as of object information. An inverse-querying scheme is also provided to help the offline inspection process. In addition, view-based querying is available, which augments the query expressiveness of our model. Hence, one of the main difference between our querying mechanism and that of

MILS' [21] is that ours provides opportunities for defining specialized queries in a more expressive manner (e.g., Queries 5.6-5.10).

Our query-expression scheme is based on VSQL, which is a more intuitive way of expressing logical predicates than Prolog. In VidMAP [41], the authors provide high-level rules for a couple of query types. On the other hand, we provide low-level feature descriptions and directional predicates as well as temporal information about events to enrich our query set. Our query-processing system also supports inverse querying and view-based querying, and hence provides a wider range of query types. Compared to the existing systems, the scenario-based query-processing technique we propose provides support for a wide range of query types and semantically enriches after-the-fact analysis.

# Chapter 8

## Conclusion

We propose a keyframe labeling technique, which simply assigns labels to the keyframes extracted based on a keyframe detection algorithm. The keyframe detection technique we use relies on an inverted tracking scheme, which is a view-based representation of the video by an inverted index. A keyframe is detected if a change occurs in the Motion Appearance Mask of the frame when compared to that of the previous frame. The keyframes are categorized into four simple types based on the appearances of the identified moving regions. As a result of the keyframe labeling process, the input stream is represented as a temporally-ordered sequence of keyframes. Since an anomaly can be considered as a sub-sequence of the keyframe labels, the anomaly detection is carried out on this sequence; hence, the complexity of the anomaly detection task is reduced. The keyframe labeling technique reduces the large input size for on-the-fly processing, and thus, reduces the storage requirements for after-the-fact analysis.

We also provide finite state automata-based mechanisms to detect a typical set of anomalous situations. We devise three separate finite state automata to recognize sequences corresponding to a typical set of anomalies, the inputs of which are the sequence of keyframe labels that we assign to the extracted keyframes. The performance experiments based on the benchmark datasets of PETS 2004 and PETS 2006 show that the finite state automata-based approach provides effective on-the-fly anomaly detection scheme together with the keyframe labeling

technique, since a reasonable detection performance is achieved. In its current form, the technique is not very suitable for very crowded scenes, which requires more sophisticated keyframe labeling structures and more powerful label assignment schemes. Further research will focus on extending the capabilities of the keyframe labeling algorithm to overcome this drawback.

One of the major contributions of this thesis is scenario-based querying, which provides a mechanism for effective offline inspection. A scenario-based query is specified as a sequence of event-based subqueries that can be enriched with object-based low-level features and directional predicates. With the help of the inverted tracking technique we propose, the framework provides support for view-based query handling by using a fixed view-based representation of the content. Support for inverse querying, which is a specialized tool for after-the-fact activity analysis, is also provided. We developed Video Surveillance Query Language, which is specialized for scenario-based querying purposes, to express the supported query types. We devised a rule-based query-processing module to provide not only an efficient processing mechanism for scenario-based querying, but also a flexible medium for external predicate definition, which allows the query model to be tailored to various domains. We also developed a visual query-specification interface to enhance the process.

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

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

# Bibliography

- [1] PETS 2004. Seventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2004) Benchmark Data [http://www-prima.inrialpes.fr/PETS04/caviar\\_data.html](http://www-prima.inrialpes.fr/PETS04/caviar_data.html), May 2004.
- [2] PETS 2006. Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006) Benchmark Data <http://www.cvg.rdg.ac.uk/PETS2006/data.html>, June 2006.
- [3] PETS 2007. Tenth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2007) <http://www.pets2007.net>, October 2007.
- [4] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.
- [5] Y. Bogomolov, G. Dror, S. Lapchev, E. Rivlin, and M. Rudzsky. Classification of moving targets based on motion and appearance. In *Proceedings of the British Machine Vision Conference*, volume 2, pages 429–438, 2003.
- [6] N. Boujemaa and C. Vertan. Integrated color texture signature for image retrieval. In *Proceedings of International Conference on Image and Signal Processing*, pages 404–411, Agadir, Morocco, 2001.
- [7] 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 *Proceedings of the Video-Based Surveillance Systems: Computer Vision and Distributed Processing*, pages 51–65. Kluwer Academic Pub., 2001.

- [8] P. Buser and M. Imbert. *Vision*. MIT Press, Cambridge, Massachusetts, 1992.
- [9] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):15:1–15:58, July 2009.
- [10] J. Choi, Y. Cho, K. Cho, S. Bae, and H.S. Yang. A view-based multiple objects tracking and human recognition for interactive virtual environments. *The International Journal of Virtual Reality*, 7(3):71–76, 2008.
- [11] R.T. Collins, A.J. 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. Technical Report CMU-RI-TR-00-12, Carnegie Mellon University, The Robotics Institute, 2000.
- [12] C. Diehl and J. Hampshire. Real-time object classification and novelty detection for collaborative video surveillance. In *Proceedings of IEEE International Joint Conference on Neural Networks*, pages 2620–2625, Honolulu, HI, 2002.
- [13] M.E. Dönderler, Ö.Ulusoy, and U. Güdükbay. Rule-based spatio-temporal query processing for video databases. *The VLDB Journal*, 13(3):86–103, 2004.
- [14] M.E. Dönderler, E. Şaykol, U. Arslan, Ö. Ulusoy, and U. Güdükbay. Bil-Video: Design and implementation of a video database management system. *Multimedia Tools and Applications*, 27(1):79–104, 2005.
- [15] T. Duong, H. Bui, D. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 838–845, 2005.
- [16] D. Duque, H. Santos, and P. Cortez. The OBSERVER: An intelligent and automated video surveillance system. In A. Campilho and M. Kamel, editors, *Lecture Notes in Computer Science (LNCS) Volume 4141, Proceedings of International Conference on Image Analysis and Recognition (ICIAR)*, pages 898–909. Springer-Verlag, 2006.



- [17] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition, Workshop on Motion*, Ft. Collins, CO, 1999.
- [18] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.
- [19] D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons, and A.K. Jain. A background model initialization algorithm for video surveillance. In *Proceedings of the International Conference on Computer Vision*, pages 733–740, Vancouver, Canada, 2001.
- [20] R. Hamid, A. Johnson, S. Batta, A. Bobick, C. Isbell, and G. Coleman. Detection and explanation of anomalous activities: representing activities as bags of event n-grams. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1031–1038, 2005.
- [21] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl, S. Pankanti, A. Senior, C.-F. Shu, and Y.L. Tian. Smart video surveillance: Exploring the concept of multiscale spatiotemporal tracking. *IEEE Signal Processing Magazine*, 22(2):38–51, March 2005.
- [22] İ. Haritaoglu, D. Harwood, and L. Davis. W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830, 2000.
- [23] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 34(3):334–352, 2004.
- [24] A.K. Jain and A. Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8):1233–1244, 1996.
- [25] C. Kim and J.N. Hwang. Fast and automatic video object segmentation and tracking for content-based applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(2):122–129, 2002.

- [26] C. Kim and J.N. Hwang. Object-based video abstraction for video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(12):1128–1138, 2002.
- [27] K. Kim, T. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground/background segmentation using codebook model. *Real-time Imaging*, 11(3):172–185, 2005.
- [28] L. Li, W. Huang, I.Y.H. Gu, and Q. Tian. Foreground object detection from videos containing complex background. In *Proceedings of the ACM international conference on Multimedia*, pages 2–10, Berkeley, CA, USA, 2003. ACM Press.
- [29] D.M. Lyons, T. Brodsky, E. Cohen-Solal, and A. Elgammal. Video content analysis for surveillance applications. In *Proceedings of the Philips Digital Video Technologies Workshop*, 2000.
- [30] J.C. Nascimento and J.S. Marques. Performance evaluation of object detection algorithms for video surveillance. *IEEE Transactions on Multimedia*, 8(4):761–774, 2006.
- [31] G. Paschos and F.P.Valavanis. A color texture based visual monitoring system for automated surveillance. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 29(2):298–307, 1999.
- [32] C.S. Regazzoni, V. Ramesh, and G.L. Foresti. Scanning the issue/technology: Special issue on video communications, processing, and understanding third generation surveillance systems. *Proceedings of the IEEE*, 89(10):1355–1367, 2001.
- [33] C.S. Regazzoni, C. Sacchi, and E. Stringa. Remote detection of abandoned objects in unattended railway stations by using a DS/CDMA video surveillance system. In C.S. Regazzoni, G. Fabri, and G. Vernezza, editors, *Proceedings of the Advanced Video-Based Surveillance System*, pages 165–178. Boston, MA: Kluwer, 1998.

- [34] E. Rivlin, M. Rudzsky, U. Bogomolov R. Goldenberg, and S. Lepchev. A real-time system for classification of moving objects. In *Proceedings of the International Conference on Pattern Recognition*, volume 3, pages 688–691, 2002.
- [35] E. Şaykol, U. Güdükbay, and Ö. Ulusoy. A keyframe labeling technique for anomaly detection in video surveillance. submitted journal paper.
- [36] E. Şaykol, U. Güdükbay, and Ö. Ulusoy. Scenario-based query processing for video-surveillance archives. *Engineering Applications of Artificial Intelligence*. accepted in August 2009.
- [37] E. Şaykol, U. Güdükbay, and Ö. Ulusoy. A database model for querying visual surveillance by integrating semantic and low-level features. In K.S. Candan and A. Celentano, editors, *Lecture Notes in Computer Science (LNCS) Volume 4457, Proceedings of 11th International Workshop on Multimedia Information Systems (MIS'05)*, pages 163–176, Sorrento, Italy, 2005.
- [38] E. Şaykol, U. Güdükbay, and Ö. Ulusoy. A histogram-based approach for object-based query-by-shape-and-color in image and video databases. *Image and Vision Computing*, 23(13):1170–1180, 2005.
- [39] E. Şaykol, A.K. Sinop, U. Güdükbay, Ö. Ulusoy, and E. Çetin. Content-based retrieval of historical Ottoman documents stored as textual images. *IEEE Transactions on Image Processing*, 13(3):314–325, 2004.
- [40] T.B. Sebastian, P.N. Klein, and B.B. Kimia. Recognition of shapes by editing shock graphs. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 755–762, Vancouver, Canada, July 2001.
- [41] V. Shet, D. Harwood, and L. Davis. Vidmap: Video monitoring of activity with prolog. In *Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance*, pages 224–229, 2005.
- [42] S. Singh and M. Markou. An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):396–407, 2004.

- [43] J.R. Smith and S.F. Chang. Tools and techniques for color image retrieval. In I.K. Sethi and R.C. Jain, editors, *Proceedings of Storage and Retrieval for Image and Video Databases IV, IS&T/SPIE*, volume 2670, pages 426–437, 1996.
- [44] L. Di Stefano, S. Mattoccia, and M. Mola. A change-detection algorithm based on structure and colour. In *Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS'03)*, pages 252–259, 2003.
- [45] E. Stringa and C.S. Regazzoni. Content-based retrieval and real time detection from video sequences acquired by surveillance systems. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 138–142, 1998.
- [46] E. Stringa and C.S. Regazzoni. Real-time video-shot detection for scene surveillance applications. *IEEE Transactions on Image Processing*, 9(1):69–79, 2000.
- [47] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
- [48] D. Thirde, M. Borg, V. Valentin, L. Barthelemy, J. Aguilera, G. Fernandez, J. Ferryman, F. Bremond, M. Thonnat, and M. Kampel. People and vehicle tracking for visual surveillance. In *Proceedings of the Sixth IEEE International Workshop on Visual Surveillance*, pages 169–176, 2006.
- [49] T. Xiang and S. Gong. Beyond tracking: modelling activity and understanding behaviour. *International Journal of Computer Vision*, 67(1):21–51, 2006.
- [50] T. Xiang and S. Gong. Incremental and adaptive abnormal behaviour detection. *Computer Vision and Image Understanding*, 111:59–73, 2008.
- [51] D.S. Zhang and G. Lu. Enhanced generic fourier descriptors for object-based image retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3668–3671, Orlando, Florida, USA, May 2002.

- [52] D.S. Zhang and G. Lu. Shape based image retrieval using generic fourier descriptors. *Signal Processing: Image Communication*, 17(10):825–848, 2002.
- [53] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 819–826, 2004.

# Appendix A

## Grammar for Scenario-Based Querying

The grammar specification for the Video Surveillance Querying Language (VSQL) and the rules for query processing are presented. A snapshot of a sample fact-base is given to illustrate how the meta-data is obtained and used in the query processing framework.

### A.1 VSQL Grammar Specification

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

<target> := segments |      /* retrieve video sequences/intervals */
           frames |        /* retrieve frames for event queries */
           events |        /* inverse querying w.r.t. events */
           objects |       /* inverse querying w.r.t. objects */
           most-popular-path |
           most-abnormal-region | /* view-based queries */
           <objectlist>      /* retrieve object(s) */
```

```

<objectlist> := [<objectlist> ','] <objlabel>

<range> := all | <videolist>

<videolist> := [<videolist> ','] <vid>

<query-condition> := [<object-assignment-list> and] <scenario> |
                    <inverse-condition>

<object-assignment-list> := [<object-assignment-list> ',']
                            <object-assignment>

<objectassignment> := <objlabel> <objoperator> <objcondition>

<scenario> := [<scenario> [<timegap>] ] <event-condition>

<event-condition> := <single-object-event-condition> |
                    <multi-object-event-condition>

<inverse-condition> := inverse '(' <intvalue> ', ' <intvalue> ')

/* event query conditions */
<single-object-event-condition> := <single-object-event-label> '('
                                <objlabel> ')

<single-object-event-label> := enter | leave | stop |
                              stop-and-go | move-<direction>

<multi-object-event-condition> := <multi-object-event-label> '('
                                <multi-object-condition> ')

<multi-object-event-label> := crossover | move-together |
                              deposit | pickup |
                              approach | depart

```

<multi-object-condition> := <objlabel> ‘,’ <objlabel> [‘,’ <direction>]

<direction> := west | east | north | south | northeast |  
southeast | northwest | southwest

/\* object conditions \*/

<objcondition> := objdata ‘(’ <objdesclist> ‘)’

<objdesclist> := [<objdesclist> ‘,’] <objdesc>

<objdesc> := <classdesc> | <colordesc> | <shapedesc>

<classdesc> := class ‘=’ <classvalue>

<colordesc> := color ‘=’ <colorlabel>

<shapedesc> := shape ‘=’ <shapelabel>

<colorlabel> := red | green | blue | yellow | white |  
black | orange | violet

<shapelabel> := box | cone | cylinder | sphere

<classvalue> := human | non-human | object-group

/\* primitive types \*/

<intvalue> := (1-9)(0-9)\*

<vid> := <intvalue>

<timegap> := <intvalue>

<objlabel> := (a-z)(A-Za-z0-9)\*

<objoperator> := ‘=’ | ‘!=’



## A.2 Rules for Query Processing

This section presents the Prolog rule set for processing VSQL queries.

```

/* single object predicates */

p_enter(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
    (enter(K,X,F,_),
    object_info(K,X,C1,R1,S1,F,_),
    assert( result(SQID,K,X,none,F))), L).

p_leave(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
    (leave(K,X,F,_),
    object_info(K,X,C1,R1,S1,F,_),
    assert( result(SQID,K,X,none,F) ) ),
    L).

p_stop(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
    (stop(K,X,F,_),
    object_info(K,X,C1,R1,S1,F,_),
    assert( result(SQID,K,X,none,F) ) ),
    L).

p_stop_and_go(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
    (stop_and_go(K,X,F,_),
    object_info(K,X,C1,R1,S1,F,_),
    assert( result(SQID,K,X,none,F) ) ),
    L).

p_move_west(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
    (move_west(K,X,F,_),
    object_info(K,X,C1,R1,S1,F,_),
    assert( result(SQID,K,X,none,F) ) ),
    L).

p_move_east(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
    (move_east(K,X,F,_),
    object_info(K,X,C1,R1,S1,F,_),
    assert( result(SQID,K,X,none,F) ) ),
    L).

```

```

p_move_north(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
      (move_north(K,X,F,_),
       object_info(K,X,C1,R1,S1,F,_),
       assert( result(SQID,K,X,none,F) ) ),
      L).

p_move_south(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
      (move_south(K,X,F,_),
       object_info(K,X,C1,R1,S1,F,_),
       assert( result(SQID,K,X,none,F) ) ),
      L).

p_move_neast(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
      (move_neast(K,X,F,_),
       object_info(K,X,C1,R1,S1,F,_),
       assert( result(SQID,K,X,none,F) ) ),
      L).

p_move_seast(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
      (move_seast(K,X,F,_),
       object_info(K,X,C1,R1,S1,F,_),
       assert( result(SQID,K,X,none,F) ) ),
      L).

p_move_nwest(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
      (move_nwest(K,X,F,_),
       object_info(K,X,C1,R1,S1,F,_),
       assert( result(SQID,K,X,none,F) ) ),
      L).

p_move_swest(K,SQID,X,C1,R1,S1,L) :- setof([K,X,F],
      (move_swest(K,X,F,_),
       object_info(K,X,C1,R1,S1,F,_),
       assert( result(SQID,K,X,none,F) ) ),
      L).

/* multi-object predicates */

p_crossover(K,SQID,X,C1,R1,S1,Y,_,_,_,D,L) :- setof([K,X,Y,F],
      (crossover(K,X,Y,D,F,_),

```

```

        object_info(K,X,C1,R1,S1,F,_),
        assert( result(SQID,K,X,Y,F) ) ),
        L).

p_move_together(K,SQID,X,C1,R1,S1,Y,_,_,_,D,L) :- setof([K,X,Y,F],
        (move_together(K,X,Y,D,F,_),
        object_info(K,X,C1,R1,S1,F,_),
        assert( result(SQID,K,X,Y,F) ) ),
        L).

p_deposit(K,SQID,X,C1,R1,S1,Y,_,_,_,D,L) :- setof([K,X,Y,F],
        (deposit(K,X,Y,D,F,_),
        object_info(K,X,C1,R1,S1,F,_),
        assert( result(SQID,K,X,Y,F) ) ),
        L).

p_pickup(K,SQID,X,C1,R1,S1,Y,_,_,_,D,L) :- setof([K,X,Y,F],
        (pickup(K,X,Y,D,F,_),
        object_info(K,X,C1,R1,S1,F,_),
        assert( result(SQID,K,X,Y,F) ) ),
        L).

/* object info retrieve */

get_obj_info(V,O,C,R,S,L) :- setof([F],
        object_info(V,O,C,R,S,F,_),
        L).

/* event info retrieve */

get_evt_info_single(V,E,C1,R1,S1,L) :- setof([F],
        (event_info(V,E,F,_),
        object_info(V,_,C1,R1,S1,F,_)),
        L).

get_evt_info_multi(V,E,C1,R1,S1,C2,R2,S2,L) :- setof([F],
        (event_info(V,E,F,_),
        object_info(V,_,C1,R1,S1,F,_),

```

```

        object_info(V,_,C2,R2,S2,F,_),
    L) .

/* inverse query */

get_inverse_evt(V,FL,FU,L) :- setof([E],
    (event_info(V,E,F,_), F=<FU, FL=<F),
    L) .

get_inverse_obj(V,FL,FU,L) :- setof([O],
    (object_info(V,O,_,_,_,F,_), F=<FU, FL=<F),
    L) .

/* view-based query */

/* E can be used to query the most abnormal region of an event */
get_ab_grids(V,E,L) :- setof([G],
    event_info(V,E,_,G),
    L) .

/* scenario-based result */

get_scenario(1,V,01,02,L) :- setof([V,01,02,F],
    result(0,V,01,02,F),
    L) .

get_scenario(2,V,01,02,03,04,L) :- setof([[V,01,02,F1],[V,03,04,F2]],
    (result(0,V,01,02,F1), result(1,V,03,04,F2),
    F1=<F2),
    L) .

get_scenario(3,V,01,02,03,04,05,06,L) :-
    setof([[V,01,02,F1],[V,03,04,F2],[V,05,06,F3]],
    (result(0,V,01,02,F1), result(1,V,03,04,F2),
    result(2,V,05,06,F3),
    F1=<F2, F2=<F3),
    L) .

```

```
get_scenario(4,V,01,02,03,04,05,06,07,08,L) :-  
    setof([[V,01,02,F1],[V,03,04,F2],[V,05,06,F3],  
          [V,07,08,F4]],  
          (result(0,V,01,02,F1), result(1,V,03,04,F2),  
           result(2,V,05,06,F3), result(3,V,07,08,F4),  
           F1=<F2, F2=<F3, F3=<F4),  
          L).  
  
get_scenario(5,V,01,02,03,04,05,06,07,08,09,010,L) :-  
    setof([[V,01,02,F1],[V,03,04,F2],[V,05,06,F3],  
          [V,07,08,F4],[V,09,010,F5]],  
          (result(0,V,01,02,F1), result(1,V,03,04,F2),  
           result(2,V,05,06,F3), result(3,V,07,08,F4),  
           result(4,V,09,010,F5),  
           F1=<F2, F2=<F3, F3=<F4, F4=<F5),  
          L).
```

### A.3 Sample Facts-base Snapshot

A snapshot of the facts-base for a sample video is given to clarify the understanding of the object information and corresponding event information. This section shows the extraction of a deposit event such that an object enters the scene, then a split occurs and one object, classified as human, continues its motion, whereas the other object stops.

```
    /* general facts */
1   grid_size(1,16).

    /* object information facts */
2   object-info(1, obj001, human, blue, box, 12, 5).
3   object-info(1, obj001, human, blue, box, 13, 6).
4   object-info(1, obj001, human, blue, box, 14, 6).
5   object-info(1, obj001, human, blue, box, 15, 7).
6   object-info(1, obj001, human, blue, box, 16, 8).
7   object_info(1, obj001, human, blue, box, 17, 8).
8   object_info(1, obj002, non_human, white, box, 14, 6).
9   object_info(1, obj002, non_human, white, box, 15, 6).
10  object_info(1, obj002, non_human, white, box, 16, 6).
11  object_info(1, obj002, non_human, white, box, 17, 6).

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

    /* scenario-based meta-data */
17  enter(1, obj001, 12, 5).
18  move-east(1, obj001, 13, 6).
19  enter(1, obj002, 14, 6).
```

```
20  move-east(1, obj001, 15, 7).
21  deposit(1, obj001, obj002, west, 15, 6).
22  move-east(1, obj001, 16, 8).
23  stop(1, obj002, 16, 6).
24  leave(1, obj001, 17, 8).
...
```