Learning, Detection, Representation, Indexing and Retrieval of Multi-Agent Events in Videos

by

Asaad Hakeem
B.Sc., Ghulam Ishaq Khan Institute, 2000
M.Sc., University of Central Florida, 2006

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2007

Major Professor: Mubarak Shah

UMI Number: 3256923

UMI Microform 3256923

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

The world that we live in is a complex network of *agents* and their interactions which are termed as *events*. An instance of an event is composed of directly measurable low-level actions (which I term *sub-events*) having a temporal order. Also, the agents can act independently (e.g. voting) as well as collectively (e.g. scoring a touch-down in a football game) to perform an event. With the dawn of the new millennium, the low-level vision tasks such as segmentation, object classification, and tracking have become fairly robust. But a representational gap still exists between low-level measurements and high-level understanding of video sequences. This dissertation is an effort to bridge that gap where I propose novel learning, detection, representation, indexing and retrieval approaches for multi-agent events in videos.

In order to achieve the goal of high-level understanding of videos, firstly, I apply statistical *learning* techniques to model the multiple agent events. For that purpose, I use the training videos to model the events by estimating the conditional dependencies between sub-events. Thus, given a video sequence, I track the people (heads and hand regions) and objects using a *Meanshift tracker*. An underlying rule-based system detects the sub-events using the tracked trajectories of the people and objects, based on their relative motion. Next, an *event model* is constructed by estimating the sub-event dependencies, that is, how frequently sub-event

B occurs given that sub-event A has occurred. The advantages of such an event model are two-fold. First, I do not require prior knowledge of the number of agents involved in an event. Second, no assumptions are made about the length of an event.

Secondly, after learning the event models, I detect events in a novel video by using graph clustering techniques. To that end, I construct a graph of temporally ordered sub-events occurring in the novel video. Next, using the learnt event model, I estimate a *weight matrix* of conditional dependencies between sub-events in the novel video. Further application of Normalized Cut (graph clustering technique) on the estimated weight matrix facilitate in detecting events in the novel video. The principal assumption made in this work is that the events are composed of highly correlated chains of sub-events that have high conditional dependency (association) within the cluster and relatively low conditional dependency (dis-association) between clusters.

Thirdly, in order to represent the detected events, I propose an extension of CASE representation of natural languages. I extend CASE to allow the representation of temporal structure between sub-events. Also, in order to capture both multi-agent and multi-threaded events, I introduce a hierarchical CASE representation of events in terms of sub-events and case-lists. The essence of the proposition is that, based on the temporal relationships of the agent motions and a description of its state, it is possible to build a formal description of an event. Furthermore, I recognize the importance of representing the variations in the temporal order of sub-events, that may occur in an event, and encode the temporal probabilities directly into my event representation. The proposed extended representation with

probabilistic temporal encoding is termed P-CASE that allows a plausible means of interface between users and the computer. Using the P-CASE representation I automatically encode the event ontology from training videos. This offers a significant advantage, since the domain experts do not have to go through the tedious task of determining the structure of events by browsing all the videos.

Finally, I utilize the event representation for indexing and retrieval of events. Given the different instances of a particular event, I index the events using the P-CASE representation. Next, given a query in the P-CASE representation, event retrieval is performed using a two-level search. At the first level, a maximum likelihood estimate of the query event with the different indexed event models is computed. This provides the maximum matching event. At the second level, a matching score is obtained for all the event instances belonging to the maximum matched event model, using a weighted Jaccard similarity measure. Extensive experimentation was conducted for the detection, representation, indexing and retrieval of multiple agent events in videos of the meeting, surveillance, and railroad monitoring domains. To that end, the *Semoran* system was developed that takes in user inputs in any of the three forms for event retrieval: using pre-defined queries in P-CASE representation, using custom queries in P-CASE representation, or query by example video. The system then searches the entire database and returns the matched videos to the user. I used seven standard video datasets from the computer vision community as well as my own videos for testing the robustness of the proposed methods.

This dissertation is dedicated to my loving wife *Madiha* and my caring parents *Rizwana* and *Masood Hakeem*.

Acknowledgments

I would like to thank my dissertation advisor *Dr. Mubarak Shah* for his guidance, help, and dedication; *Drs. Steve Fiore, Annie Wu,* and *Niels da Vitoria Lobo* for serving as my committee members and for their valuable feedback; *Cherry Tran, Anne Hunter,* and *Linda Lockey* for proof reading my dissertation; *Dr. Yaser Sheikh* and *Saad Ali* for the discussion of ideas and helpful insights; *Andrew Horner* for generating the video tracks; and *Dr. Charles Hughes* and *Jenny Shen* for all their help and support.

I would also like to thank my loving wife *Madiha* for all her encouragement, patience, and help; my mother *Rizwana*, my father *Masood*, my mother-in-law *Aaila*, my father-in-law *Rass*, my sister *Saeedeh*, and my brother *Raza* for their support, encouragement and prayers throughout my graduate studies.

TABLE OF CONTENTS

LIST OF TABLES . . . . . **xii**

LIST OF FIGURES . . . . . **xiii**

CHAPTER 1 INTRODUCTION **1**

1.1 Event Learning 2

1.2 Event Detection 3

1.3 Event Representation 4

1.4 Event Indexing and Retrieval 6

1.5 Organization of the Dissertation 7

CHAPTER 2 RELATED WORK **9**

2.1 Low-Level Feature Detection 10

2.1.1 Tracking 11

2.1.2 Object Classification 14

2.1.3 Sub-Event Detection 18

2.2 Event Detection 20
2.2.1 Pre-defined Event Models Based Event Detection 21
2.2.2 Learnt Event Models Based Event Detection 23
2.2.3 Clustering Based Event Detection 24
2.3 Event Representation 25
2.4 Event Indexing and Retrieval 27
2.5 Conclusions 28

CHAPTER 3 LOW-LEVEL FEATURE DETECTION 31

3.1 Meanshift Tracking 33
3.2 Sub-Event Detection 35
3.3 Results 37
3.4 Discussion 40

CHAPTER 4 EVENT LEARNING AND DETECTION 41

4.1 Learning the Event Model 45
4.1.1 Capturing Temporal Order of Sub-events using Allen's Temporal Algebra 46
4.1.2 Event Modeling using Edge-Weighted Directed Hypergraph 48
4.2 Event Detection 51
4.2.1 Estimating the Probabilistic Weight Matrix of Sub-event Dependencies 51

4.2.2 Graph Clustering using Normalized Cuts 53

4.3 Results 55

4.4 Discussion 58

CHAPTER 5 EVENT REPRESENTATION 65

5.1 CASE Framework 68

5.2 Extended CASE 70

5.2.1 Hierarchical Representation 70

5.2.2 Temporal Logic 71

5.2.3 Causality 73

5.2.4 Variation in Temporal Order 74

5.3 Results 77

5.3.1 P-CASE Representation for Railroad Monitoring Domain 77

5.4 Discussion 82

CHAPTER 6 EVENT INDEXING AND RETRIEVAL 84

6.1 Semoran System 90

6.2 Event Indexing 93

6.2.1 Comparison of P-CASE with HMMs 93

6.2.2 Comparison of P-CASE with Bayesian Networks 95

6.3 Event Retrieval 96

6.4 Results 98

6.5 Discussion 103

CHAPTER 7 SUMMARY AND FUTURE WORK 114

APPENDIX A: SUB-EVENT DETECTION RULES 119

APPENDIX B: PROOF OF EQUIVALENCY FOR W AND W' BASED MINIMIZATIONS 125

APPENDIX C: MOST LIKELY SEQUENCE OF SUB-EVENTS FOR EVENTS IN SEMARON SYSTEM 127

LIST OF REFERENCES 134

LIST OF TABLES

3.1 SUMMARY OF UNIQUE SUB-EVENTS 36

3.2 SUMMARY OF SUB-EVENT DETECTION RATE 39

4.1 SUMMARY OF TRAINING VIDEOS 55

4.2 SUMMARY OF DIFFERENT DATASETS USED FOR EVENT LEARNING 56

4.3 RESULTS OF TESTING VIDEOS 58

6.1 SUMMARY OF INDEXED VIDEOS IN SEMORAN DATABASE 99

6.2 SUMMARY OF EVENT RETRIEVAL USING THE SEMORAN SYSTEM 102

LIST OF FIGURES

3.1 Results of Meanshift tracking for videos in different domains. (a) Surveillance (b) Railroad Monitoring (c) Meetings. 37

3.2 Automated detection of sub-events for stealing video. Using the tracked trajectories, the sub-events of each agent are detected, and frames 37, 119, 127, and 138 of the video are shown. 38

4.1 Allen's interval algebra describing temporal relations between durative sub-events T_1 and T_2. 46

4.2 Partial graph for a sequence containing two agents performing actions simultaneously. The sub-events are represented by the vertices and the temporal relationships between sub-events are shown as directed edges between vertices. Agent1's sub-events are greyed while Agent2's are white to provide a visual distinction between their sub-events. 48

4.3 Partial sub-event dependency graph for a sample video containing multiple people performing *voting* events in various styles. The vertices represent the sub-events, while the conditional probabilities between sub-events are represented by the weights on the hyperarcs. Note that a single example of hyperarcs with cardinality of 3 and 4 are shown respectively in green and red, to keep the figure comprehendible. Also, the circled number on the hyperarc represents the temporal order index in P_i. 50

4.4 The estimated *weight matrix* and *Ncut* application for a novel video. (a) The estimated weight matrix using the SDG of *voting* event, using *Normalized cut* two voting events are automatically segmented and are shown as red and blue patches. (b) The estimated weight matrix using the SDG of *object passing* event, using *Ncut* the object passing event is automatically segmented and is shown as the green patch. 52

4.5 Event detection results using normalized cuts for meeting domain test video. (a)-(d) represent frames 328, 560, 755, and 1,375 respectively of meeting video consisting of 1,551 frames. (e) Time indexed clustering results for meeting video, where the top bar shows the actual event detection results and the bottom bar denotes the ground truth of the events. 61

4.6 Event detection results using normalized cuts for surveillance domain test video1. (a)-(d) represent frames 159, 2,388, 2,874, and 3,125 respectively of surveillance video1 consisting of 3,580 frames. (e) Time indexed clustering results for surveillance video1, where the top bar shows the actual event detection results and the bottom bar denotes the ground truth of the events. . 62

4.7 Event detection results using normalized cuts for surveillance domain test video2. (a)-(d) represent frame 223, 1,084, 2,191, and 2,703 respectively of surveillance video2 consisting of 4,256 frames. (e) Time indexed clustering results for surveillance video2. 63

4.8 Event detection results using normalized cuts for railroad monitoring domain test video. (a)-(d) represent frames 740, 1,354, 1,966, and 2,251 respectively of railroad monitoring video consisting of 2,260 frames. (e) Time indexed clustering results for railroad monitoring video, where the top bar shows the actual event detection results and the bottom bar denotes the ground truth of the events. 64

5.1 P-CASE representation for the *object passing* event. Each node is a sub-event encoded by a complete case-frame, and the weights on directed edges represent the probability of occurrence of a specific temporal relationship between sub-events, while the weights outside the brackets (in blue) are the conditional probabilities between sub-events. The white and grey vertices represent sub-events of agents one and two respectively. 74

5.2 On-line extended CASE representation of video sequences. (a) Representation at frame 150/237 for the railroad monitoring video (b) PETS sequence representation at frame 1,446/2,000. 76

6.1 Event retrieval using pre-defined queries. 90

6.2 Event retrieval using custom queries. 91

6.3 Event retrieval using query by example video. 92

6.4 Event matching using the weighted Jaccard coefficient. A predefined event graph consisting of six vertices (case-frames) is matched with an event graph of a video sequence consisting of 148 vertices (case-frames). The correct match occurs at the graph node at frame 12 (the similarity maximum is indicated by the dotted red line). From top-left to bottom-right, the pre-defined predicate is perturbed so that a progressively greater number of *cases* within the case-frames mismatch. 100

6.5 Recall level precision plots for the three domains. 102

6.6 Average time for event retrieval given a query, in the meeting (domain1), surveillance (domain2), and railroad (domain3) domains respectively. 103

6.7 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Voting (b) Chasing 105

6.8 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Enter and sit (b) Fighting 106

6.9 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Loading (b) Object drop 107

6.10 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Object exchange (b) Object passing 108

6.11 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Person enters danger-zone while gate arms moving (b) Sneaking 109

6.12 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Stand and leave (b) Stealing 110

6.13 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Train enters danger-zone while gate arms moving (b) Unloading 111

6.14 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Vehicle enters danger-zone while gate arms moving (b) Vehicle exits danger-zone while gate arms moving 112

6.15 Automatically generated P-CASE representation using indexed data in Semoran system. (a) Vehicle stops outside danger-zone (b) Argument 113

CHAPTER 1

INTRODUCTION

Computer vision research started in the 1960's as a summer project in artificial intelligence, where the goal was to understand images by recognizing blocks and defining the relationship between them. This problem evolved in the 1970's to solve low level vision problems such as object segmentation using edge detection. The 1980's concentrated around recovering 3D structure of objects using "shape from X" methods, where X referred to motion, shading, and stereo. In the 1990's, the research revolved around acquiring scene geometry and information using active vision, which involves changing the sensor orientation and location, and use low level vision to achieve the goal. With the advent of cheap high speed computing in the late 1990's, research entered the video domain where motion was used as a visual cue for object tracking and action recognition. Compared to a single image, a sequence of images introduced a new dimension: time, and a new constraint: temporal consistency.

With the dawn of the new millennium these low-level vision tasks such as segmentation, object classification, and tracking have become fairly robust; but a representational gap still exists between low-level measurements and high-level understanding of video sequences. In

this dissertation, I outline the approaches to learning, detection, representation, indexing and retrieval of multi-agent events in videos to bridge this gap.

Given a set of training videos, such that each video consists of one type of event having spatial and temporal variations, I learn the event models. This process of constructing event models using training videos is called *event learning*. Since I know the type of event in each training video, it is considered supervised learning. Using the learnt event models, I find events in novel videos, and this process is termed as *event detection*.

The primary objective of this work is to detect and learn the complex interactions of the multiple agents performing multiple actions in the form of events. I do not assume any prior knowledge about the number of agents involved in the interaction or the length of the event. Another objective of this work is to present a coherent representation of these events, as a means to encode the relationships between the agents and the objects participating in an event, and to index the detected events for future retrieval. Below I provide an overview of proposed methods for learning, detection, representation, indexing and retrieval of multiple agent events in videos.

1.1 Event Learning

In order to learn the events from training videos, firstly, I introduce a graph that depicts the temporal relationships between sub-events in a video. These temporal relationships are based on the interval algebra in [AF94], which is a more descriptive model of relationships

compared to the low level abstract relationship model of HMMs. The purpose of this graph is to encode the complete temporal order of sub-events occurring in a video. The temporal order of sub-events are further utilized in extracting the conditional dependency between sub-events. Secondly, using this graph, I determine the *sub-event dependency graph* representing the temporal conditional dependency between sub-events. The sub-event dependency graph is the learnt event model that encodes the higher order Markov dependencies between sub-events and is scalable to the number of agents involved in the event. The main advantages of my event model are:

1. The temporal relations are more descriptive relationships between sub-events compared to the low level abstract relationship models of HMMs, Dynamic Bayesian Networks etc.

2. The event model does not make any assumptions about the length of an event.

3. The event model is scalable to the number of agents involved in an event since it models the sub-event dependencies instead of the agent processes.

1.2 Event Detection

Event detection in novel videos proceeds by estimating a weight matrix of conditional dependencies between the detected sub-events. The weights on edges between sub-events are recovered using the learnt event model. This weight matrix is then used for spectral graph

partitioning. Thus, *normalized cut* is applied recursively to this weight matrix, to cluster the highly correlated sub-events. These clusters represent the detected events for a specific event model, and the event structure composed of sub-events and their temporal order is extracted using graph partitioning. Furthermore, different weight matrices are estimated for each event model, and normalized cut is applied recursively to extract all the events present in the novel video. The main advantages of my event detection scheme are:

1. The event detection does not make any assumptions about the length of an event since the graph clustering will cluster highly correlated events of any length.

2. The event detection is scalable to the number of agents involved in an event since the graph clustering will cluster highly correlated events involving any number of agents.

1.3 Event Representation

In order to represent the events, I extend the CASE [F68] representation of the natural language. CASE was primarily used for syntactic analysis of natural languages, and while it provides a promising foundation for event representation it has several limitations for that end. I therefore propose four critical extensions to CASE for the representation of events:

1. Accommodating multiple agents and multiple threads in an event.

2. Supporting the inclusion of temporal information into the representation.

3. Supporting the inclusion of causal information into the representation.

4. Accommodating variation in the temporal order of sub-events.

I also propose a novel event graph representation for the detected events in video sequences, having temporal relationships between sub-events. Hence, unlike almost all previous work, I use both temporal structure and an environment descriptor simultaneously to represent an event. I also recognize the importance of representing the variations in temporal order of sub-events, that occur in an event and encode it directly into my representation, which I term P-CASE. These variations in the temporal order of sub-events, occur due to the style of execution of events for different agents. The practical need for formal representation of events is best illustrated through possible applications. These include:

(1) ***Surveillance***: By definition, surveillance applications require the detection of peculiar events. Event representations can be used for prior definition of what constitutes an interesting event in any given domain, allowing automation of area surveillance.

(2) ***Annotation and Indexing***: In the spirit of MPEG-7, video sequences may be annotated autonomously based on their content. Using an event representation, video content may be annotated and indexed according to the occurred events.

(3) ***Event Browsing***: Given a query for a certain event, defined in terms of an event representation, similar instances can be retrieved from a database of annotated clips.

1.4 Event Indexing and Retrieval

Finally, I propose to utilize the event representation for indexing and retrieval of events. Given the different instances of a particular event, I build an event index in the form of P-CASE representation, encoding the variation in the temporal order of sub-events occurring in an event. Given a query in the P-CASE representation, event retrieval is a two-level process. At the first level, a maximum likelihood (ML) estimate is computed with the different event models. The event model with the ML estimate provides the maximum matching event. At the second level, I find the percentage match of the query event with all the event *instances* belonging to the maximum matched event model, using a weighted Jaccard similarity measure. The weights in the Jaccard measure are obtained using term-frequency and inverse document frequency (TF-IDF) scheme, borrowed from Lucene full-text indexing. In text search, the TF-IDF scheme is used to return the ranked search results, whereas for event retrieval, the TF-IDF scheme helps in weighting the importance of a particular *case* in an event. The main advantages of my event indexing and retrieval scheme are:

1. The event indexing scheme is human readable, since the P-CASE representation is an extension of CASE representation of natural languages.

2. The event retrieval requires less number of hits since it rules out most of the events (and their instances) during the first level search.

3. The event indexing and retrieval is scalable since new events can be added to the database.

With the different methods in my dissertation summarized, the following section outlines the organization of the rest of the dissertation.

1.5 Organization of the Dissertation

In Chapter 2, I discuss the related work for low-level feature detection, as well as learning, detection, representation, indexing and retrieval of events. Section 2.1 discusses the various low-level feature detection methods such as object tracking and sub-event detection. Section 2.2 describes the event detection methods where I detail the different approaches used to achieve the goal. Section 2.3 discuses the various event representations used to describe an event. Section 2.4 details the current state-of-the-art in event indexing and retrieval and discuss their limitations.

In Chapter 3, I describe the low-level feature detection and tracking methods. I further detail the Meanshift tracker used in my experiments. I end the chapter by describing the rule-based system used for sub-event detection. In Chapter 4, I detail how I utilize the detected events to learn the event model. I further describe the graph clustering technique used to detect the events in novel video. In Chapter 5, I discuss the CASE representation of the natural language and describe its limitation for event representation. I detail the proposed extensions to CASE representation to cater for multiple agent and multi-threaded

events, incorporation of temporal order of sub-events and variation in the temporal order of sub-events. All of these extensions to the CASE representation allows the representation of events and I term the final event representation as P-CASE.

Chapter 6 summarizes the Semoran system, which uses the P-CASE representation of the detected events for the purpose of event indexing and retrieval. I describe the event indexing scheme where I use P-CASE representation to build the indexed event models. I then calculate a maximum likelihood estimate of the query event with all the event models to find the matching event, and further utilize a weighted Jaccard measure to find the similarity score of all instances of the matching event with the query event. Finally, in Chapter 7, I summarize my proposed methods for event learning, detection, representation, indexing and retrieval of multiple agent events and provide some future directions to the current work.

CHAPTER 2

RELATED WORK

Events are high level concepts that are composed of sub-events having a temporal order. The agents can act independently (e.g. voting) as well as collectively (e.g. touch-down in a football game) to perform an event. Hence, in the enterprise of machine vision, the ability to detect and learn the observed events must be one of the ultimate goals. Another important aspect in the artificial intelligence and multimedia communities is the ability to represent and index the detected events for future retrieval of similar events. With the dawn of the new millennium, the low-level vision tasks such as segmentation, object classification, and tracking have become fairly robust. But a representational gap still exists between low-level measurements and high-level understanding of video sequences. In this chapter, I discuss the related work for the different steps involved in achieving high-level video understanding, and point at their limitations for videos containing multiple agent events.

The first step towards achieving high-level video understanding is the detection and tracking of different agents and objects participating an event. Using the tracked trajectories of agents and objects, the different sub-events are detected. This step is discussed in Section 2.1 where I detail the various low-level feature detection methods for reaching the goal. The second step is the detection of events using the detected sub-events and their temporal order.

This is described in Section 2.2 where I detail the different event detection methods. The third step is the representation of the detected events for human understanding, interpretation, and alert (in case of suspicious or unusual behavior). This step is given in Section 2.3 where I discuss the various event representations used to describe an event. The final step in achieving high level understanding of videos is the indexing of events for future retrieval. This is provided in Section 2.4 which details the current state-of-the-art in event indexing and retrieval and discuss their limitations. I now detail the related work for each step in the following sections.

2.1 Low-Level Feature Detection

An instance of an event is composed of directly measurable low-level actions, which are termed sub-events, having a temporal order. In this section, I discuss the various methods used to detect the sub-events in different domains. In order to detect sub-events, two low-level tasks of object tracking and classification are required as a pre-requisite. Tracking is defined as a problem of finding the trajectory of an object as it moves in the scene, while object classification is defined as a problem of labeling what type of object is in the scene (e.g. car or person). The various methods for object tracking and classification are discussed in Sections 2.1.1 and 2.1.2 respectively. Once the objects are detected, tracked, and classified I can detect the sub-events. The various sub-event detection methods are described in Section 2.1.3.

2.1.1 Tracking

Object tracking is one of the fundamental problems in the field of computer vision. It has gained wide spread attention with the advent of high speed computing and increased need for automated surveillance. Tracking is defined as a problem of finding the position of an object as it moves along in the scene. There are several complexities in object detection and tracking such as:

1. occlusion of objects (partial/complete/self)
2. changes in scene illumination (due to cloud cover etc.)
3. image noise (due to sensor limitations etc.)
4. size of the object (being very small)
5. shape of the object (non-rigid, articulated, etc.)
6. non-linear or non-smooth object motion
7. shadows
8. drift in object appearance model

A good tracker should be able to tackle most of the above mentioned problems by imposing certain constraints on the object motion and appearance. Almost all tracking algorithms make certain assumptions to simplify the problem. For example, most algorithms assume

that the object motion and appearance will change smoothly over time, some make assumptions on the number and size of objects, while other make assumptions about object appearance (e.g. skin colored) and shape (e.g. elliptical, rectangular etc.). Numerous approaches to object tracking exists, based on the domain under consideration. I discuss object tracking in the two scenes (in which I conducted my experiments) and the interested reader is referred to [YJS06] for a detailed review of object tracking methods.

2.1.1.1 Tracking in Surveillance Scenes

The videos in various surveillance scenes usually consist of objects that are smaller in size, as compared to the background, and are fast moving. A background is defined as that part of the video that either does not move/change over time or changes smoothly over time. In surveillance videos there are usually multiple objects in the scene and they have frequent occlusions with each other. Background subtraction techniques [SG00, JSS02] are commonly used to detect objects in these videos. Due to the presence of multiple concurrent objects in the scene, it is important to have consistent labelling of these objects throughout the video sequence. That is, given two frames and a set of objects in each frame, in order to learn the appearance model of each object I must know which object in the first set corresponds to which object in the second set. The problem becomes even more complex when these objects occlude each other in the scene. Thus, in the surveillance domain, a system must detect and track objects as well as handle occlusion, entries and exits in the scene.

The background subtraction was initially proposed by Stauffer and Grimson [SG00] which was further extended by Javed et al. [JSS02]. In their extension, they propose multiple levels of processing during background subtraction. The first level is the pixel level processing that separately uses color-based and gradient-based distributions to find pixels belonging to the foreground or the background. The second level is the region level processing that integrates the gradient and color information. A connected component algorithm is applied to group all foreground pixels into regions. Any foreground region that corresponds to an object will have high values of gradient-based background subtraction at its boundaries. This will not be true for falsely detected regions and so they are added to the background regions. This method handles common problems in most background subtraction algorithms such as quick illumination changes due to adverse weather conditions, relocation of the background objects (e.g. repositioning of a chair), and initializing the background model with moving objects.

2.1.1.2 Tracking in Meeting Scenes

Background modelling techniques detect people and vehicles fairly robustly in the surveillance and railroad monitoring scenes. However, for the videos in the meeting scenes, objects are usually present at the start of the video and they persist in field of view of the camera throughout the entire sequence. Therefore a model of the background cannot be estimated in such videos, and thus background subtraction techniques cannot be used for tracking.

Appearance-based methods such as Meanshift [CRM03] or contour-based object tracking [YLS04] methods can be used in such scenes.

The Meanshift algorithm tracks the initially detected object using its color histogram. A histogram captures the distribution of color in an image patch. The color space is quantized into several discrete bins and the number of pixels with the same color are recorded in the corresponding bin. Thus, the object model $\mathbf{q}$ is an m-bin histogram, representing the object's probability density function (pdf). In the subsequent frame, the candidate location $\mathbf{y}$ of an object is characterized by its pdf $\mathbf{p(y)}$.

In contour-based methods, object tracking is treated as a two-class discriminant analysis of pixels into regions belonging to object R_{obj} and background R_{bck}, which depends upon object features, energy functional, and the energy minimization technique. The object features under consideration are appearance and shape, and the appearance features are composed of color and texture. Pixels are clustered as object or background by the *independent opinion pooling* strategy [B85]. The shape of the object is learnt over time (t), based on the object contour Γ and is given by $P_{shape} = P(\varphi(R_i^t)|\Gamma^t)$, where R_i are the object regions and φ is the partitioning operator that divides the image into object and background regions.

2.1.2 Object Classification

Object classification is also one of the fundamental problems in the field of computer vision. It has gained wide spread attention with the advent of high speed computing and

increased need for object categorization in automated surveillance. Classification is defined as a problem of finding the type of an object as it moves along in the scene, using its motion and appearance. There are several complexities in object classification such as:

1. change in object viewpoint
2. change in object pose
3. occlusion of objects (partial/complete/self)
4. image noise (due to sensor limitations etc.)
5. size of the object (being very small)
6. shape of the object (non-rigid, articulated, etc.)
7. shadows

A good classifier should be able to tackle most of the above mentioned problems by imposing certain constraints on the object motion and appearance. Almost all classification algorithms make certain assumptions to relax some of the above complexities and simplify the problem. I discuss object classification using three broadly used methods: motion periodicity, supervised learning, and semi-supervised learning based classification.

2.1.2.1 Motion Periodicity based Classification

These methods classify objects as person or vehicle based on the periodicity of their motion. The intuition behind such methods is that walking people undergo periodic motion while vehicles do not, thus periodicity detection can be used distinguish between them. Tsai *et. al.* [TSKK00] analyzed certain points on the objects and periodicity was detected by analyzing Fourier descriptors of the smoothed spatio-temporal curvature of those point trajectories. Polana and Nelson [PN97] also recognized periodic activities using Fourier transform of point trajectories of an object that was obtained by using normal flow. Javed and Shah [JS02] introduced Recurrent Motion Images (RMI) to detect periodic motion. These images were obtained by iteratively performing the XOR operation and adding the scale and motion compensated silhouettes of the objects. Large values in the RMIs indicated periodic motion.

One limitation of all the above mentioned methods is that object trajectories are required to compute periodicity, thus any errors in tracking also degrade the classification. In addition, these approaches are not extendable for classification of the objects that exhibit non-periodic motion.

2.1.2.2 Classification using Supervised Learning

The above limitation of periodicity based classifiers is overcome by learning the functions that map the image features of a particular class to a label using training examples from that

class. A variety of approaches have been proposed using this methodology including naive Bayes classifiers [SK00], Support Vector Machines (SVMs) [PP99] and Adaboost [SG00]. For surveillance scenarios, Adaboost is particularly suitable since it has been demonstrated to give low false alarm and high detection rates in real-time using simple Haar-like features. Boosting is a machine learning method, that combines simple classifiers (weak learners) into a single (strong) classifier, which is more accurate than any one of the weak learners. SVMs have been used in different scenarios, where the method maps the feature vectors to a higher dimension and chooses those feature vectors as support vectors, that are at the boundary between the two classes. During testing, the new feature vector is compared to the support vectors and the new feature is given the same class label as the support vectors that give the highest score. The limitations of the supervised classification approaches is that they require large number of training examples to learn the mapping functions. Also, they are not adaptive to view and pose variations in the object.

2.1.2.3 Classification using Semi-supervised Learning

One possible means to overcome the limitation of supervised learning methods requiring a large labeled training set is to learn from unlabeled data. A number of methods have been developed by the machine learning and pattern recognition community for training of classifiers using unlabeled data. The use of both labeled and unlabeled data for solving classification tasks was introduced by Nigam *et al.* [NMTM00] in the area of text and

information retrieval. They employed the EM algorithm to infer the missing labels of the unlabeled data. The co-training approach to learn from unlabeled data was proposed by Blum and Mitchell [BM98]. The basic idea is to train classifiers on two independent features (views) of the same data, using a relatively small number of examples. Then to use each classifier's prediction on the unlabeled examples to expand the training set of the other. Blum and Mitchell prove that co-training can find a very accurate classification rule, starting from a small quantity of labeled data if the two feature sets are statistically independent. Recently, Balcan *et al.* [BBY04] have shown that independence between the two views of the data is not a necessary assumption for co-training, instead weekly correlated views can also be used for co-training.

2.1.3 Sub-Event Detection

After discussing the object detection, tracking, and classification methods, I now discuss the sub-event detection methods. Initial approaches for sub-event detection involved methods that had pre-defined rules or constraints that formed event models. Later, event models were learnt using training examples. The techniques broadly used for low-level sub-event detection include rule-based systems, finite state machines (FSMs), stochastic context free grammar (SCFG), Hidden Markov Models (HMMs) and their variants, Bayesian Networks (BNs), and Dynamic Bayesian Networks (DBNs).

Rule-based system is an amalgamation of a list of functions that fire independently upon certain conditions in the feature set being satisfied. In case of sub-event detection, each function represents a particular sub-event that requires certain conditions in the motion trajectories to be satisfied. Later, simultaneous firing of functions was replaced by the concept of states that was introduced in a FSM. A FSM consists of nodes that represent states (which have a high level meaning e.g. sub-events move or stop etc.) and arcs that represent transition from one state to another. A transition only occurs upon certain a condition being satisfied in a particular state. Furthermore, probabilities were added on the arcs of a FSM to form HMM. As opposed to a FSM, an HMM looses the high level meaning of each state and is able to detect sub-events of varying length due to its dynamic time warping properties. An HMM is the most widely used model for sub-event detection and activity recognition, and has several variants, Coupled HMM and Hierarchical HMM being the most popular.

Bayesian Networks go one step further to HMMs where the structure has nodes representing states having high level meaning and arcs that represent conditional dependencies (casuality). It is able to model static dependencies between concepts, which is further extended by Dynamic Bayesian Networks that model concept dependencies that vary with time. Both HMMs and DBNs define hidden concepts using a set of hidden nodes. Using the hidden nodes and the transitions, they are able to detect streams of sub-events.

A stochastic context free grammar is composed of symbols and production rules. A probability is attached to each symbol and production rule that is learnt via training examples.

During testing, the overall probability of a feature set is evaluated by expanding the production rules and multiplying the probabilities using the Markovian assumption. The model that maximizes the probability is considered the detected sub-event.

Each of the above mentioned approaches have several limitations. They either manually encode the event models or provide constraints such as grammar or rules, to detect sub-events in novel videos. Most of these grammar rules are hand crafted and do not work in general scenarios. Also, the learning methods either model single person sub-events or require prior knowledge about the number of people involved in the events and variation in data may require complete re-training, so as to modify the model structure and parameters to accommodate those variations. Therefore, there is no single best method for the detection of sub-events and the choice of model varies with the domain under consideration.

2.2 Event Detection

In literature, a variety of approaches have been proposed for the detection of events in video sequences. Most of these approaches can be arranged into three categories based on their approach to event detection. First, there are approaches that detect events in videos based on pre-defined event models in the form of templates, rules, or constraints. These models are typically hand crafted. Among these include methods that utilize motion verbs [KHN91], intermodal collaboration using domain knowledge [BJ98], spatio-temporal motion segmentation [RA00], indexing of pose and velocity vectors of body parts [BWP02],

image ontology based recognition [MTB03], and taxonomy and ontology of domains [HS04]. Second, approaches that learn the event models [FMR98, BK00, IB00, HL04] using training data have been widely used in the area of event detection. Third, approaches that do not model the events [ZI01, RYS02, ZSV04], but utilize clustering methods for event detection.

2.2.1 Pre-defined Event Models Based Event Detection

Initial approaches for event detection involved methods that had pre-defined rules or constraints that formed event models. Among these methods were approaches that modelled events using state machines. Badler [B75] proposed event models using state graphs and primitive rules on artificial environments. The method used prior knowledge of the environment to resolve complex events rather than using motion information for event detection. Ayers and Shah [AS01] also used state machines for event detection, but as opposed to the above method, it utilized the motion data for event detection. Their method detected events in specified areas, which restricted the system and required a priori knowledge of the environment. Haering *et al.* [HQS00] proposed a three level event detection model, which applied neural networks on low-level features for object classification and shot summarization, and state machines as the high level event model that detected events based on the output of the low and mid-level models. Unlike previous methods, their method did not require prior knowledge of the environment for event detection. Later, there were methods that concentrated on detecting periodic events based on pre-defined event models. Polana and Nelson

[PN93] proposed a method for detecting periodic events such as "walking of a person", by using Fourier transform of different point trajectories, and classified events as periodic or non-periodic. The periodic motion was detected by averaging the fundamental frequency of the point trajectories, but was limited to detecting periodic events that had a constant cycle length. Yacoob and Black [YB98] also detected cyclic human motion using their parameterized model, but unlike the previous method, it detected periodic events with a variable cycle length. The detection was performed through eigenspace transformation of the observed data to the model data using Principal Component Analysis (PCA). Though these methods had high detection rates, they had limited application to recognition of events consisting of repeated patterns. Further approaches utilized constraints and grammar rules for the detection of events that had variability, and did not strictly follow the event model. The various methods in this event detection approach include causal grammar [B97], PNF propagation of temporal constraints [PB98], dynamic perception of actions [MJ98], force dynamics [S00], angular constraints on body components [AA01], stochastic grammars [MES03], and appearance and geometric constraints [SSL04]. All the above approaches either manually encode the event models or provide constraints such as grammar or rules, to detect events in novel videos.

2.2.2 Learnt Event Models Based Event Detection

Next, I discuss approaches that learn the event models using training data, instead of manually encoding the event models. [DB97] proposed temporal templates for event detection. Their method utilized a Motion Energy Image (MEI) and a scalar valued Motion History Image (MHI) to learn the temporal templates. Their method was sensitive to partial occlusions, since the motion of the trained temporal template did not have the occluded MEI and its associated MHI. [FMR98] detected events by learning the structure and parameters of Dynamic Bayesian Networks (DBNs) from both complete and incomplete data. Though their method was partially insensitive to occlusion, it was limited to inference of simple events, and required approximations for detecting complex events.

Methods that adopt Hidden Markov Models (HMMs) [MEH99, BK00, OLW02] and its variations such as coupled HMMs [ORP99], dynamic multi-linked HMMs [GX03], abstract hidden Markov memory models [NBV03], and layered HMMs [XMZ03] have been widely used in the area of event detection. These methods were the work horses for event detection in the surveillance of indoor and office environments, and sports domains. [IB00] proposed a Stochastic Context Free Grammar (SCFG) for event detection. The grammar rules were assigned probabilities based on the training data and were further utilized for event detection in novel videos. The primitive events were detected using HMMs and were parsed by the SCFG for error detection and correction of the detected events. Other methods for event detection using learnt event models include belief networks [IB99], shape activities [VCC03],

Support Vector Machines (SVMs) [HL04], and Bayesian Networks and probabilistic finite state machines [HNB04].

The above learning methods either model single person events or require prior knowledge about the number of people involved in the events and variation in data may require complete re-training, so as to modify the model structure and parameters to accommodate those variations. Also, there is no straight-forward method of expanding the domain to other events, once training has been completed. That is, if more events are added to the current domain or if I want to model events in a new domain, then, the existing models are re-trained using the new data and/or the model structure is defined manually for the new events.

2.2.3 Clustering Based Event Detection

Finally, I describe approaches that do not model events, but utilize clustering techniques for event detection. These methods of event detection include spatio-temporal derivatives [ZI01], and co-embedding prototypes [ZSV04]. Both methods find event segments by spectral graph partitioning of the weight matrix. The weight matrix is estimated by calculating a heuristic measure of similarity between video segments. These methods assume maximum length of an event and were restricted to a single person and a single threaded event detection. [RYS02] proposed human action recognition using spatio-temporal curvatures of 2D trajectories. Their method initiated without an event model and formed clusters of similar events based on their spatio-temporal curvatures. Their method is also restricted to single

person event detection, but it makes no assumptions about the length of an event and the spatio-temporal curvature based event representation was also view-invariant.

2.3 Event Representation

Although there are several methods in the computer vision and pattern recognition community that deals with the detection of events. What is missing in these approaches is the ability to extend their abstract event models to representations related to human understanding of events. One such natural language representation called CASE was proposed by Fillmore [F68] for language understanding. The basic unit of this representation is a case frame that has several elementary cases, such as an agentive, an instrumental, and a predicate. Using these case frames Fillmore analyzed languages, treating *all* languages generically.

However, CASE was primarily used for syntactic analysis of natural languages, and while it provides a promising foundation for event representation it has several limitations for that end. Firstly, since events are typically made up of a hierarchy of sub-events it is impossible to describe them as a succession of case frames. Second, these sub-events often have temporal and causal relationships between them, and CASE provides no mechanisms to represent these relationships. Furthermore, there might be simultaneous dependent or independent sub-events with multiple agentives, and change of location and instrumentals during events.

CASE was first investigated for event representation [N89], but the author did not investigate the temporal structure of events as the author was not concerned with event detection. More recently [KTF01] addressed some shortcomings in CASE for single person event detection with, SO- (source prefixed to case), GO- (goal prefixed to case) and SUB (child frame describing a sub-event). SO- and GO- are prefixed to the LOC (locative) case mostly describing the source and destination locations of the agent in the event. A concept hierarchy of action rules (case frames) was used to determine an action grammar (ontology) for the sequence of events. Also, using case frames based on events, they reconstructed the event sequence in the form of sentences. Their method worked well for single person action analysis using the CASE representation.

Also, there are other event representations such as hierarchical Event Representation Language (ERL) by Nevatia et al. [NZH03], graph representation of object trajectories by Medioni et al. [MCB01], Bayesian Networks based event representation by Hongeng et al. [HNB04], and video event ontologies by Nevatia et al. [NHB04] that provide varying degrees of representation to the actions and agents involved in the events. However, these works did not address important issues of temporal and causal relationships. Moreover, no mechanisms were proposed for multiple-agents or multi-threaded events.

2.4 Event Indexing and Retrieval

I can further utilize the event representation for indexing and retrieval of events in videos. Although there is a plethora of literature devoted to content-based image retrieval evident from the survey by Rui *et al.* [RHC99], most of the work is based on features retrieved from a single image. Naphade and Huang [NH01] use HMMs to index the database, in which low-level features are mapped to high-level concepts called multijects (which are probabilistic multimedia objects). Other methods for probabilistic video indexing and retrieval include Boreczky and Wilcox [BW97] that utilize audio and image features and Dimitrova *et al.* [DAW00] that use text and face features for indexing and retrieval. Non-probabilistic methods include the work by Katayama and Satoh [KS97] that employ an SR-tree to index the high dimensional feature vector and utilize nearest neighbor query search for retrieval of relevant data.

Recently, works by Chang *et al.* [CCMSZ97] and Natarajan and Nevatia [NN05] utilize video information in the feature vector for retrieval of similar videos from the database. Chang *et al.* [CCMSZ97] employ color, texture, shape, motion and time information as the feature vector for indexing and retrieval. Using their method, the user can retrieve videos of objects that have similar color and motion trajectories. Natarajan and Nevatia [NN05] in their EDF framework use an ontology of entities, actions and events to index the video events. Further, they utilize relational algebra to find complex events in the database. Though the methods proposed by Chang *et al.* [CCMSZ97] and Natarajan and Nevatia [NN05] are

promising for retrieval of video data, they lack the representative power to extend their abstract event model to representations related to human understanding of events. Also, these representations are not directly related to the human understanding of events, and thus, the user cannot define the query in a human representative language.

2.5 Conclusions

In this chapter I presented the related work on object detection, tracking, and classification which was used for the detection of sub-events in videos. These sub-events form a hierarchy to define an event and methods dealing with event detection were discussed. Furthermore, different event representations were detailed that try to bridge the gap between the low-level features and high-level concepts. Finally, I presented the work on indexing and retrieval of image and video data.

What is missing in these approaches is the ability to model long, complex events involving multiple agents performing multiple actions. Can these approaches be used to automatically learn events involving an unknown number of agents? Will the learnt event model still hold for a novel video, in the case of interfering events from an independent agent? Can these approaches extend their abstract event model to representations related to human understanding of events? Can events be compared on the basis of these representations? How are these representations related to human understanding of events?

The above mentioned questions are discussed in detail in their respective sections but are briefly answered here for completeness. Almost all event models require prior knowledge about the number of agents involved in the event, where as my learnt event model is scalable to the number of agents involved in an event. The reason is that instead of modelling agent processes (like Hidden Markov Models, Dynamic Bayesian Networks etc.) I model the sub-event dependency for all agents simultaneously i.e. which sub-event occurs more frequently after another sub-event for a given event. Thus, my event model is agnostic to the number of agents involved in the event. Also, since most methods detect events by estimating a posterior probability of a sub-event sequence, independent agent actions in the sub-event sequence reduces the overall posterior probability, where as I use graph clustering techniques for event detection. Thus, my method clusters events with high edge weights within the cluster and segments out independent agent actions as those actions have low edge weights with the rest of the sub-events belonging to the event. Furthermore, most event models and representations are abstract and complex, with no notion of human readability, where as my event representation is an extension of the CASE representation that was used for syntactic analysis of the natural language. It has explicit cases for agents, location, dative etc. that completely describes the event. I believe it is easier for humans to relate to this event representation.

In this dissertation, event models are learnt from training data, and are used for event detection in novel videos. *Event learning* is formulated as modelling conditional dependencies between sub-events while *event detection* is treated as a graph-theoretic clustering problem.

The novelty of my method, compared to the above mentioned methods, is the ability to detect multiple agents performing multiple actions in the form of events, without prior knowledge about the number of agents involved in the interaction and the length of the event. Also, I present a coherent representation of these events as a means to encode the relationships between agents and objects participating in an event and to index these events for future retrieval using human understandable queries.

CHAPTER 3

LOW-LEVEL FEATURE DETECTION

An instance of an event is a composition of directly measurable low-level actions, which I term sub-events, having a temporal order. In this chapter I discuss the methods to detect these sub-events. In order to detect sub-events, three low-level tasks of object detection, classification and tracking are addressed. The videos in various surveillance domains usually consist of objects that are smaller in size (as compared to the background) and are fast moving. There are usually multiple objects in the scene and they have frequent occlusions with each other. Background subtraction techniques [SG00, JSS02] are commonly used to detect objects in these videos. Due to the presence of multiple concurrent objects in the scene, it is important to have consistent labelling of these objects throughout the video sequence. That is, given two frames and a set of objects in each frame, in order to learn the appearance model of each object I must know which object in the first set corresponds to which object in the second set. The problem becomes even more complex when these objects occlude each other in the scene. Thus, in the surveillance domain, a system must detect and track objects as well as handle occlusion, entries and exits in the scene.

The background subtraction used for object detection is based on an extension of Stauffer and Grimson [SG00] by Javed et al. [JSS02]. In their extension, they propose multiple levels

of processing during background subtraction. The first level is the pixel level processing that separately uses color-based and gradient-based distributions to find pixels belonging to the foreground or the background. The second level is the region level processing that integrates the gradient and color information. A connected component algorithm is applied to group all foreground pixels into regions. Any foreground region that corresponds to an object will have high values of gradient-based background subtraction at its boundaries. This will not be true for falsely detected regions and so they are added to the background regions. This method handles common problems in most background subtraction algorithms such as quick illumination changes due to adverse weather conditions, relocation of the background objects (e.g. repositioning of a chair), and initializing the background model with moving objects.

Background modelling techniques detect people and vehicles fairly robustly in the surveillance and railroad monitoring domains, however, for the meeting domain and for smaller objects such as bags and books, the initial object identification and labelling were performed manually. Further tracking was attained using MEANSHIFT [CRM03] algorithm, which is described next. Though techniques like [BR94, WB96, WAD97, JS02, JSC04, SS05] could have been used for automated object labelling and multiple agent tracking, I opted for a simpler solution as my contribution is in event detection.

3.1 Meanshift Tracking

In simple terms, the Meanshift algorithm tracks the initially detected object using its color histogram. A histogram captures the distribution of color in an image patch. The color space is quantized into several discrete bins and the number of pixels with the same color are recorded in the corresponding bin. Thus, the object model **q** is an m-bin histogram, representing the object's probability density function (pdf). In the subsequent frame, the candidate location **y** of an object is characterized by its pdf **p(y)**. Thus I have,

$$object\ model \quad : \quad q = \{q_u\}_{u=1\ldots m} \qquad \sum_{u=1}^{m} q_u = 1 \tag{3.1}$$

$$object\ candidate \quad : \quad p(y) = \{p_u(y)\}_{u=1\ldots m} \qquad \sum_{u=1}^{m} p_u(y) = 1 \tag{3.2}$$

An object is represented by an ellipsoidal region in an image. All the objects are normalized to a unit circle by independently rescaling the row and column dimensions by h_x and h_y respectively. Thus, the normalized object model and object candidate are given by:

$$q_u = C \sum_{i=1}^{n} k(x_i^2)\delta[b(x_i) - u] \tag{3.3}$$

$$p_u(y) = C \sum_{i=1}^{n} k[(y - x_i)^2]\delta[b(x_i) - u] \tag{3.4}$$

where C is the normalizing constant, x_i are the pixel locations in the image, $k(x)$ is the gaussian kernel profile, $b(x)$ is the histogram bin index at pixel x, and u is one of the m histogram bins.

Therefore, given the object model $\mathbf{q}$ and its initial location y_0, the summary of the algorithm is given below:

1. Initialize the location of the target in the current frame with y_0 and evaluate $\{p_u(y_0)\}_{u=1...m}$ and

$$\rho[p(y_0), q] = \sum_{u=1}^{m} \sqrt{p_u(y_0) q_u}$$

2. Determine the weights $\{w_i\}_{i=1...n}$ using

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{q_u}{p_u(y_0)}} \delta[b(x_i) - u]$$

3. Find the next location of the object using

$$y_1 = \frac{\sum_{i=1}^{n} x_i w_i g[(y_0 - x_i)^2]}{\sum_{i=1}^{n} w_i g[(y_0 - x_i)^2]}$$

where $g(x)$ is the derivative of the gaussian kernel profile.

4. Evaluate $\{p_u(y_1)\}_{u=1...m}$ and

$$\rho[p(y_1), q] = \sum_{u=1}^{m} \sqrt{p_u(y_1) q_u}$$

5. While $\rho[p(y_1), q] < \rho[p(y_0), q]$

Do $y_1 = \frac{y_0 + y_1}{2}$

Evaluate $\rho[p(y_1), q]$

6. If $|y_1 - y_0| < \epsilon$ Stop.

Otherwise, Set $y_0 = y_1$ and go to Step 2.

3.2 Sub-Event Detection

The sub-events are detected by a rule-based system that takes in the tracked trajectories of entities as input. Let $f(\mathbf{p}, t)$ represent a continuous video signal, indexed by spatial and temporal coordinates, respectively. By indexing on the discrete-time variable k I can *temporally* represent the video signal as the set $\{f[\mathbf{x}, k]\}$ for $1 \leq k \leq N$, where N is the temporal support of the video signal, and $\mathbf{x} = (x, y)$ denotes the spatial coordinate. Each entity is represented in terms of its label (person, object, hand, or head) and motion, e.g. $\{\mathsf{vehicle}_a, \ \mathbf{u}_a\}$, where $\mathbf{u}_a = \{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$ is the trajectory of $\mathsf{vehicle}_a$'s centroid. Here it is assumed that the lower-level tasks of object detection, classification and tracking have been performed for a stationary camera (as described in the previous section). It is important to note that since it is the *relative* concept of motion that I am interested in (e.g. where did agent_1 move to with respect to object_2?), two-dimensional projections of three-dimensional world trajectories are sufficient for event representation, barring some degenerate configurations.

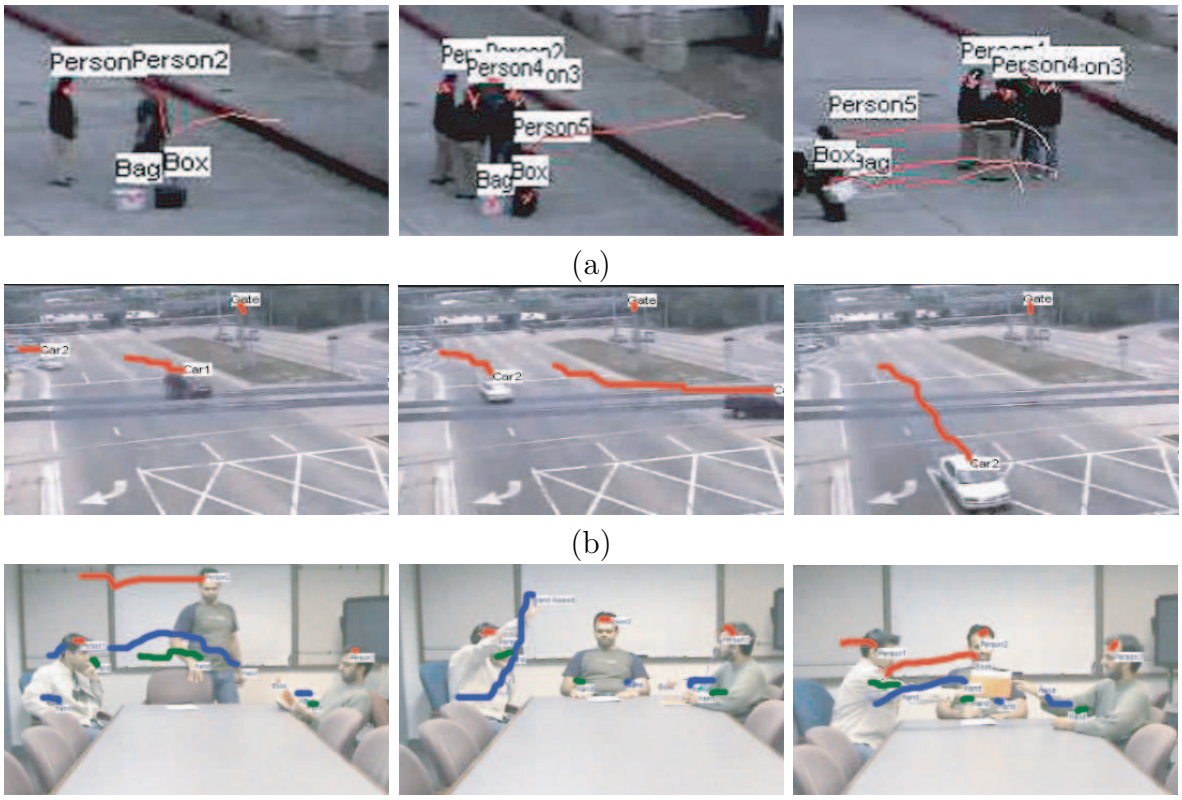
Using the tracked trajectories, the temporally correlated sub-events were detected in real-time and were further utilized for event learning (as described in the next chapter). A list of all unique sub-events for the surveillance, railroad monitoring and meeting domains are summarized in Table 3.1, and the details of the rule-based system for detecting each sub-event is provided in Appendix A.

Table 3.1: SUMMARY OF UNIQUE SUB-EVENTS

Function argument sets:
Agent={set of animates, e.g. person, vehicle etc.}
Object={set of non-animates, e.g. book, gate, railway signal, etc.}
Entity=Agent $\bigcup$ Object

Sub-event function definitions:

Moves(Entity),	function detecting movement of an entity
Stops(Entity),	function detecting seizure of movement of an entity
Enters(Agent),	function detecting entry of an agent in the field of view
Exits(Agent),	function detecting exiting of an agent from the field of view
Approaches(Agent,Entity),	function detecting movement of an agent towards an entity
Leaves(Agent,Entity),	function detecting movement of an agent away from an entity
Extends(Agent,{hand}),	function detecting movement of an agent's hand away from the body
Holds(Agent,Object),	function detecting proximal movement of an agent with an object
Picks(Agent,Object),	function detecting initial movement of an object with an agent
Passes(Agent,Agent,Object),	function detecting movement of object from one agent to another
Drops(Agent,Object),	function detecting seizure of object movement with agent movement
Raises({head},{hand}),	function detecting positioning of an agent's hand above head
Lowers({head},{hand}),	function detecting positioning of an agent's hand below head
Sits(Agent),	function detecting seizure of movement of an agent with downward motion
Stands(Agent),	function detecting initial movement of an agent with upward motion
Pushes(Agent,Agent),	function detecting quick/short movement of one agent away from the other
Blocks(Agent,Agent,Object),	function detecting occlusion of agent's view of an object by another agent
Crouches(Agent),	function detecting downward movement of an object
Hides(Agent,Object),	function detecting occlusion of an agent with an object
Emerges(Agent,Object),	function detecting reappearance of an agent from behind an object
Collides(Agent,Entity),	function detecting fast movement of an agent into an entity
Breaks(Agent,Entity),	function detecting collision of an agent with change in entity shape
Switches(Object),	function detecting change in object state i.e. signal switching on or off



(a)

(b)

(c)

Figure 3.1: Results of Meanshift tracking for videos in different domains. (a) Surveillance (b) Railroad Monitoring (c) Meetings.

3.3 Results

I performed three sets of experiments corresponding to each domain. All were implemented to run in real time (30 fps) on a 2.1 GHz Pentium Machine. The three domains in my experiments for tracking and detection of sub-events in videos were the meeting, railroad monitoring and surveillance domains. The videos contain multiple agents that act independently or interact with each other or objects. The results for the Meanshift tracking for various sequences in the three domains are shown in Figure 3.1.

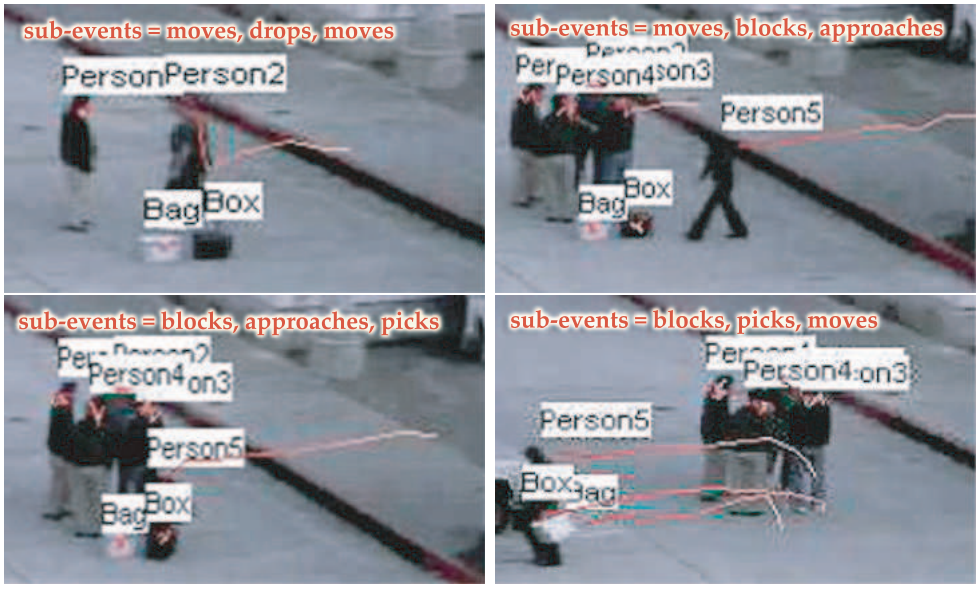


Figure 3.2: Automated detection of sub-events for stealing video. Using the tracked trajectories, the sub-events of each agent are detected, and frames 37, 119, 127, and 138 of the video are shown.

The 57 videos in my experiments totalled 40,977 frames having 2,673 sub-events. I used three standard video datasets, namely Performance Evaluation for Tracking and Surveillance (PETS), Context Aware Vision using Image-based Active Recognition (CAVIAR), and Video Analysis and Content Extraction (VACE), as well as my own videos for testing the tracking and sub-event detection rate. An example of the sub-event detection results for stealing event video is shown in Figure 3.2. Frame 37 shows the snapshot of *moves* sub-event being performed by Person1, while Person2 *drops* the bag and *moves* forward. Frame 119 shows the snapshot of Person4 *moves* in front of Person1 and *blocks* Person1's view of the bag, while Person5 *approaches* the bag. Frame 127 shows the snapshot of Person4 still blocking the view of Person1, and Person5 approaches and *picks* the bag. Frame 138 shows Person5

Table 3.2: SUMMARY OF SUB-EVENT DETECTION RATE

No. of Frames	Sub-Events Detected	Ground Truth	False Positive %	Precision %	Recall %
272	18	17	1	94.44	100
311	24	27	2	91.67	81.48
330	40	36	5	87.50	97.22
161	18	16	3	83.33	93.87
187	39	32	7	82.05	100
184	13	13	1	92.03	92.03
165	18	18	0	100	100
247	40	38	4	90	94.73
342	61	51	11	81.96	98.03
2000	102	108	7	93.13	87.96
335	32	29	5	84.37	93.10
402	34	28	7	79.41	96.42
237	9	10	1	88.89	80
223	4	4	0	100	100
486	12	9	3	75	100
192	9	8	1	88.89	100

moves away after picking the bag, while Person4 is still blocking the view of Person1. The sequence of these sub-events form the steal event.

The results for the sub-event detection for selected videos are shown in Table 3.2. The videos ranged from 161 to 2000 frames having 4 to 102 sub-events. As can be seen from the table, the precision ranges from 75% to 100% while recall ranges from 80% to 100%. Therefore, the rule-based system is able to detect the sub-events reasonably well, given the object labels and their trajectories. Although these precision and recall values are lower than the state-of-the-art in sub-event detection methods that use Hidden Markov Models, Dynamic Bayesian Networks, Stochastic Context Free Grammar, etc. which are in the

high 90s; our main contribution is in multi-agent event detection which requires sub-event detection as a pre-processing step. The point of using a method that obtains sub-optimal results is that our method of multi-agent event detection is not sensitive to the sub-event detection errors of insertion, deletion, and substitution errors, and is able to detect events fairly robustly (see next chapter for details).

3.4 Discussion

Object detection, tracking, and classification are usually a preprocessing step in a surveillance system. Once these tasks are performed a typical surveillance system detects sub-events of people or objects in the camera's field of view, generates computational models of the observed events using the detected sub-events, and generates alerts to a human supervisor when suspicious or abnormal event patterns are detected. Sub-events of the tracked objects are detected by a rule-based system, which are further utilized in the building of computational models to represent the observed event. Building computational models for events, however, is not an easy task due to the loss of one dimension when the 3D world points are projected onto a 2D camera image. In the next chapter, I describe my method of detecting events using the detected sub-events.

CHAPTER 4

EVENT LEARNING AND DETECTION

In the previous chapter, I used tracked trajectories to detect the temporally correlated sub-events in real-time. These sub-events are further utilized for learning the event structure and to detect events in novel video, which is the focus of this chapter. An *event* is a high-level concept that is composed of sub-events (actions) having a temporal order. There are several difficulties involved in event detection which are described below:

1. **Variation in Temporal Order of Sub-events**: The complexity of the event model increases with the increase in the variation in the temporal order of sub-events, since the event model captures the variation in each person's actions.

2. **Number of Agents**: Modeling the agent interactions becomes more complex with the increase in the number of agents involved in an event. This is true since the event model captures the variation in each person's actions. Also, most of the event models capture each agent's process and their interactions, thus, if the number of agents changes in the novel video, the event detection will fail. For example, if event model captured 2 agent interaction and the novel video has the same event being performed by 3 agents, the event will not be detected as the event model does not capture 3 agent interactions.

3. **Length of an Event**: The complexity of event model increases with the length of an event, since the event model captures the variation in each person's actions. Most of the methods assume a certain length of an event, so that the complexity the event model is reduced. But the event detection in these methods fail if the length of an event in the novel video exceeds their assumption.

4. **Interference in Event Sequence**: An event may not be detected in the novel video if the event sequence has actions performed by an independent agent. For example, if 2 agents are involved in a fight event while a 3rd agent is acting independently, then his actions will cause interference in the sub-event sequence of the fight event. This will result in the misdetection of the fight event, since the event model did not have the independent agent sub-events.

5. **Uncertainty in Sub-event detection**: The problem of event detection may be aggravated if the underlying vision system may have insertion, deletion, or substitution errors in the sub-event detection. An insertion error occurs when the vision system falsely detects the presence of a sub-event, also known as false positive. A deletion error occurs when the vision system misdetects the presence of a sub-event, also known as true negative. A substitution error occurs when the vision system misclassifies the detected sub-event.

In literature, a variety of approaches have been proposed for the detection of events in video sequences. Most of these approaches can be arranged into three categories based on their approach to event detection:

1. **Pre-defined Event Model**: These approaches detect events in videos based on pre-defined event models. Among these methods were approaches that modelled events using state machines. Further approaches utilized constraints and grammar rules for the detection of events that had variability, and did not strictly follow the event model. The various methods in this event detection approach include causal grammar, PNF propagation of temporal constraints, angular constraints on body components, stochastic grammars, and appearance and geometric constraints. All the above approaches either manually encode the event models or provide constraints such as grammar or rules, to detect events in novel videos.

2. **Learnt Event Model**: These approaches learn the event models using training data and have been widely used in the area of event detection. Among these methods were approaches that modelled events using temporal templates for event detection. Other approaches that utilized training data for learning the event model include Dynamic Bayesian Networks (DBNs), Hidden Markov Models (HMMs), coupled HMMs, dynamic multi-linked HMMs, abstract hidden Markov memory models, layered HMMs, Stochastic Context Free Grammar (SCFG), belief networks, shape activities, Support Vector Machines, Bayesian Networks, and probabilistic finite state machines. The above learning methods either model single person events or require prior knowledge

about the number of people involved in the events and variation in data may require complete re-training, so as to modify the model structure and parameters to accommodate those variations. Also, there is no straight-forward method of expanding the domain to other events, once training has been completed.

3. **Clustering based Methods**: These approaches that do not model the events but utilize clustering methods for event detection. These methods of event detection include spatio-temporal derivatives, co-embedding prototypes, and spatio-temporal curvatures of 2D trajectories. The first two methods find event segments by spectral graph partitioning of the weight matrix. The weight matrix is estimated by calculating a heuristic measure of similarity between video segments. The last method initiated without an event model and formed clusters of similar events based on their spatio-temporal curvatures. These methods were restricted to a single person and a single threaded event detection. The first two methods also assumed a maximum length of an event while the spatio-temporal curvature based event representation had the advantage of being view-invariant.

My approach to event detection is a hybrid of the learning and clustering based event detection. I use graph clustering for event detection in the novel videos, but instead of using a heuristic based similarity measure (used by above approaches) to construct the weight matrix, I utilize training data to learn the event models that are further used for event detection in novel videos. In my approach, *Event learning* is formulated as modelling conditional dependencies between sub-events while *event detection* is treated as a graph-

theoretic clustering problem. The novelty of my method is the ability to detect multiple agents performing multiple actions in the form of events, without prior knowledge about the number of agents involved in the interaction and without making any assumptions about the length of an event. I now describe each of these steps in detail in the following sections.

4.1 Learning the Event Model

In order to learn the event models from training videos, firstly, I introduce a graph that depicts the temporal relationships between sub-events in a video. These temporal relationships are based on the interval algebra in [AF94], which is a more descriptive model of relationships compared to the low level abstract relationship model of HMMs. The purpose of this graph is to encode the complete temporal order of sub-events occurring in a video. The temporal order of sub-events are further utilized in extracting the conditional dependency between sub-events. Thus, secondly, using this graph, I determine the *sub-event dependency graph* representing the temporal conditional dependency between sub-events. The sub-event dependency graph is the learnt event model that encodes the higher order Markov dependencies between sub-events and is scalable to the number of agents involved in the event.

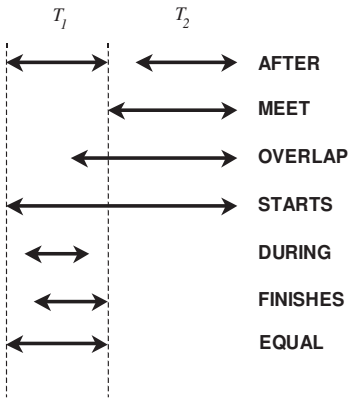


Figure 4.1: Allen's interval algebra describing temporal relations between durative sub-events T_1 and T_2.

4.1.1 Capturing Temporal Order of Sub-events using Allen's Temporal Algebra

Events are rarely instantaneous and are often defined by the temporal order of their sub-events. The temporal structure of events in a video can be intuitively represented as a *Directed Acyclic Graph* (DAG), with each vertex corresponding to a sub-event, and each edge corresponding to the temporal relationship between two vertices (e.g. **AFTER**). The graph is directed since there is a temporal order between sub-events and acyclic since time is monotonically increasing. More formally, my graph is a DAG, $G = (V, E)$ where $V = \{v_1, v_2, ..., v_n\}$; $v_i \in C$, and C is the set of n automatically measured sub-events; $E = \{e_1, e_2, ..., e_m\}$, where $e_i \in T$ and e_i are directed edges, and T is the set of temporal variables in the interval algebra of [AF94]. I use this algebra to represent seven temporal relationships[1] as shown in Figure 4.1. The entire list of temporal relations, for two sub-events T_1 and T_2 is as follows,

[1] A minor modification was made by replacing **BEFORE** with **AFTER** for ease of use

AFTER : $T_2^{start} > T_1^{end}$

MEETS : $T_1^{end} = T_2^{start}$

DURING : $(T_1^{start} < T_2^{start}) \wedge (T_1^{end} > T_2^{end})$

FINISHES : $(T_1^{end} = T_2^{end}) \wedge (T_1^{start} < T_2^{start})$

OVERLAPS : $(T_1^{start} < T_2^{start}) \wedge (T_1^{end} > T_2^{start}) \wedge (T_1^{end} < T_2^{end})$

EQUAL : $(T_1^{start} = T_2^{start}) \wedge (T_1^{end} = T_2^{end})$

STARTS : $(T_1^{start} = T_2^{start}) \wedge (T_1^{end} \neq T_2^{end})$

A naive formulation of the problem would be to consider a complete graph for estimating the conditional dependencies between sub-events. The problem with the complete graph formulation is that sub-events are not dependent on *all* their predecessor sub-events, rather they are dependent on their proximal predecessor sub-events. For example, a person *raising* a hand at the start of the video has nothing to do with *picking* a book sub-event, occurring after a few minutes have passed. Thus *transitive reduction* based upon proximity x is applied to the video event graph. This does not imply that I constrain my events to be a maximum of x length, rather it denotes that the events are composed of x-1th order Markov chain of sub-events. That is, each sub-event is conditionally dependent upon (at most) x-1 parent sub-events, which is true for most of the events in the considered domains. An example graph for a small sequence containing a voting event is shown in Figure 4.2.

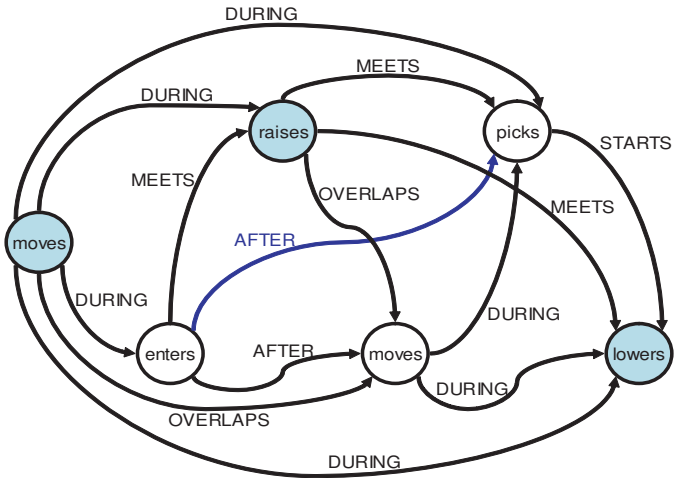


Figure 4.2: Partial graph for a sequence containing two agents performing actions simultaneously. The sub-events are represented by the vertices and the temporal relationships between sub-events are shown as directed edges between vertices. Agent1's sub-events are greyed while Agent2's are white to provide a visual distinction between their sub-events.

4.1.2 Event Modeling using Edge-Weighted Directed Hypergraph

Using the graph of a training video, I automatically learn the event model in the form of the Sub-event Dependency Graph (SDG). I model the events by estimating the conditional dependencies between unique sub-events that occur in a video. Thus, the SDG is the learnt event model that encodes the higher order Markov conditional dependencies between unique sub-events. The reason for estimating higher order Markov dependencies is that the sub-events are usually conditionally dependent upon more than one parent sub-events. The SDG is represented by an *Edge-Weighted Directed Hypergraph* (EWDH). More formally, an SDG is a hypergraph $G = (V, E, W)$ having a number of *vertices* $V = \{v_1, v_2, ..., v_n\}$ representing n *unique* sub-events present in an event. *Hyperarcs* $E = \{e_1, e_2, ..., e_m\}$ are backward arcs (*B-arcs*), where each B-arc is an *ordered* pair of vertices $e_i = (P_i, v_i)$ such that $P_i \subseteq V$, and P_i is

representing the temporally ordered parent sub-events of v_i. Also, $W = \{w_1, w_2, ..., w_m\}$ are the *weights* on the B-arcs, which represent the conditional dependencies of child sub-events upon a sequence of parent sub-events.

An ordinary graph is a 2-uniform hypergraph, where k-uniform signifies that each hyperedge has a *cardinality* of k vertices. I do not enforce a k-uniform hypergraph, rather I allow the hypergraph to have a maximum x edge cardinality (4 in my experiments). This allows the encoding of conditional probabilities of sub-events v_i, having a maximum of x-1 parent sub-events. The equations for estimating the weights w_i on hyperarcs e_i for cardinality of $X \in \{2, 3, 4\}$ are respectively given by (1), (2), and (3):

$$P(v_i^t|v_j^{t-1}) = \frac{P(v_i^t, v_j^{t-1})}{P(v_j^{t-1})} \tag{4.1}$$

$$P(v_i^t|v_j^{t-1}, v_k^{t-2}) = \frac{P(v_i^t, v_j^{t-1}, v_k^{t-2})}{P(v_j^{t-1}|v_k^{t-2})P(v_k^{t-2})} \tag{4.2}$$

$$P(v_i^t|v_j^{t-1}, v_k^{t-2}, v_l^{t-3}) = \frac{P(v_i^t, v_j^{t-1}, v_k^{t-2}, v_l^{t-3})}{P(v_j^{t-1}|v_k^{t-2}, v_l^{t-3})P(v_k^{t-2}, v_l^{t-3})} \tag{4.3}$$

where v_i^t represents a sub-event i occurring at index t, and $Agent(v_i^t) = Agent(v_a^b)$ $(a \in \{j, k, l\},\ b \in \{t\text{-}1, t\text{-}2, t\text{-}3\})$, which enforces the current and parent sub-events to be performed by the *same* agent. This is necessary since sub-events performed by different agents are not conditionally dependent on each other. If both the agents are involved in a sub-event, there is a conditional dependency between their sub-events. Equation (4.1) represents the conditional probability of sub-event v_i occurring at time index t, given that sub-event v_j occurred at t-1. Similarly, equation (4.2) represents the conditional probability of sub-event v_i occurring at t, given that sub-event v_j occurred at t-1, which was preceded by the sub-event v_k that occurred at t-2. An example of a partial SDG estimated from a video containing *voting* events is given

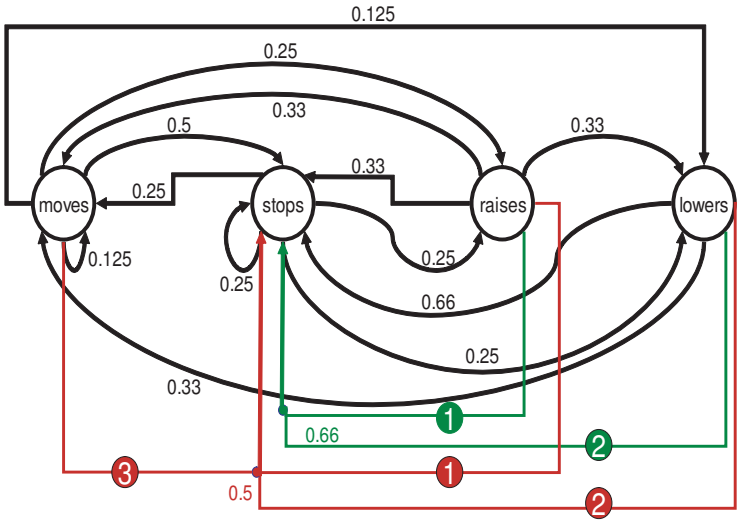


Figure 4.3: Partial sub-event dependency graph for a sample video containing multiple people performing *voting* events in various styles. The vertices represent the sub-events, while the conditional probabilities between sub-events are represented by the weights on the hyperarcs. Note that a single example of hyperarcs with cardinality of 3 and 4 are shown respectively in green and red, to keep the figure comprehendible. Also, the circled number on the hyperarc represents the temporal order index in P_i.

in Figure 4.3. In the figure, the B-arc of cardinality 4 represents P(stops|moves,lowers,raises) i.e. the probability of 'stop' occurring after a sequence of 'raises', 'lowers' and 'moves' sub-events. Note that the SDG captures all the variations in temporal order of sub-events as well as the conditional dependencies of all sub-events in a video. The SDG is also scalable to the number of agents involved in an event, as it will accumulate the conditional dependencies between sub-events, that are being performed by various agents.

4.2 Event Detection

After learning all the event models ξ_i in a supervised manner, *event detection* is defined as clustering the highly correlated chain of sub-events present in the novel video. Given the graph of detected sub-events in a novel video, I estimate a weight matrix of conditional dependencies between sub-events using the learnt event model. This weight matrix is used for spectral graph partitioning. Thus, *normalized cut* is applied recursively to this weight matrix in order to cluster the highly correlated sub-events. These clusters represent the detected events for a specific event model and the event structure composed of sub-events and their temporal order is extracted using graph partitioning. Furthermore, different weight matrices are estimated for each event model, and normalized cut is applied recursively, to extract all the events present in the novel video.

4.2.1 Estimating the Probabilistic Weight Matrix of Sub-event Dependencies

In order to determine the probabilistic weight matrix for a specific event model ξ_p, I generate a graph G of the detected sub-events in the novel video and obtain θ_p (edge weights representing conditional probabilities) from the learnt event model (SDG). Each edge weight w_{ij} between two nodes of G is estimated using:

$$w_{\alpha\beta} = P(v_l^t | Pa(v_l^t)) = P(v_l^t | v_k^{t-1}, v_j^{t-2}, v_i^{t-3})$$

	moves agent1	stops agent1	raises agent1	lowers agent1	raises agent2	moves agent2	lowers agent2	stops agent2	moves agent1	holds agent1	passes agent1,agent2	holds agent2	moves agent2
moves ag1	0	0.1826	0.1027	0.1598	0	0	0	0	0	0	0	0	0
stops ag1	0.1826	0	0.1141	0.0776	0	0	0	0	0	0	0	0	0
raises ag1	0.1027	0.1141	0	0.0717	0	0	0	0	0	0	0	0	0
lowers ag1	0.1598	0.0776	0.0717	0	0.013	0.1928	0	0.2029	0	0	0	0	0
raises ag2	0	0	0	0.013	0	0.877	0.437	0.2087	0.03	0	0.0361	0	0
moves ag2	0	0	0	0.1928	0.877	0	0.4337	0.1826	0.1027	0.1153	0.1923	0	0
lowers ag2	0	0	0	0	0.437	0.4337	0	0.2029	0.1014	0.0203	0.0203	0.0838	0
stops ag2	0	0	0	0.2029	0.2087	0.1826	0.2029	0	0.1141	0.0228	0.0228	0.0228	0.0833
moves ag1	0	0	0	0	0.03	0.1027	0.1014	0.1141	0	0.0457	0.0307	0.0755	0.0211
holds ag1	0	0	0	0	0	0.1153	0.0203	0.0228	0.0457	0	0.1904	0.1842	0.0263
passes ag1,ag2	0	0	0	0	0.0361	0.1923	0.0203	0.0228	0.0307	0.1904	0	0.221	0.0856
holds ag2	0	0	0	0	0	0	0.0838	0.0228	0.0755	0.1842	0.221	0	0.331
moves ag2	0	0	0	0	0	0	0	0.0833	0.0211	0.0263	0.0856	0.331	0

(a)

	moves agent1	stops agent1	raises agent1	lowers agent1	raises agent2	moves agent2	lowers agent2	stops agent2	moves agent1	holds agent1	passes agent1,agent2	holds agent2	moves agent2
moves ag1	0	0.0403	0.0092	0.0015	0	0	0	0	0	0	0	0	0
stops ag1	0.0403	0	0.044	0.0375	0	0	0	0	0	0	0	0	0
raises ag1	0.0092	0.044	0	0.0174	0.017	0.15	0.12	0	0	0	0	0	0
lowers ag1	0.0015	0.0375	0.0174	0	0.0563	0.002	0.012	0.0005	0	0	0	0	0
raises ag2	0	0	0.017	0.0563	0	0.266	0.132	0.087	0	0	0	0	0
moves ag2	0	0	0.15	0.002	0.266	0	0.274	0.126	0	0	0	0	0
lowers ag2	0	0	0.12	0.012	0.132	0.274	0	0.291	0.045	0.12	0	0	0
stops ag2	0	0	0	0.0005	0.087	0.126	0.291	0	0.2445	0.1322	0.017	0	0
moves ag1	0	0	0	0	0	0	0.045	0.2445	0	0.4766	0.371	0.325	0.299
holds ag1	0	0	0	0	0	0	0.12	0.1322	0.4766	0	0.364	0.285	0.2167
passes ag1,ag2	0	0	0	0	0	0	0	0.017	0.371	0.364	0	0.2201	0.195
holds ag2	0	0	0	0	0	0	0	0	0.325	0.285	0.2201	0	0.1333
moves ag2	0	0	0	0	0	0	0	0	0.299	0.2167	0.195	0.1333	0

(b)

Figure 4.4: The estimated *weight matrix* and *Ncut* application for a novel video. (a) The estimated weight matrix using the SDG of *voting* event, using *Normalized cut* two voting events are automatically segmented and are shown as red and blue patches. (b) The estimated weight matrix using the SDG of *object passing* event, using *Ncut* the object passing event is automatically segmented and is shown as the green patch.

where α is the index of sub-event v_l^t, and β is the index of $Pa(v_l^t)$ sub-event. $Pa(v_l^t)$ is the oldest parent sub-event that v_l^t conditionally depends upon such that $Pa(v_l^t) \in \{v_k^{t-1}, v_j^{t-2}, v_i^{t-3}\}$. Note that a sub-event may be dependent upon one or two parent sub-events, hence the conditional probabilities from hyperarcs of cardinality one and two respectively are inserted from the SDG to the weight matrix. Summarily, the above weight estimation assigns higher weights to the longer chain of sub-events that occur frequently in

the training video of ξ_p. The final weight matrix $\hat{W}_p$ is an upper triangle, since G is a directed acyclic graph. The weight matrix is made symmetric by $\tilde{W}_p = \hat{W}_p + \hat{W}_p^T$ [D04], where $\hat{W}_p^T$ is the transpose matrix of $\hat{W}_p$. The *Ncut* minimization function for weight matrices W_p and $\tilde{W}_p$ are equivalent, and the proof is given in Appendix B.

4.2.2 Graph Clustering using Normalized Cuts

Normalized cut [SM00] is an unbiased method of partitioning a graph V, into two (or more) segments A and B, since it uses a global criterion for graph segmentation rather than focusing on the local features. The global criterion is given by:

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)}$$

where $cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$, $w(u, v)$ is the edge weight between vertices u and v, and $asso(A, V) = \sum_{u \in A, v \in V} w(u, v)$. If the *Ncut* criterion is minimized, then the graph is partitioned at the edges with the minimum cut weight. The two partitions have maximum association within, and minimum disassociation between their respective partitions. The minimization of the *Ncut* criterion is achieved by finding the second smallest eigenvector of the generalized eigensystem:

$$(D - W)x = \lambda Dx$$

where D is a $N \times N$ diagonal matrix with $d(i) = \sum_j w(i,j)$ as the diagonal elements, W is a $N \times N$ symmetric weight matrix, and λ and x are the eigenvalues and eigenvectors, respectively.

4.2.2.1 Algorithm

The sub-event clustering algorithm using normalized cuts is summarized below:

1. Compute the weight matrix W and estimate the diagonal matrix D.

2. Solve $(D - W)x = \lambda Dx$ to obtain the eigenvector with the second smallest eigenvalue and use it to bipartition the graph by finding the splitting point such that the *Ncut* is minimized.

3. Decide if the current partition should be subdivided by checking that *Ncut* and *average edge weight* (that determines the association within a partition) are below their respective thresholds and then recursively repartition the segmented parts (if necessary).

The sub-event clusters determined by normalized cuts are the maximally correlated sub-events, given the likelihood estimates of the chain of sub-events. These segmented events have high weights between sub-events within the cluster and relatively low weights between sub-events outside the cluster. An example *weight matrix*, estimated using the SDGs of voting and object passing events and their segmentation obtained after recursive application of *Ncut*, is shown in Figure 4.4.

Table 4.1: SUMMARY OF TRAINING VIDEOS

Event Name	Total Frames	Sub-Events	Events
Voting	2938	221	26
Argument	913	82	7
Object Passing	532	70	4
Stealing	1386	129	4
Chasing	680	55	3
Fighting	2492	137	4
Object Exchange	1805	94	3
Object Drop	4484	81	4
Loading	761	62	3
Unloading	1485	38	6
Sneaking	2259	77	3
Railroad Event1	2731	199	17
Railroad Event2	2314	85	6
Railroad Event3	1228	44	4
Railroad Event4	1577	131	10
Railroad Event5	1745	93	4

Railroad Domain Events
Event1: Vehicle Stops outside danger zone
Event2: Vehicle entering zone while gate arms are in motion
Event3: Vehicle exiting zone while gate arms are in motion
Event4: Person enters danger zone while signal is on
Event5: Train crossing while gate arms are in motion

4.3 Results

I performed experiments for event detection in videos for the meeting, railroad monitoring and surveillance domains. These videos contain multiple agents that act independently or interact with each other or objects. In my experiments, the videos in all domains totalled 194,519 frames comprising of 11,540 sub-events and 1013 events. I used seven standard video

datasets as well as other videos for training and testing the event detection framework. A total number of 494 videos were adopted for training 16 events. The summary of event learning using different datasets is provided in Table 4.2. The two sections in the table show the standard dataset and my dataset description respectively.

Table 4.2: SUMMARY OF DIFFERENT DATASETS USED FOR EVENT LEARNING

Dataset Name	Videos	Total Frames	Events	Sub-Events
NIST	6	2480	11	214
Kojima	20	2406	29	264
Sadiye	10	6461	13	104
VACE	40	18724	118	1296
PETS	143	45430	343	2040
CAVIAR	33	28692	91	1507
ETISEO	32	33184	131	1870
Alexei	12	1894	12	48
FDOT	85	14271	98	1524
Meeting	37	4383	37	373
Surveillance	30	15352	35	673
Railroad	46	9595	51	552

The voting, argument and object passing events are from the meeting domain. The stealing, chasing, fighting, object exchange, object drop, loading, unloading, and sneaking events are from the surveillance domain. The vehicle stops outside danger-zone, vehicle entering danger-zone while gate arms are in motion, vehicle exiting danger-zone while gate arms are in motion, person enters danger-zone while signal is on, and train crossing while gate arms are in motion are from the railroad monitoring domain. The number of event

instances necessary for training the event model depends upon the variation in the temporal order of sub-events present in each event.

Using the learnt event models, event detection in novel video proceeded by estimating the weight matrices for each event model. Furthermore, normalized cuts are applied to obtain event clusters in the novel video. The results for event detection using normalized cuts are respectively summarized in Figures 4.5, 4.6, 4.7, and 4.8 for the meeting, surveillance and railroad monitoring domains. Figure 4.5(a) and (d) show snapshots of the voting event in progress, Figure 4.5(b) depict the object passing event, while Figure 4.5(c) shows the argument event. Figure 4.6(a) shows the snapshot of stealing event, Figure 4.6(b) shows the fighting event, Figure 4.6(c) depict the chasing event, while Figure 4.6(d) portray the object exchange event under progress. Figure 4.7(a) shows the loading event in progress, Figure 4.7(b) depict the object drop event, Figure 4.7(c) shows the snapshot of sneaking event, while Figure 4.7(d) portray the stealing event. Figure 4.8(a) show the vehicle entering danger zone event, Figure 4.8(b) depict train entering the zone, Figure 4.8(c) portray person entering danger zone event, while Figure 4.8(d) show the snapshot of vehicle exiting the danger zone event.

The precision and recall values for test videos are estimated using $Precision = \frac{\sum_{i,j} \psi(tde_i^j)}{\sum_{i,j} \psi(de_i^j)}$ and $Recall = \frac{\sum_{i,j} \psi(tde_i^j)}{\sum_{i,j} \psi(te_i^j)}$ respectively, where $\psi(tde_i^j)$ is the *true detected sub-events*, $\psi(de_i^j)$ is the *detected sub-events*, and $\psi(te_i^j)$ is the *true sub-events*, belonging to the i^{th} cluster of the j^{th} event. A summary of event detection results with precision and recall values is supplied in Table 4.3. As can be seen from the table, the precision and recall values drop significantly

across different test videos. The reason for such a drop is that the experiments were setup to test the robustness and stability of the event detection method for increasing complexity in multiple agent events in the videos. Thus, the meeting test video had the simplest events with minimal interaction between multiple agents. Surveillance video1 had slightly more complex events with multiple agent interaction. Surveillance video2 had even more complex events with heavy multiple agent interactions. Finally, railroad monitoring test video had the most complex events with simultaneous multiple agent and independent agent events. Thus, the precision and recall values dropped about 15% for this video compared to the meeting test video. But the overall event detection results are fairly decent and none of the current methods in literature (to the best of my knowledge) tackle such complex events with such high precision and recall values..

Table 4.3: RESULTS OF TESTING VIDEOS

Test Video	Frames	Events	Sub-Events	Precision %	Recall %
Meeting	1551	15	224	92.8	87.9
Surveillance Video1	3580	13	335	92.5	88.9
Surveillance Video2	4256	12	209	77.5	88.5
Railroad Monitoring	2260	9	307	85.6	79.8

4.4 Discussion

In order to learn the events from training videos, firstly, I introduce a graph that depicts the temporal relationships between sub-events in a video. These temporal relationships are

based on the interval algebra in [AF94], which is a more descriptive model of relationships compared to the low level abstract relationship model of HMMs. The purpose of this graph is to encode the complete temporal order of sub-events occurring in a video. The temporal order of sub-events are further utilized in extracting the conditional dependency between sub-events. Secondly, using this graph, I determine the *sub-event dependency graph* representing the temporal conditional dependency between sub-events. The sub-event dependency graph is the learnt event model that encodes the higher order Markov dependencies between sub-events and is scalable to the number of agents involved in the event. The main advantages of my event model are:

1. The temporal relations are more descriptive relationships between sub-events compared to the low level abstract relationship models of HMMs, Dynamic Bayesian Networks etc.

2. The event model does not make any assumptions about the length of an event.

3. The event model is scalable to the number of agents involved in an event since it models the sub-event dependencies instead of the agent processes.

Event detection in novel videos proceeds by estimating a weight matrix of conditional dependencies between the detected sub-events. The weights on edges between sub-events are recovered using the learnt event model. This weight matrix is then used for spectral graph partitioning. Thus, *normalized cut* is applied recursively to this weight matrix, to cluster the highly correlated sub-events. These clusters represent the detected events for a specific

event model, and the event structure composed of sub-events and their temporal order is extracted using graph partitioning. Furthermore, different weight matrices are estimated for each event model, and normalized cut is applied recursively to extract all the events present in the novel video. The main advantages of my event detection scheme are:

1. The event detection does not make any assumptions about the length of an event since the graph clustering will cluster highly correlated events of any length.

2. The event detection is scalable to the number of agents involved in an event since the graph clustering will cluster highly correlated events involving any number of agents.

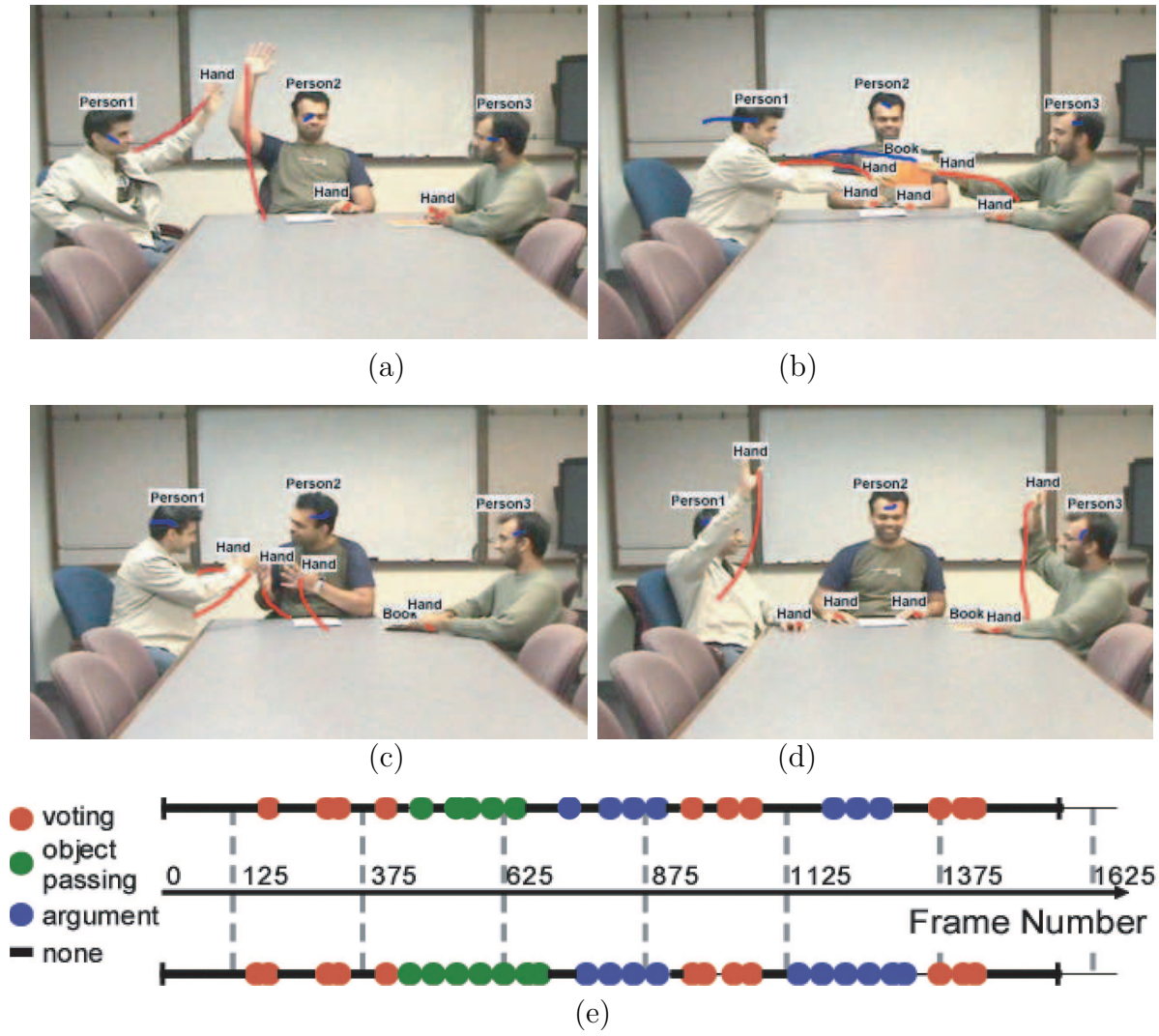


Figure 4.5: Event detection results using normalized cuts for meeting domain test video. (a)-(d) represent frames 328, 560, 755, and 1,375 respectively of meeting video consisting of 1,551 frames. (e) Time indexed clustering results for meeting video, where the top bar shows the actual event detection results and the bottom bar denotes the ground truth of the events.

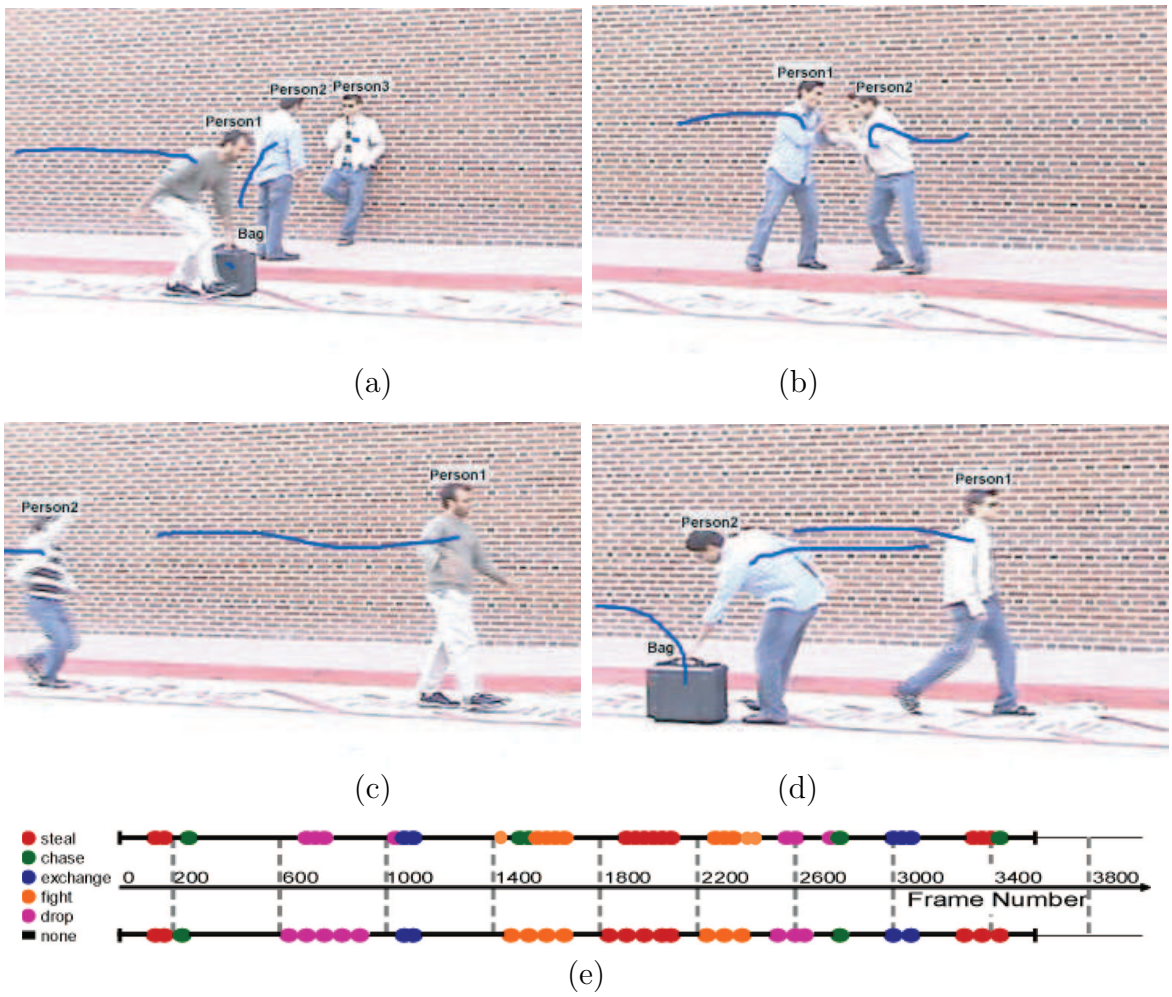


Figure 4.6: Event detection results using normalized cuts for surveillance domain test video1. (a)-(d) represent frames 159, 2,388, 2,874, and 3,125 respectively of surveillance video1 consisting of 3,580 frames. (e) Time indexed clustering results for surveillance video1, where the top bar shows the actual event detection results and the bottom bar denotes the ground truth of the events.

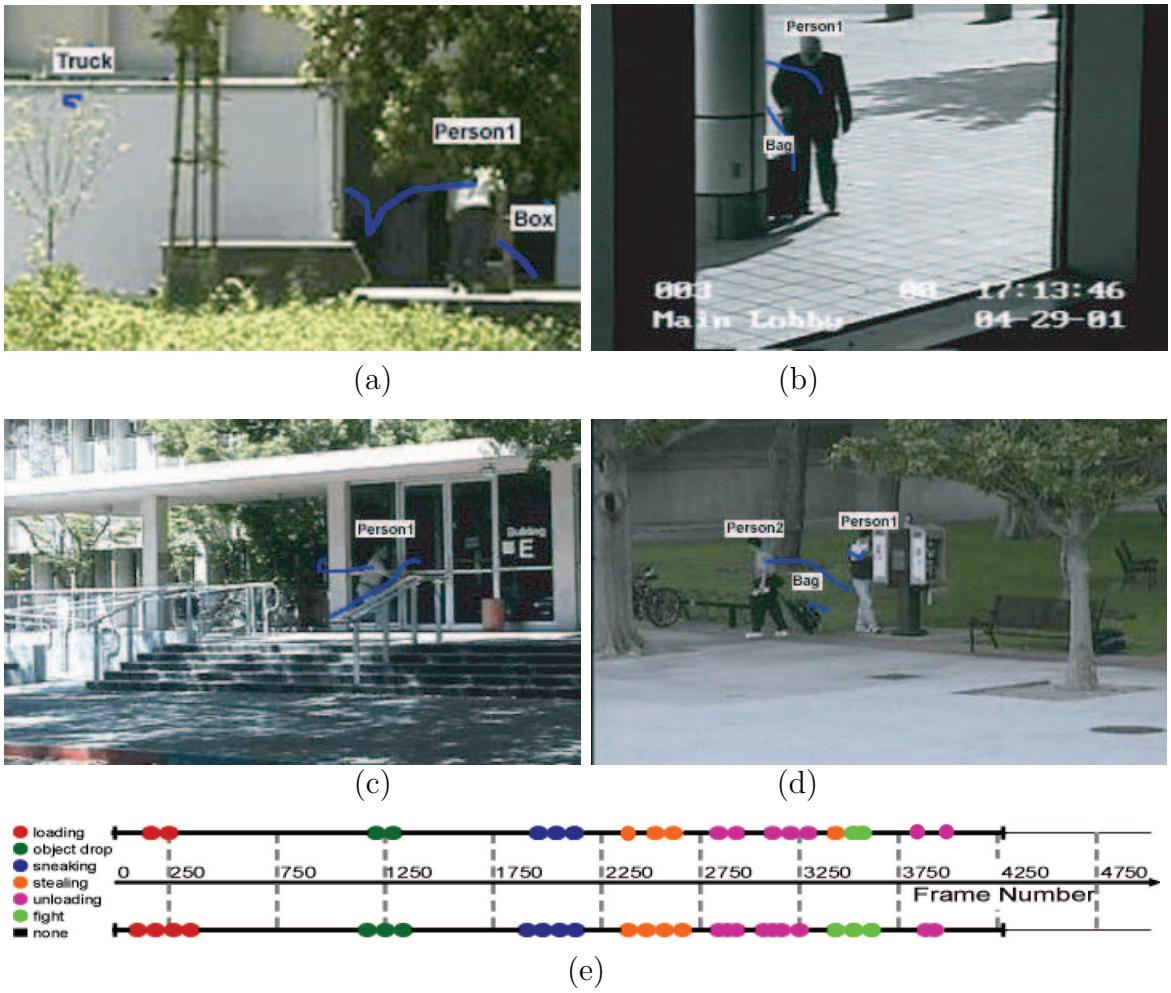


Figure 4.7: Event detection results using normalized cuts for surveillance domain test video2. (a)-(d) represent frame 223, 1,084, 2,191, and 2,703 respectively of surveillance video2 consisting of 4,256 frames. (e) Time indexed clustering results for surveillance video2.

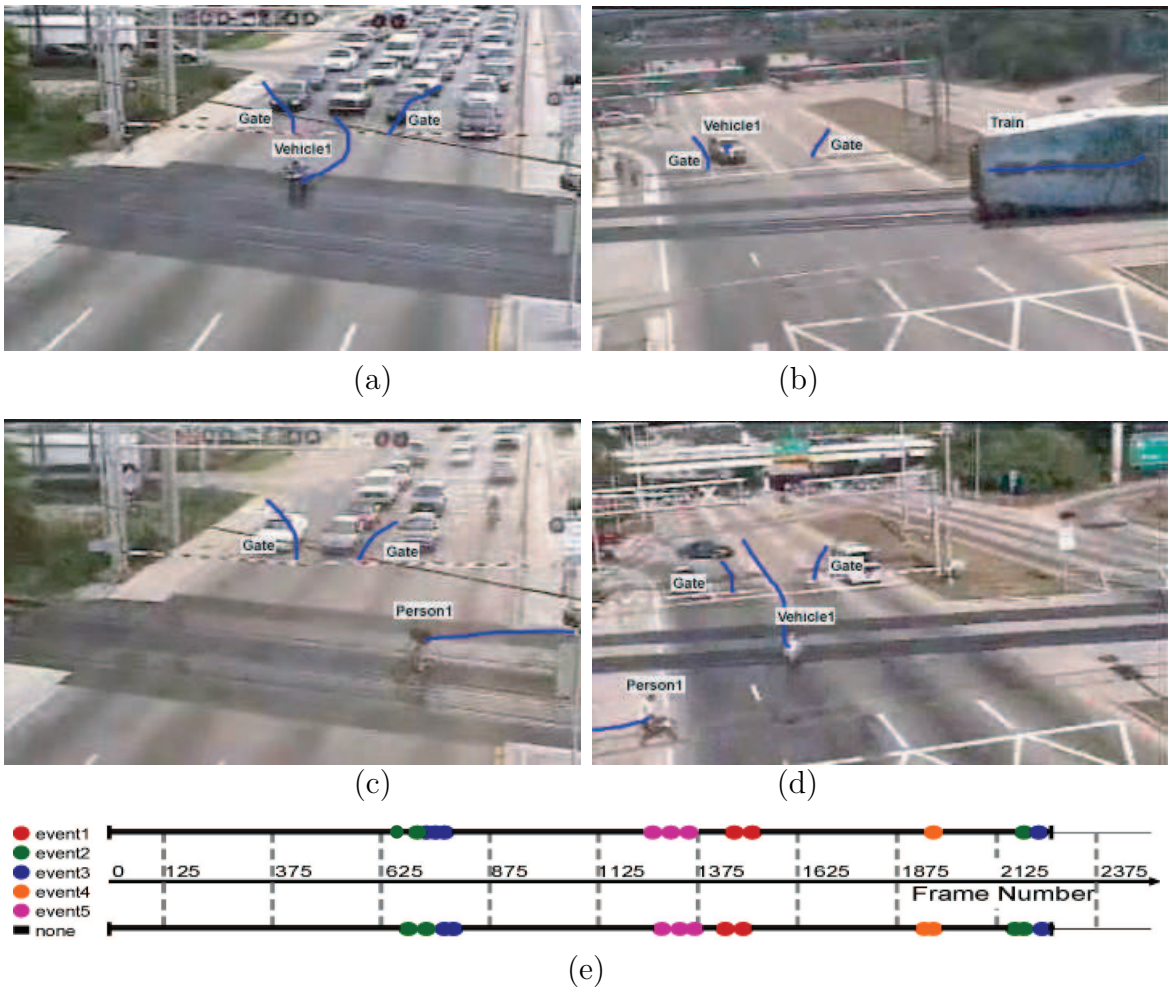


Figure 4.8: Event detection results using normalized cuts for railroad monitoring domain test video. (a)-(d) represent frames 740, 1,354, 1,966, and 2,251 respectively of railroad monitoring video consisting of 2,260 frames. (e) Time indexed clustering results for railroad monitoring video, where the top bar shows the actual event detection results and the bottom bar denotes the ground truth of the events.

CHAPTER 5

EVENT REPRESENTATION

The event clusters obtained through *Ncut*, in the form of temporally related sub-events, do not provide a sufficient interface between humans and computers. Although the event clusters represent the sub-events performed by various agents in an event, they do not provide details about the event. For example, the object exchange event is represented as a temporally related sequence of moves, holds, passes, holds and moves sub-events. This representation does not provide the details about what was being passed or which hand was the object being held. This chapter focuses on an extension of an existing natural language framework to an event representation that provides a plausible interface between humans and computers.

Let us first describe what are the desirable features of a good representation. A good event representation should have the following:

1. **Reconstructability**: Given an event representation, the user should be able to infer the complete details of the event i.e. who was involved in the event, how was the event performed (the order of sub-events), what objects (if any) were transferred during the event, etc.

2. **Scalability**: Given an event representation, the user should be able to scale the representation to other events in different domains.

3. **Modular**: Given an event representation, the user should be able to add more elements to the representation.

4. **Uniqueness**: Given an event, the generated representation should be unique i.e. no redundant elements should be present in the representation to generate a one-to-many mapping.

5. **Similarity**: Given an event representation, the user should be able to find similar instances using a distance metric.

6. **Completeness**: Given an event representation, the user should be able to define any type of event with the given representation constructs/elements.

In literature, there are several event representations that have certain advantages and limitations over other representations. These event representations are broadly categorized into two methods: representations that use first-order predicate logic and representations that utilize graphical schemas. Among the representations that use first-order predicate logic include CYC [CYC01] and VERL [NHB04]. CYC was an artificial intelligence project that started in 1984 with the goal of assembling a comprehensive common sense knowledge based ontology, enabling artificial intelligence applications to perform human-like reasoning. CYC knowledge base is built up of several million human-defined assertions, rules, facts, predicates

and common sense ideas. The common sense reasoning is based on first-order predicate logic with the language syntax similar to Lisp programming language. More recently, Nevatia et al. [NHB04] proposed a video event ontology by using their Video Event Representation Language (VERL). Their event representation is based on causal relationships based on conditionals and sequences of sub-events. The different event representation elements are defined using functions and linked via temporal logic encoded using Allen's temporal algebra [AF94]. Using their event representation, they are able to represent both single and multiple agent events. There are several disadvantages of first-order predicate logic based event representations. First, the complexity of the system in terms of concepts and predicates in the system and difficulty in adding to the system by hand. Second, scalability problems including reification of constants and difficulty in adding more concepts. Third, the related difficulty in measuring the completeness of the system, given the predicates and concepts.

Among the representations that utilized graphical schemas include hierarchical Event Representation Language (ERL) [NZH03], graph representation of object trajectories [MCB01], P-Net representation of human actions [SHMBE04], Factor Graph representation of semantics [NKHR00], and Bayesian Networks based event representation [HNB04] that provide varying degrees of representation to the actions and agents involved in the events. However, most of these works did not explicitly address important issues of temporal (except for P-Net) and causal relationships. Moreover, most of the event structures were encoded manually by observing the activities in a video and no mechanisms were proposed for human language based queries of multiple-agent or multi-threaded events.

The representation of my choice is called CASE that was proposed by Fillmore [F68] for natural language understanding. The basic unit of this representation is a case frame that has several elementary cases, e.g. agentive, instrumental and predicate. However, CASE was primarily used for syntactic analysis of natural languages, and while it provides a promising foundation for event representation it has several limitations for that end. I therefore propose an extended event representation that addresses the shortcomings of CASE. I also recognize the importance of representing the variations in the temporal order of sub-events, occurring in an event, and encode it directly into my representation, which I term P-CASE. These variations in the temporal order of sub-events occur due to the style of execution of events for different agents. Finally, I automatically learn the event structure from training videos and encode the *event ontology* using P-CASE. This has a significant advantage, since the domain experts need not go through the tedious task of determining the structure of events by browsing all the videos in the domain.

5.1 CASE Framework

CASE was proposed by Fillmore [F68] for natural language understanding and was primarily used for syntactic analysis of natural languages. The basic unit of this representation is a case frame that has several elementary cases. The description of the basic cases used by Fillmore is as follows, explained through an example:

Jack carefully advised Jim about the presentation in the meeting room using examples

1. **PRED**: predicate

   A function name or a label given to the case-frame i.e. *advised*

2. **AG**: agentive

   Main person or entity actively involved in a sub-event i.e. *Jack*

3. **I**: instrumental

   Device used by the agentive during the sub-event i.e. *examples*

4. **D**: dative

   The person or entity affected by the actions of the agentive i.e. *Jim*

5. **LOC**: locative

   The location of the sub-event i.e. *meeting room*

6. **OBJ**: objective

   The neutral acted upon by the agentive during the sub-event i.e. *presentation*

7. **FAC**: factative

   The state or action supporting the predicate i.e. $carefully$

The collection of all case-frames forms the CASE framework. The above example in case-frame notation is given below:

[**PRED**: advised, **AG**: Jack, **D**: Jim, **I**: examples, **OBJ**: presentation,

LOC: meeting room, **FAC**: carefully]

In my representation, the case-frames correspond to sub-events, and the collection of these sub-events form the event. While the CASE framework provides a promising foundation for event representation, it has several limitations to that end, which are discussed in the next section.

5.2 Extended CASE

In this section I discuss the four extensions to the CASE framework. Firstly, in order to capture both multi-agent and multi-thread events, I introduce a hierarchical CASE representation of events in terms of sub-events and case-lists. Secondly, since the temporal structure of events is critical to understanding and hence representing events, I introduce temporal logic into the CASE representation based on the interval algebra in [AF94]. Thirdly, I recognize the importance of causal relationships between sub-events and extend CASE to allow the representation of such causality between sub-events. Lastly, I recognize the importance of representing the variations in the temporal order of sub-events, that may occur in an event, and encode the probabilities directly into my event representation.

5.2.1 Hierarchical Representation

Except in constrained domains, events typically involve multiple agents engaged in several dependent or independent actions. Thus, any representation of events must be able to capture the composite nature of *real* events. To represent multiple objects, I introduce the idea of having case-lists of elements for a particular case. For example, if there are two or

more agents involved in an event, CASE cannot represent it in a case-frame as it only allows one agentive in the representation, therefore I add them in a case-list within **AG**,

[**PRED**: move, **AG**:{ person1, person2 }, ...]

To represent multiple threads I introduce the concept of sub-event lists (SUB case). An example is shown below to clarify the concept:

"Clyde robs the bank with the help of Bonnie who distracted the cashier by talking to him"

[**PRED**: rob, **AG**: Clyde, **OBJ**: Bank, **SUB**: {

[**PRED**: distract, **AG**: Bonnie, **D**: cashier],

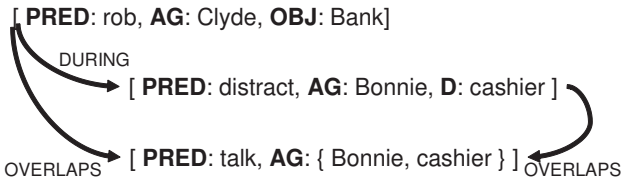
[**PRED**: talk, **AG**:{ Bonnie, cashier }] }]

In the above example, there are three sub-events occurring simultaneously, which cannot be represented in CASE representation as it only allows a single sub-event to occur at a particular time.

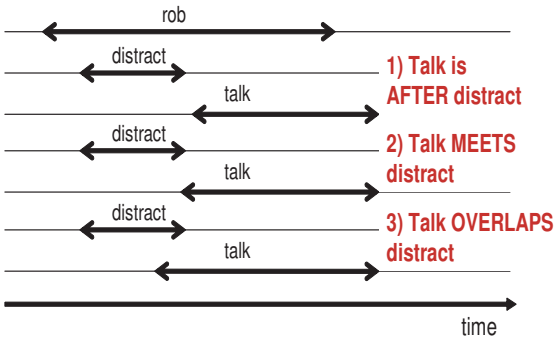
5.2.2 Temporal Logic

It should be immediately evident to the reader that the above event representation is ambiguous as temporal relations have not been defined. When did Bonnie talk to the cashier? Was it before, during or after the rob event? In order to have an unambiguous representation, I need to incorporate temporal relations in my representation.

Events are rarely instantaneous and often largely defined by the temporal order and relationships of their sub-events. In order to represent temporal relationships of sub-events, I introduce temporal logic into the CASE representation based on the interval algebra of [AF94]. Since temporal relationships exist between sub-events, they are represented on the directed edges between parent and child sub-events. Consider, once again, the above example of the steal event by Bonnie and Clyde. The case frames with the temporal logic incorporated are,



The above directed event graph representation is an extension of the tree structure of CASE^E [HSS04], where each directed edge represents a temporal parent-child relationship. The tree structure depicted a temporal relation with only its parents and not sibling sub-events (i.e. no OVERLAPS relationship between distract and talk sub-events) and thus had ambiguity in representing the complete order of sub-events. This is explained through the following example:



With sub-event 'distract' occurring DURING 'rob' sub-event, and 'talk' OVERLAPS 'rob' sub-event; there are three temporal possibilities between the 'distract' and 'talk' sub-events: 1) talk is AFTER distract, 2) talk MEETS distract, or 3) talk OVERLAPS distract. Thus, without the temporal relationship between distract and talk, there is ambiguity in temporal order of sub-events in the event representation. This ambiguity in the order of sub-events is resolved using the above graph representation.

5.2.3 Causality

In understanding the nature of events, the causal relationships between the constituent sub-events are indispensable. Some events might not occur if certain conditions were not satisfied, while some events may be dependent on other events. In order to explain this concept I show a simplistic example below,

"Caravaggio pulled the chair therefore Michelangelo fell down."

[**PRED**: pull, **AG**: Caravaggio, **OBJ**: chair, **CAUSE**:

[**PRED**: fell, **D**: Michelangelo, **FAC**: down]]

In the above example, Michelangelo would not have fallen down if Caravaggio had not pulled the chair. Therefore the 'fell' and 'pull' event have a causal relationship.

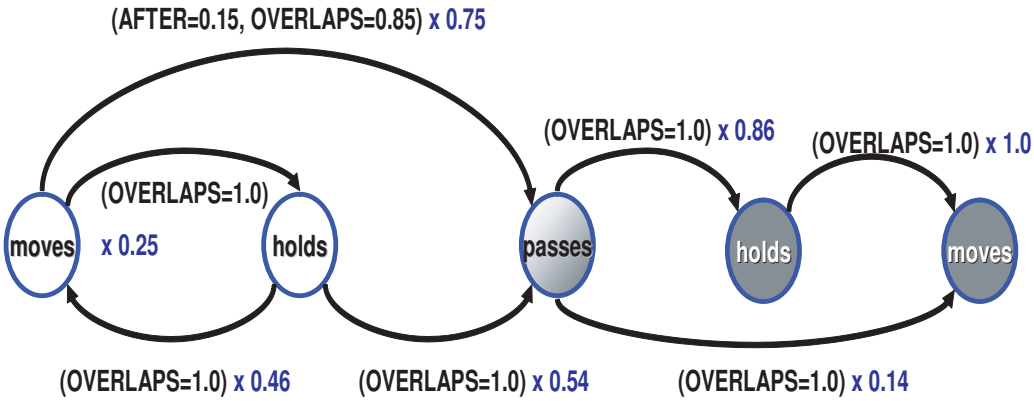


Figure 5.1: P-CASE representation for the *object passing* event. Each node is a sub-event encoded by a complete case-frame, and the weights on directed edges represent the probability of occurrence of a specific temporal relationship between sub-events, while the weights outside the brackets (in blue) are the conditional probabilities between sub-events. The white and grey vertices represent sub-events of agents one and two respectively.

5.2.4 Variation in Temporal Order

The above mentioned extensions to CASE provide a plausible interface for event representation. But events rarely occur with the same temporal order of sub-events. The variations in the temporal order of sub-events occur due to the different styles of execution of events by various agents present in the video. Thus, I modify the extended CASE representation to encode the temporal variations present in events.

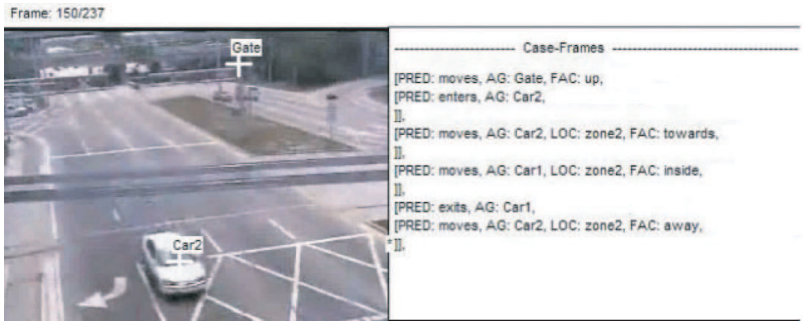
Given the detected sub-events and SDG for a training video, I estimate the probabilistic weight matrix using the procedure described in previous chapter. Each vertex in the weight matrix is encoded with a complete case-frame, instead of just the sub-event and agent information. Further application of normalized cuts to the weight matrix segments the different the event instances.

After obtaining the different event instances, the conditional dependencies between sub-events are estimated using equation (4.1) while the variation weights w_i^j in temporal relationships are computed using $w_i^j = \frac{\psi(T_i^j)}{\sum_k^n T_k^j}$, where w_i^j denotes the i^{th} weight for the j^{th} edge, $\psi(T_i^j)$ is the frequency of occurrence of the i^{th} temporal relationship for the j^{th} edge, and $\sum_k^n T_k$ is the normalizing factor representing all the n temporal relationships in the interval algebra for the j^{th} edge. The extended CASE representation is further modified by introducing these conditional dependencies and temporal variation weights w_i^j on directed edges of the segmented event graph. The final event representation is termed P-CASE, and an example of object passing event representation is shown in Fig.5.1.

One might argue that the SDG should be a sufficient event representation that inherently captures the variations in temporal order of sub-events. It should be noted that the SDG only encodes the conditional dependencies between ***unique*** sub-events. In contrast, P-CASE encodes *all* the sub-events with their temporal order that occur in an event. Also, it encodes more information, such as which agent performed what sub-event, that is lost while encoding the SDG. As shown in Figure 5.1, P-CASE encodes the variations in moves, holds, passes, holds and moves sub-events, while the SDG would only encode the conditional dependencies between moves, holds, and passes sub-events.

Furthermore, P-CASE has several advantages over existing event representations. First, it simultaneously uses both temporal structure and an environment descriptor to represent an event. Second, various events may have alternate starting sub-events, e.g. 'holds' may be before 'moves' in the given example. The ability to encode events with alternate starting

sub-events is yet another advantage that the previous representations lacked. Third, I find the most likely sequence of sub-events by finding the sequence with the maximum likelihood estimate (see details in results section). Finally, the representation is scalable to the number of agents involved in an event. A complete list of the most likely sequence of sub-events extracted from the P-CASE representation of the railroad monitoring domain is provided in the next section.



(a)

(b)

Figure 5.2: On-line extended CASE representation of video sequences. (a) Representation at frame 150/237 for the railroad monitoring video (b) PETS sequence representation at frame 1,446/2,000.

5.3 Results

I performed automatic annotation of events and sub-events in videos of different domains using my P-CASE representation. Firstly, I automatically generated case frames in real-time, corresponding to the detected sub-events. Figure 5.2 shows snapshots of individuals interacting in an unconstrained environment and their corresponding sub-event representations. Secondly, these temporally related sub-events were encoded in a graph and events were segmented using *Ncut* (as described above). These segmented events were further utilized for automated event annotation of the videos in the meeting, surveillance and railroad monitoring domains. Event-based retrieval of video is an interesting application of this video annotation scheme, and both the annotation and retrieval schemes are proposed in the next chapter. The automatically extracted event ontology using the P-CASE representation for the railroad monitoring domain is given in the next section.

5.3.1 P-CASE Representation for Railroad Monitoring Domain

Given the P-CASE representation of an event, I find the most likely sequence of sub-events by calculating the maximum likelihood estimate of all event instances using:

$$E_{ML}(e_i) = argmax_{(e_i)} P(e_i|E) \tag{5.1}$$

where, e_i are all the event instances belonging to the event E, where $P(e_i|E)$ is computed using:

$$\begin{aligned} P(e_i|E) &= P(s_1, s_2, ..., s_n|E) \\ &= P(s_1|E) \prod_{j=1}^{n-1} P(s_{j+1}, t_j^{j+1}|s_j, E) P(s_{j+1}|s_j, E) \end{aligned}$$

where, $s_1, ..., s_n$ are the sub-events belonging to event instance e_i, n are the number of sub-events in the event instance, and t_j^{j+1} is the temporal relationship between s_j and s_{j+1}. Thus, the likelihood estimate of the following event instance belonging to the object passing P-CASE (shown in Fig.5.1) is calculated by:

$$P(\text{holds} \xrightarrow{\text{Overlaps}} \text{moves} \xrightarrow{\text{After}} \text{passes} \xrightarrow{\text{Overlaps}} \text{holds} \xrightarrow{\text{Overlaps}} \text{moves} \mid \text{Object\_Passing})$$

$$= P(\text{holds})\ (P(\text{moves} \mid \text{holds})\ P(\text{moves,OVERLAPS} \mid \text{holds}))\ (P(\text{passes} \mid \text{moves})\ P(\text{passes,AFTER} \mid \text{moves}))\ (P(\text{holds} \mid \text{passes})\ P(\text{holds,OVERLAPS} \mid \text{passes}))\ (P(\text{moves} \mid \text{holds})\ P(\text{moves,OVERLAPS} \mid \text{holds}))$$

$$= 1.0 \times (1.0 \times 0.46) \times (0.15 \times 0.75) \times (1.0 \times 0.86) \times (1.0 \times 1.0) = \mathbf{0.0445}$$

The following is the most likely P-CASE representation for the 10 events in the railroad monitoring domain. Note that for ease of notation and visualization, the temporal relations are given inside the case-frame as was done in CASE^E [HSS04], instead of being on an edge between case-frames. Thus, I automatically estimate the most likely sequence of sub-events for all the events, and is the same representation that was derived manually by sifting through hours of videos in [HSS04].

Domain Entities

Vehicle	A vehicle in the universe
Person	A person in the universe
Train	A train on the tracks
Gate	Gate at the railroad crossing
Signal	Signal at the railroad crossing
Zone1	Zone covering the area of activation for the signal
Zone2	Zone covering a designated high-risk area
Tracks	The tracks that the train travels on

Domain Predicates *Moves*, *Enters*, *Exits*, *Switches*, *Signals*, *Breaks*, *Collides*, *Stops*.

Domain Events

1. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle stops outside Zone2**

   [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

   [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

   [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

   [**PRED**: Stops, **AG**: Vehicle, **LOC**: Zone2, **FAC**: Outside, **AFTER**: Moves]]]]

2. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle stops inside Zone2**

[**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

[**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

[**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

[**PRED**: Stops, **AG**: Vehicle, **LOC**: Zone2, **FAC**: Inside, **AFTER**: Moves]]]]

3. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle breaks the gate arm while entering Zone2**

   [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

   [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

   [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

   [**PRED**: Enters, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves, **SUB**:

   [**PRED**: Breaks, **AG**: Vehicle, **OBJ**: Gate, **DURING**: Enters]]]]]

4. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle breaks the gate arm while exiting Zone2**

   [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

   [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

   [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

   [**PRED**: Exits, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves, **SUB**:

   [**PRED**: Breaks, **AG**: Vehicle, **OBJ**: Gate, **DURING**: Exits]]]]]

5. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle enters while gate is in motion**

   [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

   [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

   [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

   [**PRED**: Enters, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves]]]]

6. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle exits while gate is in motion**

   [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

   [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

   [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

   [**PRED**: Exits, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves]]]]

7. **Vehicle collides with train**

   [**PRED**: Moves, **AG**: Train, **LOC**: Zone2, **FAC**: Inside, **SUB**:

   [**PRED**: Moves, **AG**: Vehicle, **FAC**: Inside, **LOC**: Zone2, **DURING**: Move, **SUB**:

   [**PRED**: Collides, **AG**: { Vehicle, Train }, **AFTER**: Moves]]]

8. **Person being hit by train**

   [**PRED**: Moves, **AG**: Train, **LOC**: Zone2, **FAC**: Inside, **SUB**:

   [**PRED**: Moves, **AG**: Person, **FAC**: Inside, **LOC**: Zone2, **DURING**: Move, **SUB**:

[**PRED**: Collides, **AG**: { Person, Train } ,**AFTER**: Moves]]]

9. **Person enters zone2 while signal was switched on**

   [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **SUB**:

   [**PRED**: Moves, **AG**: Person, **LOC**: Zone2, **FAC**: Towards, **OVERLAPS**: Switches, **SUB**:

   [**PRED**: Enters, **AG**: Person, **LOC**: Zone2, **AFTER**: Moves]]]

10. **Train entering zone2 while gates are in motion**

    [**PRED**: Moves, **OBJ**: Gates, **FAC**: Down, **SUB**:

    [**PRED**: Moves, **AG**: Train, **LOC**: Zone2, **FAC**: Towards, **OVERLAPS**: Moves, **SUB**:

    [**PRED**: Enters, **AG**: Train, **LOC**: Zone2, **DURING**: Moves]]

5.4 Discussion

In order to represent the events, I extend the CASE [F68] representation of the natural language. CASE was primarily used for syntactic analysis of natural languages, and while it provides a promising foundation for event representation it has several limitations for that end. I therefore propose four critical extensions to CASE for the representation of events:

1. Accommodating multiple agents and multiple threads in an event.

2. Supporting the inclusion of temporal information into the representation.

3. Supporting the inclusion of causal information into the representation.

4. Accommodating variation in the temporal order of sub-events.

I also proposed a novel event graph representation for the detected events in video sequences, having temporal relationships between sub-events. Hence, unlike almost all previous work, I use both temporal structure and an environment descriptor simultaneously to represent an event. I also recognize the importance of representing the variations in temporal order of sub-events, that occur in an event and encode it directly into my representation, which I term P-CASE. These variations in the temporal order of sub-events, occur due to the style of execution of events for different agents. It should also be noted that only definite causal relations are represented by the **CAUSE** case, instead of using temporal relationship. While the proposed extension allows the representation of causal relationships, it is noted that causal relationships cannot be inferred from video measurements alone. In other words, it is impossible to make a distinction between two successive events, and two causal events without some reasoning. Thus, from the point of view of on-line processing of measurements, videos are represented in terms of a *temporal representation*. Events and sub-events are arranged in a hierarchy according to the order of their temporal incidence and duration. Inferring causality solely from these temporal representations is a promising future direction. The next chapter describes the event indexing scheme that utilizes the P-CASE representation, which is further used for event based retrieval of videos.

CHAPTER 6

EVENT INDEXING AND RETRIEVAL

The semantic nature of event representation in P-CASE would allow two additional functionalities: (1) an accessible video retrieval system, indexed based on the semantics of events that occurred. In time-critical situations (e.g. after the London bombing, it took several days to sift through the video evidence), indexing videos in terms of events would allow users to browse the video database based on semantic precepts rather than sequentially piece through hours of raw video. (2) It would allow analysts to flag specify events of interest as they occur. The P-CASE framework reasons about events in terms of agents, actions, and objects allowing human input into the processes that occur within the system.

Although there is a plethora of literature devoted to content-based image retrieval evident from the survey by Rui *et al.* [RHC99], most of the work is based on features retrieved from a single image. Methods that utilize probabilistic image indexing and retrieval include Boreczky and Wilcox [BW97] that utilize audio and image features and Dimitrova *et al.* [DAW00] that use text and face features for indexing and retrieval. Non-probabilistic methods include the work by Katayama and Satoh [KS97] that employ an SR-tree to index the high dimensional feature vector and utilize nearest neighbor query search for retrieval of relevant data. Recently, works by Natarajan and Nevatia [NN05] utilize video information

in the feature vector for retrieval of similar videos from the database. Natarajan and Nevatia [NN05] in their EDF framework use an ontology of entities, actions and events to index the video events. Further, they utilize relational algebra to find complex events in the database.

Naphade and Huang [NH01] use Factor Graphs (FGs) to index the database. The factor graphs are dependency mapping functions that map low-level features to high-level concepts called multijects. The low-level features used include color, texture, edginess (edge regions), shape, and motion. While, multijects are probabilistic multimedia objects that belong to either of the three categories: objects (vehicle, person, etc.), sites (indoor, outdoor, beach, etc.), or events (explosion, walking, running, etc.). The dependencies between the multijects are modelled using the factor graphs where the dependencies are estimated using training data, while the factor graph structure is manually constructed. The event (concept) retrieval is achieved via inference using the sum-product algorithm. Though the above methods are promising for retrieval of video data, they lack the representative power to extend their abstract event model to representations related to human understanding of events.

Also, there are several content-based video retrieval tools that include IBM's Multimedia Analysis and Retrieval System (MARVEL) [MARVEL], University of Cental Florida's PEGASUS [PEGASUS], and Columbia University's VideoQ [CCMSZ97]. MARVEL allows searching over large video repository using automatically generated semantic labels. The MARVEL system consists of two components: the MARVEL multimedia analysis engine and the MARVEL multimedia search engine. The MARVEL multimedia analysis engine applies multi-modal machine learning techniques to model semantic concepts in video from

automatically extracted audio, speech, and visual content. It automatically assigns labels (with confidence scores) to new video data to reduce manual annotation load and improve searching and organizes semantic concepts using ontologies that exploit semantic relationships for improving detection performance. For example, if in a video clip, the system indicates the presence of sky, water, sand, and people then the confidence score for detecting beach are boosted. Furthermore, all of this boosting information are learned automatically by the system by extracting correlations and statistical information using training examples. The MARVEL multimedia retrieval engine fuses multimedia semantics-based searching with other search techniques (speech, text, meta-data, audio-visual features, etc.). It also combines content-based (low-level features), model-based (semantic concepts), and text-based (automatic speech recognition) searching for video searching.

PEGASUS utilizes text (Optical Character Recognition and Automatic Speech Recognition), image regions, word histogram and color histogram as the feature vector and indexes the videos using an SR-tree. The user queries are in the form of keywords with logical operators and the system initially returns the results based on text search. The user can further refine the search results by selecting relevant results and deselecting irrelevant results. The user can also utilize image region, word or color histogram based query refinement during the interactive search. The final query results are ranked based on the Earth Mover's distance.

VideoQ employs color, texture, shape, motion and time information as the feature vector for indexing the videos. Using VideoQ, the user can retrieve videos of objects that have similar color and motion trajectories. The user queries can be either of the two types: query

by example or query by sketch. The user supplies a video for 'query by example' and system returns similar videos from the database. The user sketches a trajectory for 'query by sketch' and the system returns videos with similar motion trajectories.

Furthermore, there are a few event representations that attempt to utilize their representations for retrieval of similar events and concepts. Among these representations are those that use first-order predicate logic for inference and retrieval and include CYC [CYC01] and VERL [NHB04]. CYC was an artificial intelligence project that started in 1984 with the goal of assembling a comprehensive common sense knowledge based ontology, enabling artificial intelligence applications to perform human-like reasoning. CYC knowledge base is built up of several million human-defined assertions, rules, facts, predicates and common sense ideas. The common sense reasoning is based on first-order predicate logic with the language syntax similar to Lisp programming language. The summary of the CYC elements are as follows:

1. **Constants**: The concept name in CYC are termed constant.

2. **Individuals**: Individual items such as Paris, Bill Gate etc.

3. **Collections**: Collective items such as trees which contains all types of individual tree elements. A member of the collection is termed an instance of that collection. E.g. an oak tree is an instance of the tree collection.

4. **Truth Functions**: Functions that can applied to one or more concepts and return a true or false value.

5. **Functions**: Functions that produce new terms from the given ones using inference. E.g. fruit-function when provided with a plant type collection will return a collection of fruits produced by those plants.

6. **Predicates**: Special functions in CYC.

7. **Isa**: Predicate that describes that one element is an instance of another. E.g. George-Bush isa President-of-US.

8. **Generalize**: Predicate that describes that one collection is a sub-collection of another. E.g. Generalize Oak Tree (i.e. oak is a sub-collection of tree).

More recently, Nevatia et al. [NHB04] proposed a video event ontology by using their Video Event Representation Language (VERL). Their event representation is based on causal relationships based on conditionals and sequences of sub-events. The different event representation elements are defined using functions and linked via temporal logic encoded using Allen's temporal algebra [AF94]. Using their event representation, they are able to represent both single and multiple agent events. The summary of the VERL elements are as follows:

1. **Process**: Tightly coupled actions with necessary causality e.g. washing your car.

2. **Activity**: Loosely coupled actions with causality not being necessary e.g. my washing my car every Saturday last year.

3. **Cause**: Causal relation producing a change.

4. **Change**: Effect of cause in properties or actions.

5. **Sequence**: Two sub-events occurring one after the other.

6. **Iteration**: Repeating of events 'n' times.

7. **Alteration**: Choose between alternating events (OR relationship).

8. **Conditionals**: Actions occurring based on conditions.

9. **Interruptions/Resumptions**: Pause/resume of a process.

10. **Fork/Join**: Parallel/sequential execution of a process.

There are several disadvantages of first-order predicate logic based event representations. First, the complexity of the system in terms of concepts and predicates in the system and difficulty in adding to the system by hand. Second, scalability problems including reification of constants and difficulty in adding more concepts. Third, the related difficulty in measuring the completeness of the system, given the predicates and concepts. Fourth, these representations are not directly related to the human understanding of events, and thus, the user cannot define the query in a human representative language.

Keeping the limitations of the above tools, representations, and methods I built the Semoran system for the indexing and retrieval of events in videos, which is described next.

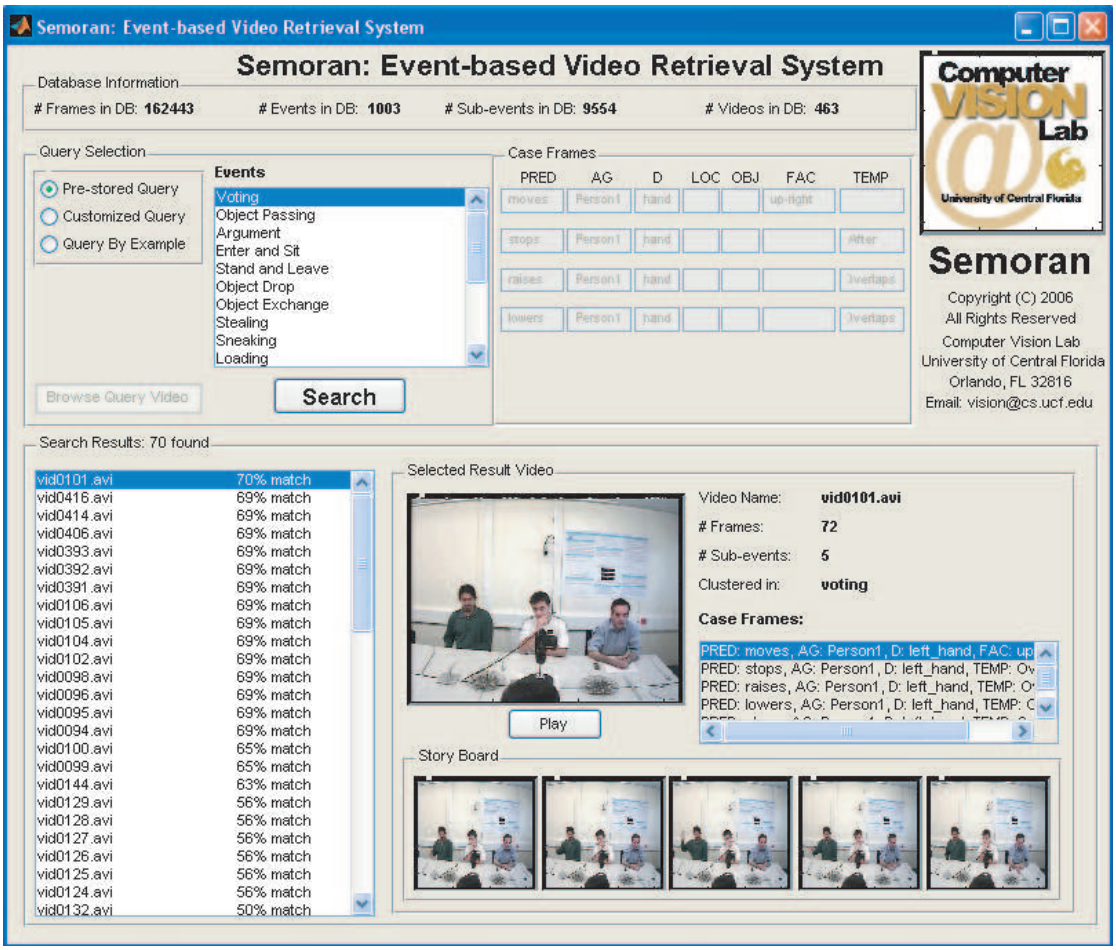


Figure 6.1: Event retrieval using pre-defined queries.

6.1 Semoran System

The goal of the Semoran system is to have a user friendly event retrieval system. It takes in user inputs in either of the three forms for event retrieval:

1: Using Pre-Defined Queries in P-CASE Representation

The user can select a particular event and the system will return the relevant events from the database as shown in Figure 6.1.

Figure 6.2: Event retrieval using custom queries.

2: Using Custom Queries in P-CASE Representation

The user can create a custom event by filling in the cases and the system will return the relevant events from the database as shown in Figure 6.2.

3: Using Query by Example Video

The user can supply a query in the form of a video. The video has the tracked trajectories of the different objects stored in a file. The system reads them and generates the case-

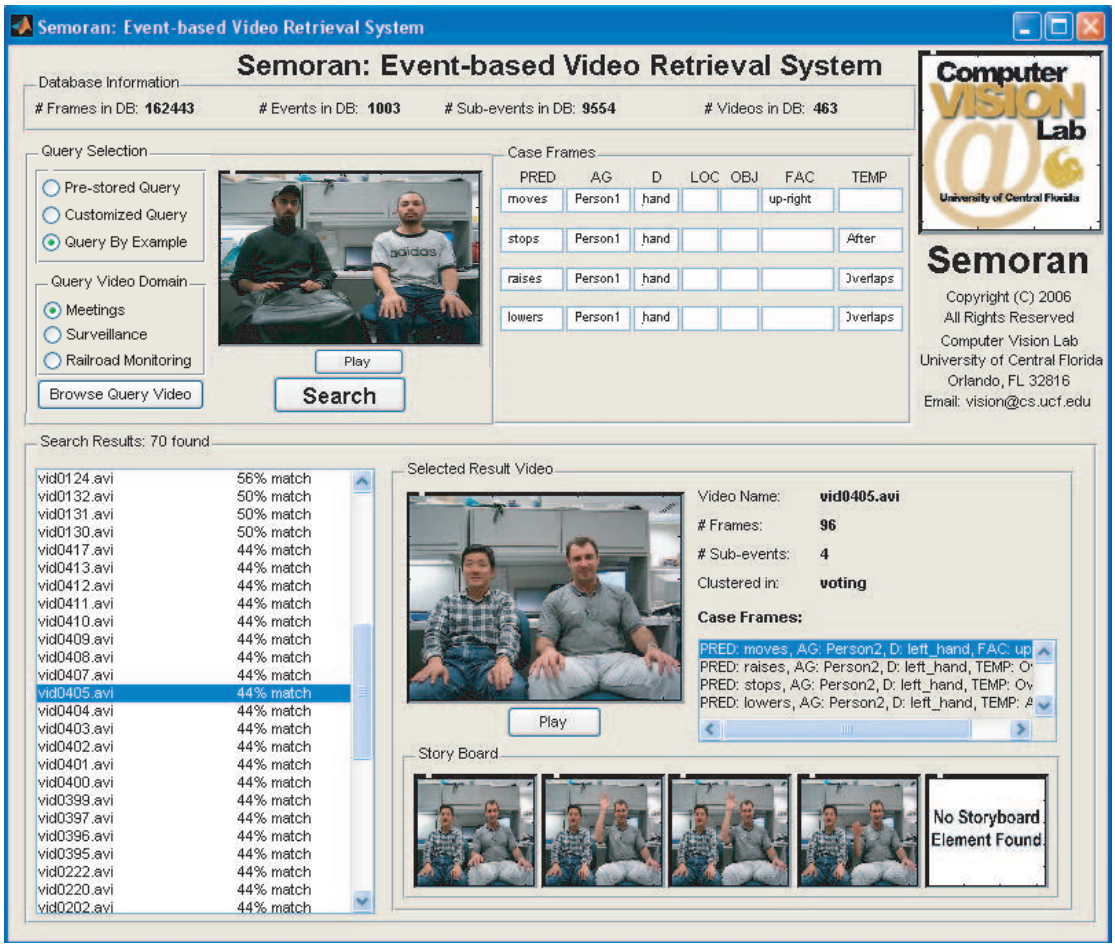


Figure 6.3: Event retrieval using query by example video.

frames, which forms the query and relevant events from the database are returned as shown in Figure6.3.

Once the query is constructed, the system then searches the entire database and returns the matched videos to the user. The user can then play the returned videos, look at the actions performed by various agents in the storyboard, and visualize the actions in P-CASE representation. Internally, the system performs a two-level search. At the first level, the system matches the query with the different event models indexed in the database, using

maximum likelihood estimates. Once it finds a match, at the second level, it performs a thorough search with all the relevant events belonging to the matched model, using weighted Jaccard similarity. The details of event indexing and retrieval follow.

6.2 Event Indexing

Given the different instances of a particular event extracted using normalized cut I can index the events in P-CASE representation using the method described in section 5.2.4. Note that this indexing scheme preserves the order of predicate occurrence, which was a limitation of all predecessor indexing methods. I next compare P-CASE with well known event models and end the section by listing the advantages of P-CASE over existing methods.

6.2.1 Comparison of P-CASE with HMMs

Hidden Markov Models (HMMs) are the most widely used event models for activity recognition. I provide its strengths and weaknesses compared to P-CASE as follows:

1. P-CASE is a static graph where the number of nodes are pre-determined based on the high-level concepts i.e. sub-events in an event, while HMM is a non-static graph where the number of hidden nodes are unknown and are estimated based on training data.

2. P-CASE has each node representing a high-level concept while HMM's node represents a hidden variable that has no high-level concept associated with it.

3. P-CASE can model/represent events having *any* number of agents while HMM can model *single* agent events, where as its variant Coupled HMM can model two agent interactive events, but the number of agents has to be known as a prior.

4. P-CASE has temporal relations on arcs that uniquely represents the complete order of sub-events, while HMM can have self-loops on nodes but it cannot count how many times to loop, thus there is ambiguity in the order and duration of sub-events.

5. Both P-CASE and HMMs derive the model structure and parameters automatically using training examples but HMMs have a more complex and abstract model structure with unknown number of nodes, while P-CASE is more human readable structure with known nodes (equal to the number of sub-events in events).

6. Both P-CASE and HMMs can be used to detect events by finding the maximum likelihood estimates with all the event models, given a novel video (see the next section for details).

7. P-CASE builds models by assuming a closed world problem i.e. it requires knowledge of all the sub-events as a prior, while HMM does not require this and it builds its model structure independent of that knowledge by adding more hidden nodes to fit the model, given the training videos.

6.2.2 Comparison of P-CASE with Bayesian Networks

Bayesian Networks (BNs) are the used for modeling multiple agent events for activity recognition. I provide its strengths and weaknesses compared to P-CASE as follows:

1. Both P-CASE and BN are static graphs.

2. P-CASE is a directed graph while BNs are Directed Acyclic Graphs (DAGs) therefore BN inference (or likelihood estimation) is bottom up, using the marginal probabilities, as there are no cycles while P-CASE estimates maximum likelihood by function maximization (see the next section for details).

3. Both P-CASE and BNs have nodes representing high-level concept.

4. P-CASE has temporal relations on arcs that uniquely represents the complete order of sub-events, while BNs have conditional probabilities on their arcs which just encodes that a certain node occurs after another node. Thus, P-CASE has more temporal information encoded.

5. P-CASE derive the model structure and parameters automatically using training examples but BN structure are usually more complex and is usually derived manually while the parameters are derived automatically.

6. Both P-CASE and BN can model/represent events having *any* number of agents.

Both HMMs and BNs lack the representative power to extend their abstract event model to representations related to human understanding of events. Also, they lack the ability to uniquely represent the complete order of sub-events, which is necessary for indexing of events for future retrieval.

6.3 Event Retrieval

After indexing the events using the P-CASE representation, event retrieval is a two step process. Given a query in the P-CASE representation, I first find the Maximum Likelihood (ML) estimate of the query event with the different event models. The model that has the ML estimate of the query event is the maximum matching event. The ML is calculated using:

$$E_{ML}(q) = argmax_{E_i}\ P(q|E_i)$$

where, E_i is the event, q is the query, $P(q|E_i)$ is the likelihood term calculated using:

$$\begin{aligned} P(q|E_i) &= P(s_1, s_2, ..., s_n|E_i) \\ &= P(s_1|E_i) \prod_{j=1}^{n-1} P(s_{j+1}, t_j^{j+1}|s_j, E_i) P(s_{j+1}|s_j, E_i) \end{aligned}$$

where, $s_1, ..., s_n$ are the sub-events belonging to query event q, n are the number of sub-events in the query event, and t_j^{j+1} is the temporal relationship between s_j and s_{j+1}. Note that the user does not need to supply all the *cases* inside each case-frame used in the search query.

E.g. instead of searching for events in which "Omar passes a yellow book", the user can search for all events in which "anyone passes any book". This is achieved by enumerating all possible agentives (persons) and datives (books) in the search query and matching it to all stored events in the database.

At the second level of search, I find the percentage match of the query event with all the event instances belonging to the maximum matched event model. For that purpose I use the weighted Jaccard measure to find similarity scores between the query event and all event instances. Two case-frames C_i and C_j are matched using the weighted Jaccard measure given by:

$$\rho(C_i, C_j) = \frac{\sum_{k=1}^{n} w_k I(\psi(c_{ik}, c_{jk} = 1))}{\sum_{k=1}^{n} w_k I(\psi(c_{ik}, c_{jk} = 1)) + \sum_{k=1}^{n} w_k I(\psi(c_{ik}, c_{jk} = 0))}$$

where, $I(x)$ is the indicator function, w_k are the weights calculated using the TF-IDF scheme (described later) and $\psi(x, y)$ is the pair-wise comparison of each *case* c_{ik} and c_{jk} where:

$$\psi(c_{ik}, c_{jk}) = \begin{cases} undef & \text{if } c_{ik} = c_{jk} = \phi \\ 1 & \text{if } c_{ik} = c_{jk} \neq \phi \\ 0 & \text{otherwise} \end{cases}$$

The weights w_k in the Jaccard measure are obtained using term-frequency and inverse document frequency (TF-IDF) scheme, borrowed from Lucene full-text indexing. The term-frequency in the given document gives a measure of the importance of the term within the particular document. The inverse document frequency is a measure of the general importance of the term. It is given by the log of the number of all documents divided by the number of documents containing the term. Thus,

$$TF = \frac{N_i}{\sum_k N_k}$$

$$TF - IDF = TF.log(\frac{D}{d_j \supset t_i})$$

where N_i is the number of occurrences of the considered term, D is the total number of documents in the corpus, and $d_j \supset t_i$ is the number of documents where the term t_j appears such that $N_j \neq 0$. A high weight in TF-IDF is reached by a high term-frequency (in the given document) and a low document frequency of the term in the whole collection of documents. Thus, the weights tend to filter out common terms. TF-IDF is a statistical technique used to evaluate how important a word is to a document. The importance increases proportionally to the number of times a word appears in the document but is offset by how common the word is in all of the documents in the collection or corpus. TF-IDF is often used by search engines to find the most relevant documents to a user's query. Whereas for event retrieval, the TF-IDF scheme helps in weighting the importance of a particular *case* in an event. The final ranking of the event instances is based on the weighted Jaccard measure with the query event, returned in descending order of percentage matching.

6.4 Results

I performed experiments for event indexing and retrieval in videos for the meeting, railroad monitoring and surveillance domains. These videos contain multiple agents that act

Table 6.1: SUMMARY OF INDEXED VIDEOS IN SEMORAN DATABASE

Dataset Name	Videos	Total Frames	Events	Sub-Events
NIST	6	2480	11	214
Kojima	20	2406	29	264
Sadiye	10	6461	13	104
VACE	40	18724	118	1296
PETS	143	45430	343	2040
CAVIAR	33	28692	91	1507
ETISEO	32	33184	131	1870
Alexei	12	1894	12	48
FDOT	85	14271	98	1524
Meeting	37	4383	37	373
Surveillance	30	15352	35	673
Railroad	46	9595	51	552
Meeting Test	15	1551	15	224
Surveillance Test	25	7836	25	544
Railroad Test	8	2260	9	307

independently or interact with each other or objects. The indexed videos, in all domains (in my experiments) totalled 194,519 frames, having 11,540 sub-events and 1013 events. A total number of 541 videos were adopted for indexing 16 events. I used seven standard video datasets as well as other videos for indexing the database and testing the event retrieval using Semoran system. The summary of event indexing using different datasets is provided in Table 6.1. The three sections in the table show the standard dataset, my dataset, and event retrieval testing dataset respectively.

A weighted Jaccard distance was used as a metric for retrieving relevant events from the database. The evaluation of the Jaccard Coefficient is shown in Figure 6.4, where its

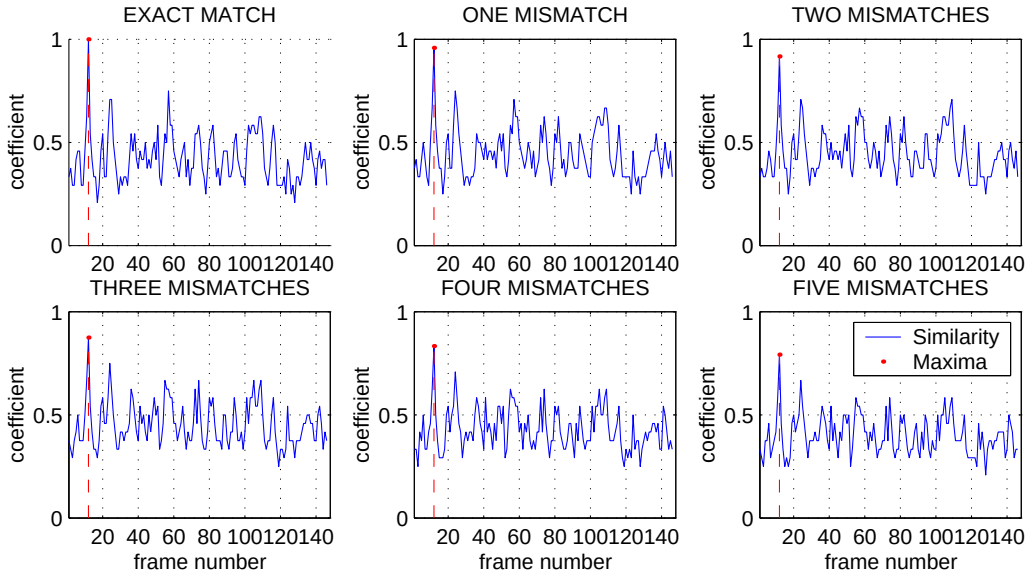


Figure 6.4: Event matching using the weighted Jaccard coefficient. A predefined event graph consisting of six vertices (case-frames) is matched with an event graph of a video sequence consisting of 148 vertices (case-frames). The correct match occurs at the graph node at frame 12 (the similarity maximum is indicated by the dotted red line). From top-left to bottom-right, the pre-defined predicate is perturbed so that a progressively greater number of *cases* within the case-frames mismatch.

robustness to event matching is depicted. I also estimated the average precision for the ranked event retrieval results using:

$$Average\ Precision = \frac{\sum_{r=1}^{N} Precision(r)}{Number\ of\ relevant\ events\ retrieved}$$

where, N is the total number of events retrieved and r is the rank of the retrieved event.

Since the retrieved events are ranked based on percentage match of the weighted Jaccard measure with the query event, I calculate the recall level precision estimates that provide a better picture of the ranking in event retrieval used in the Semoran system. The resulting

plots in the meeting, surveillance, and railroad monitoring domains are shown in Figure 6.5. As can be seen from the figure, the meeting event queries return the best ranked events from the database, followed by surveillance event queries and then the railroad monitoring domain queries. The reason behind such a ranking is that the meeting events are simpler and easily distinguishable from the rest of the events. The railroad events are the most complicated events with large variation in the order of sub-events. To make matters worse they are also similar to other railroad events, differing by a few keywords in the query. Thus, the ranked results for railroad monitoring queries are the worst among the three domains. The surveillance domain is composed of simple as well as complicated events, thus their ranked results fall in between the meeting and railroad monitoring event ranking.

The estimation of recall level precision plot is explained through the following example. Suppose 17 events are retrieved and there are 10 relevant events in the database. Also, the ranking of the retrieved events is R, R, I, R, R, I, I, R, R, R, R, I, I, R, R, I, I; where R = relevant event and I = irrelevant event. Firstly, calculate the average precision at recall values of {0.1, 0.2, ... 1.0}, as shown in the following table, and secondly, fit a spline to these points to draw the recall level precision plot:

Recall	Avg. Prec.
0.1	1/1=1.0
0.2	2/2=1.0
0.3	3/4=0.75
0.4	4/5=0.8
0.5	5/8=0.625

Recall	Avg. Prec.
0.6	6/9=0.67
0.7	7/10=0.7
0.8	8/11=0.72
0.9	9/14=0.64
1.0	10/15=0.67

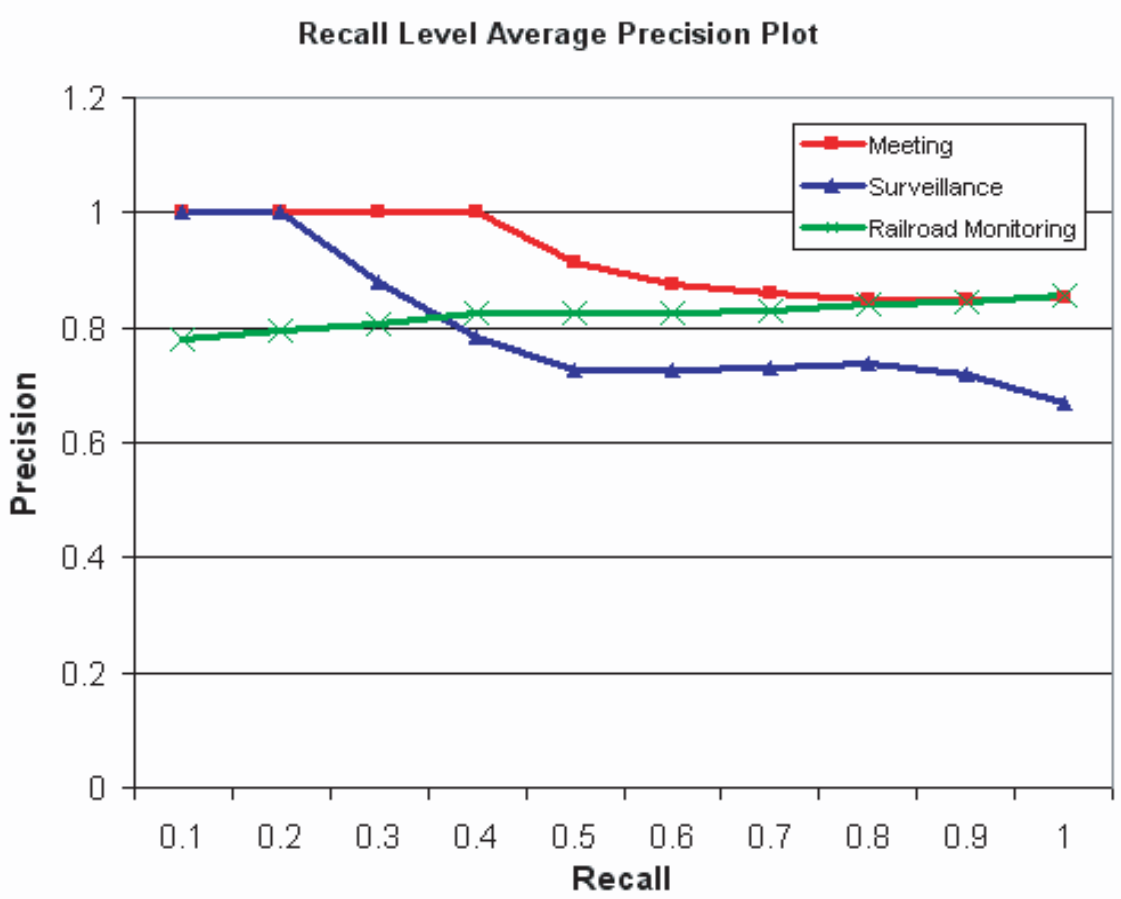


Figure 6.5: Recall level precision plots for the three domains.

A summary of event retrieval results with average precision (non-interpolated) values is supplied in Table 6.2. The average non-interpolated precision is computed by averaging the precision at recall values of {0.1, 0.2, ... 1.0}.

Table 6.2: SUMMARY OF EVENT RETRIEVAL USING THE SEMORAN SYSTEM

Test Video	Query Events	Average Precision (non-interpolated) %
Meeting	15	91.19
Surveillance	25	79.71
Railroad Monitoring	9	82.27

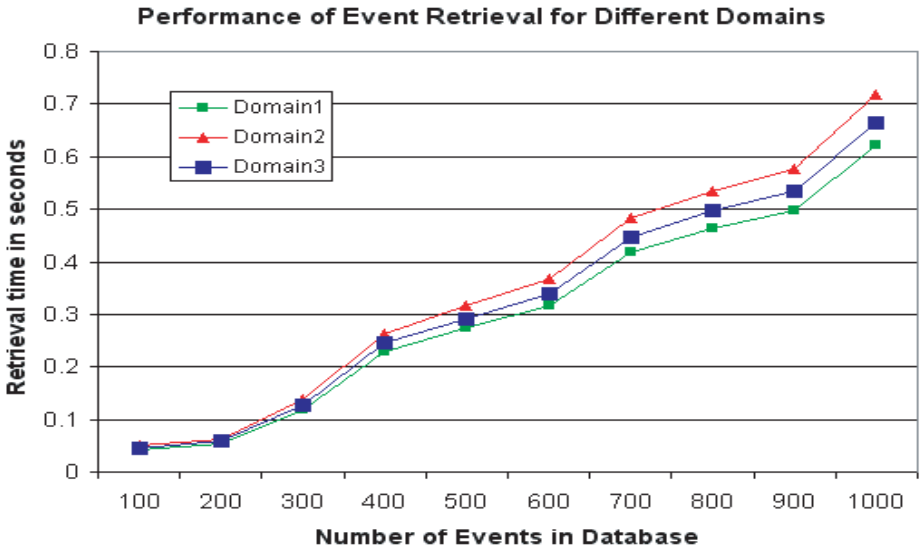


Figure 6.6: Average time for event retrieval given a query, in the meeting (domain1), surveillance (domain2), and railroad (domain3) domains respectively.

I also evaluated the time taken for event retrieval for varying size of event database, given query events, in Figure 6.6. I ran the Semoran system in Matlab 7.0 on a Intel Core Duo 2 E6600 machine running at 2.4 GHz with 2 GB of RAM. Each query was run 10 times with varying size of event database and the average time taken for each query is plotted for the three different domains.

6.5 Discussion

In this chapter, I described the Semoran system that is built for indexing and retrieval of events. Given the different instances of a particular event, I build an event index in the form of P-CASE representation. Given a query in the P-CASE representation, event retrieval

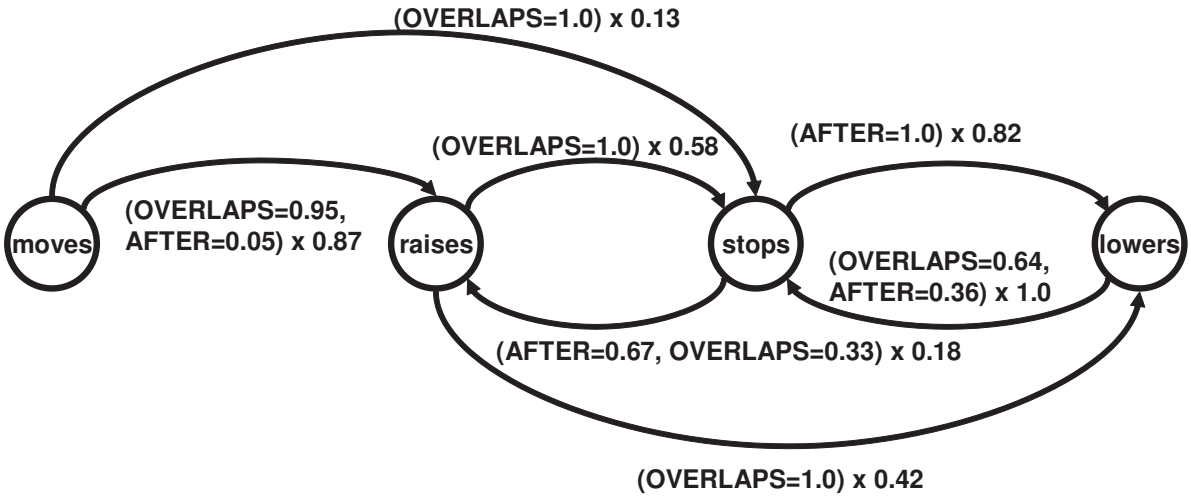
is a two-level process. At the first level, a ML estimate is computed with the different event models. The event model with the ML estimate provides the maximum matching event. At the second level, I find the percentage match of the query event with all the event *instances* belonging to the maximum matched event model, using a weighted Jaccard similarity measure. The weights in the Jaccard measure are obtained using term-frequency and inverse document frequency (TF-IDF) scheme, borrowed from Lucene full-text indexing. In text search, the TF-IDF scheme is used to return the ranked search results, whereas for event retrieval, the TF-IDF scheme helps in weighting the importance of a particular *case* in an event. The main advantages of my event indexing and retrieval scheme are:

1. The event indexing preserves the order of predicate occurrence that is necessary for event retrieval.

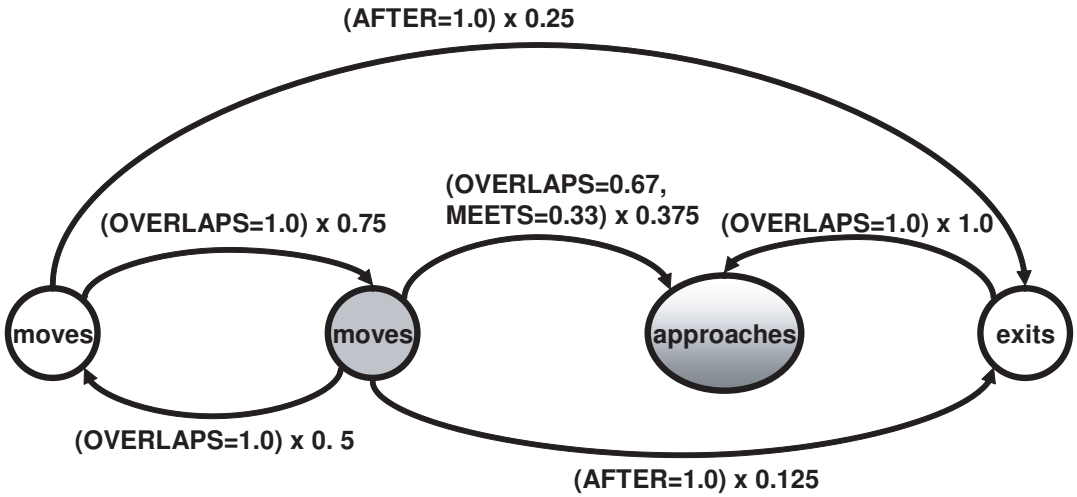
2. The event indexing scheme is incremental, thus the event model is updated incrementally using the new event instances, instead of recalculating the event model using all previous and new event instances.

3. The event retrieval requires less number of hits since it rules out most of the events (and their instances) during the first level search.

4. The event indexing and retrieval is scalable since new events can be added to the database.



(a)



(b)

Figure 6.7: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Voting (b) Chasing

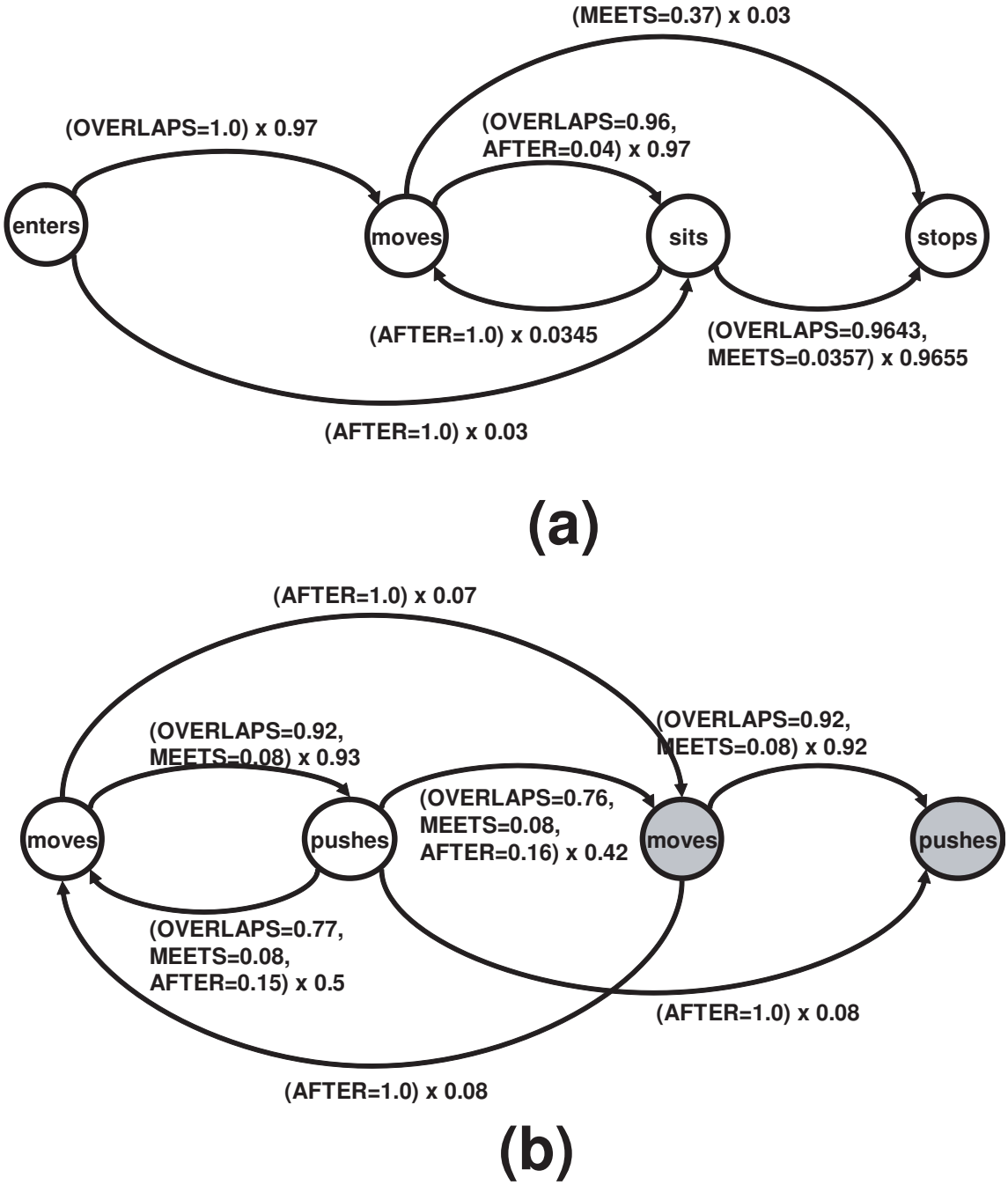


Figure 6.8: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Enter and sit (b) Fighting

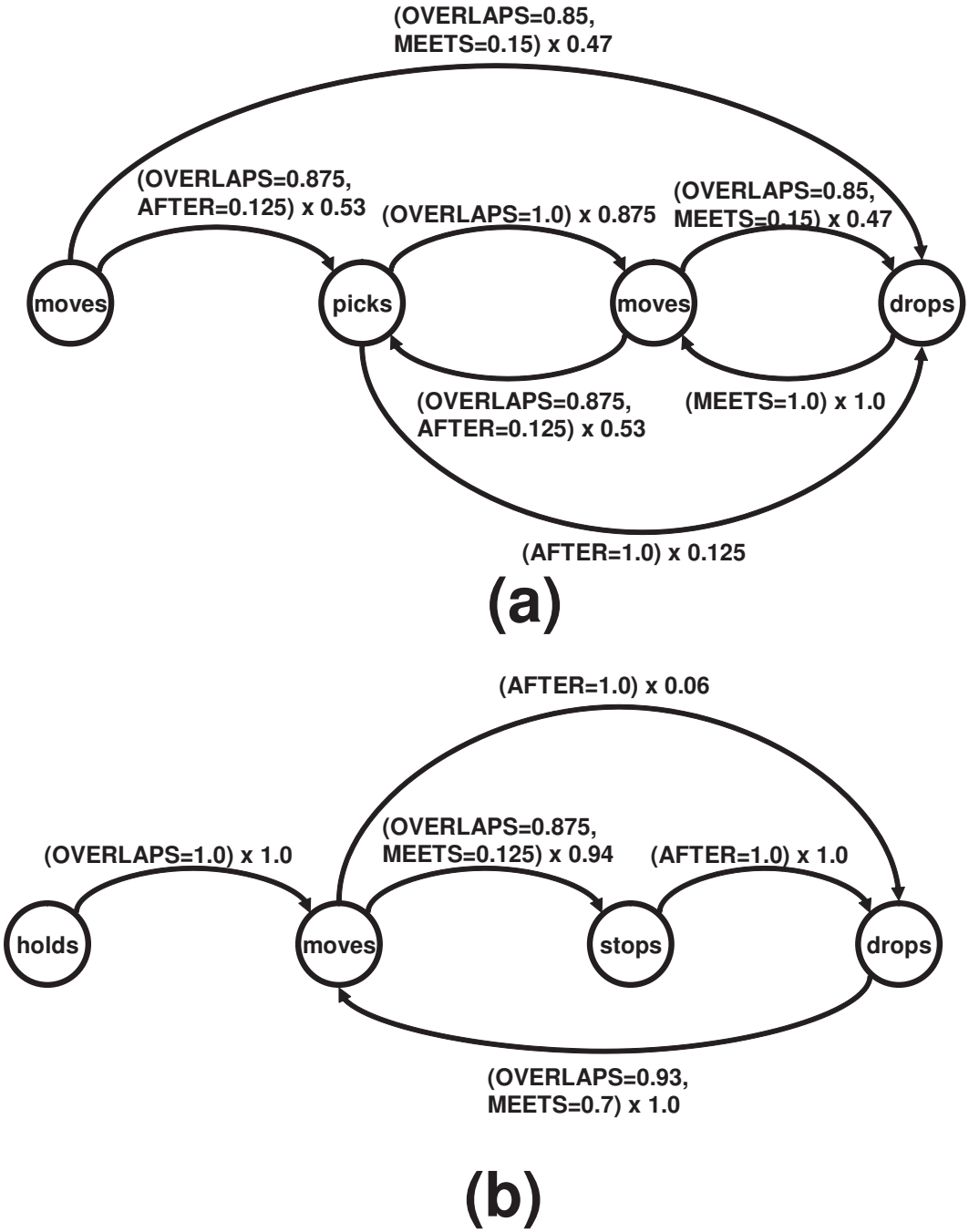


Figure 6.9: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Loading (b) Object drop

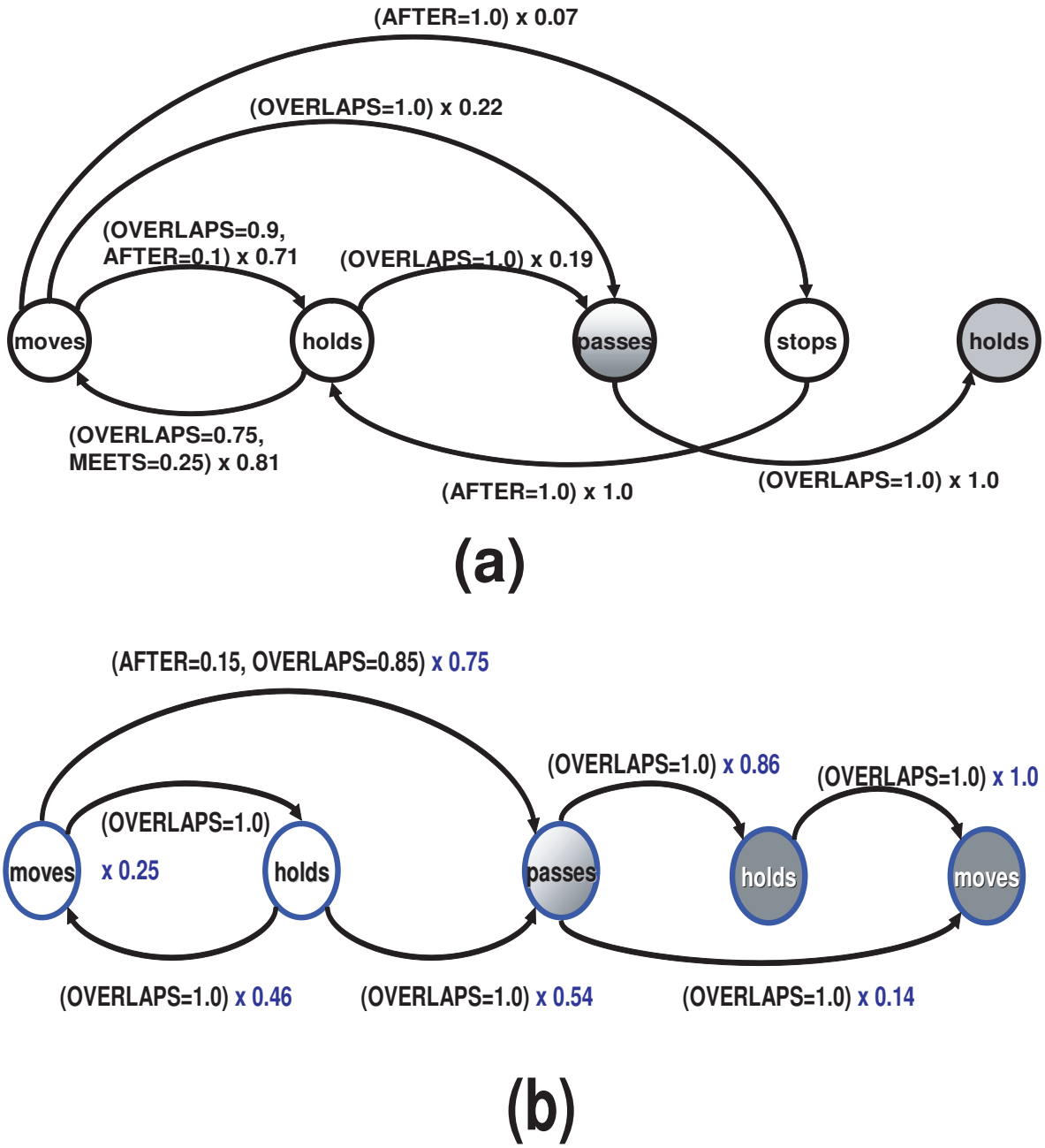


Figure 6.10: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Object exchange (b) Object passing

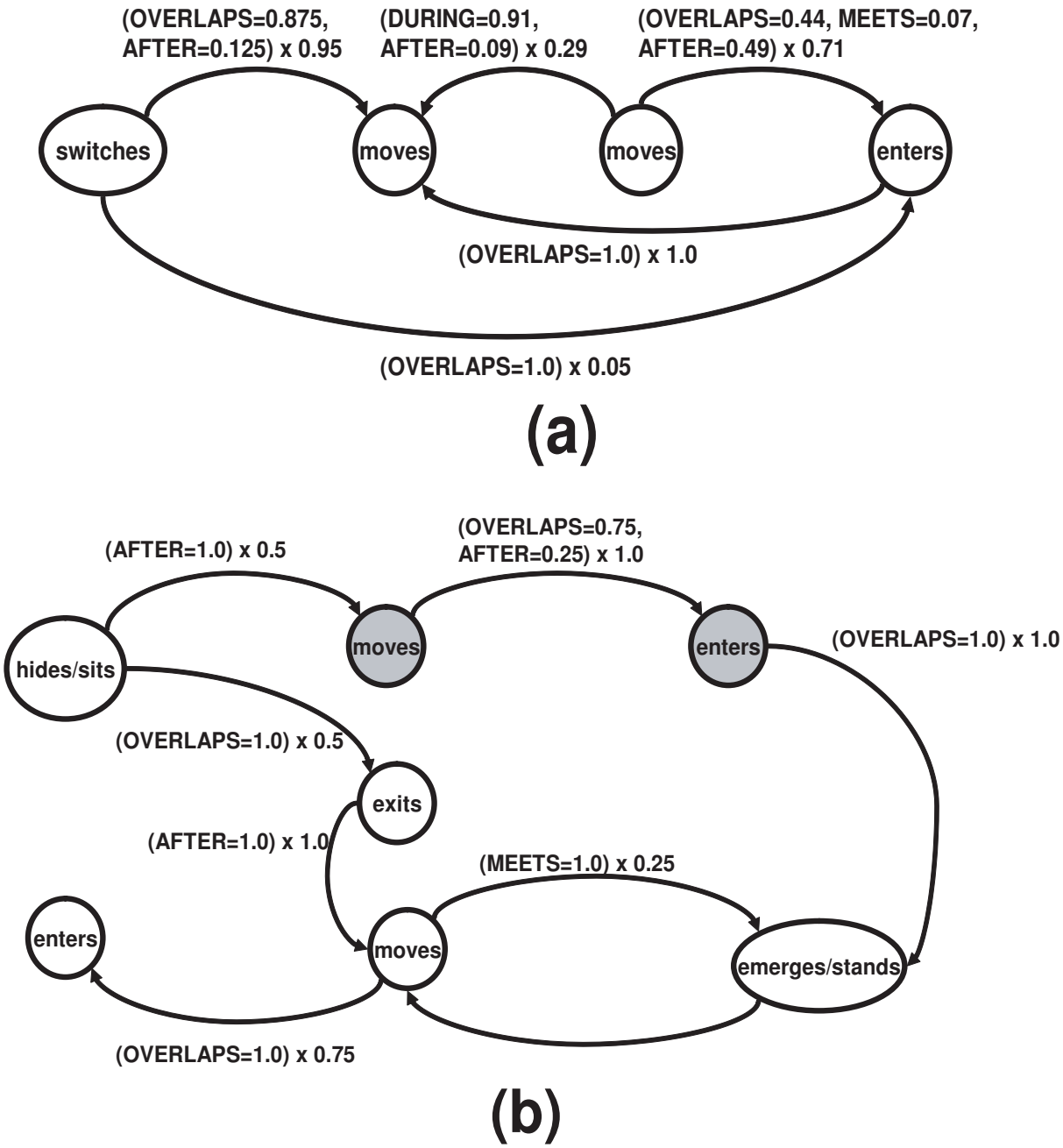


Figure 6.11: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Person enters danger-zone while gate arms moving (b) Sneaking

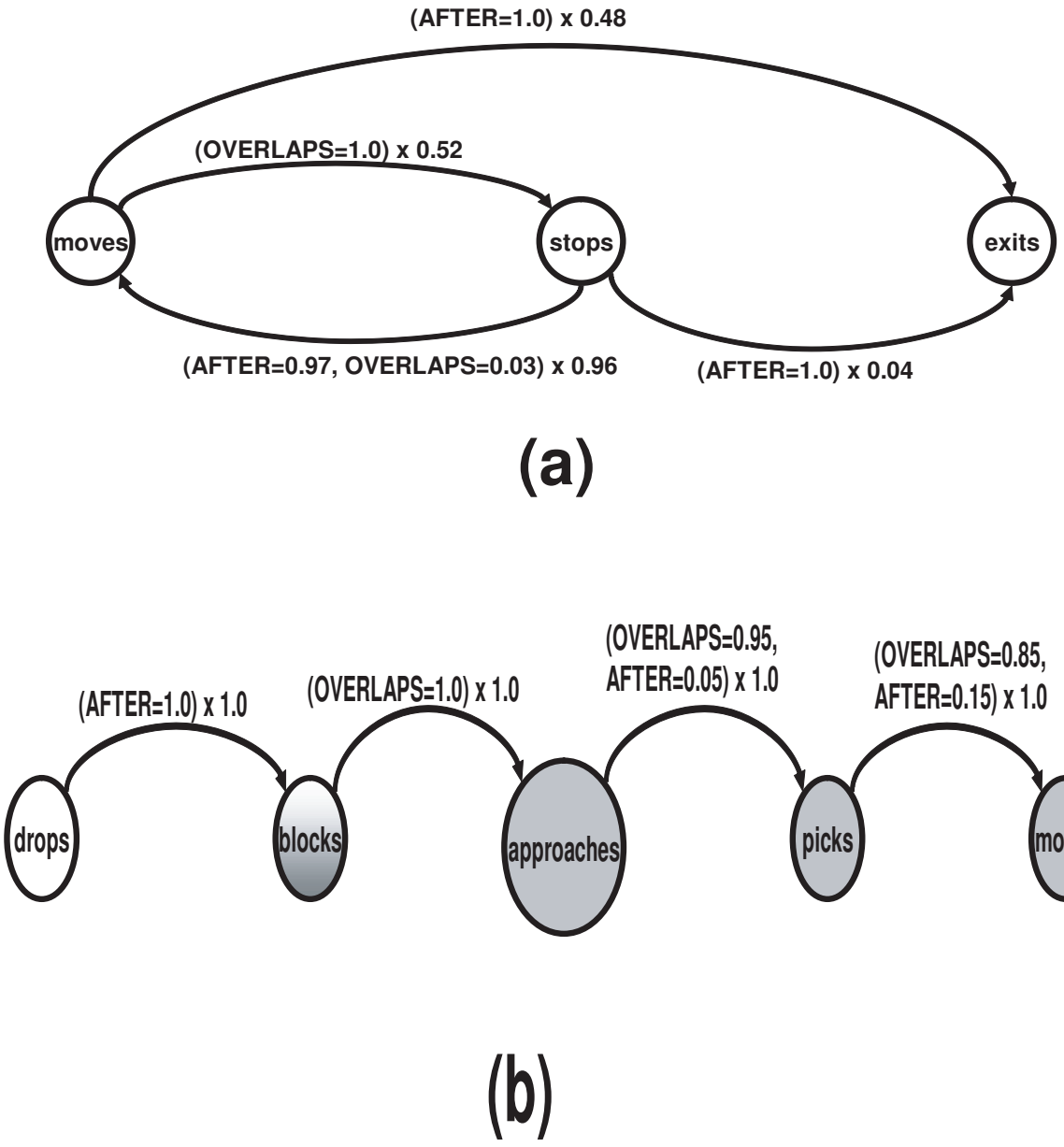


Figure 6.12: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Stand and leave (b) Stealing

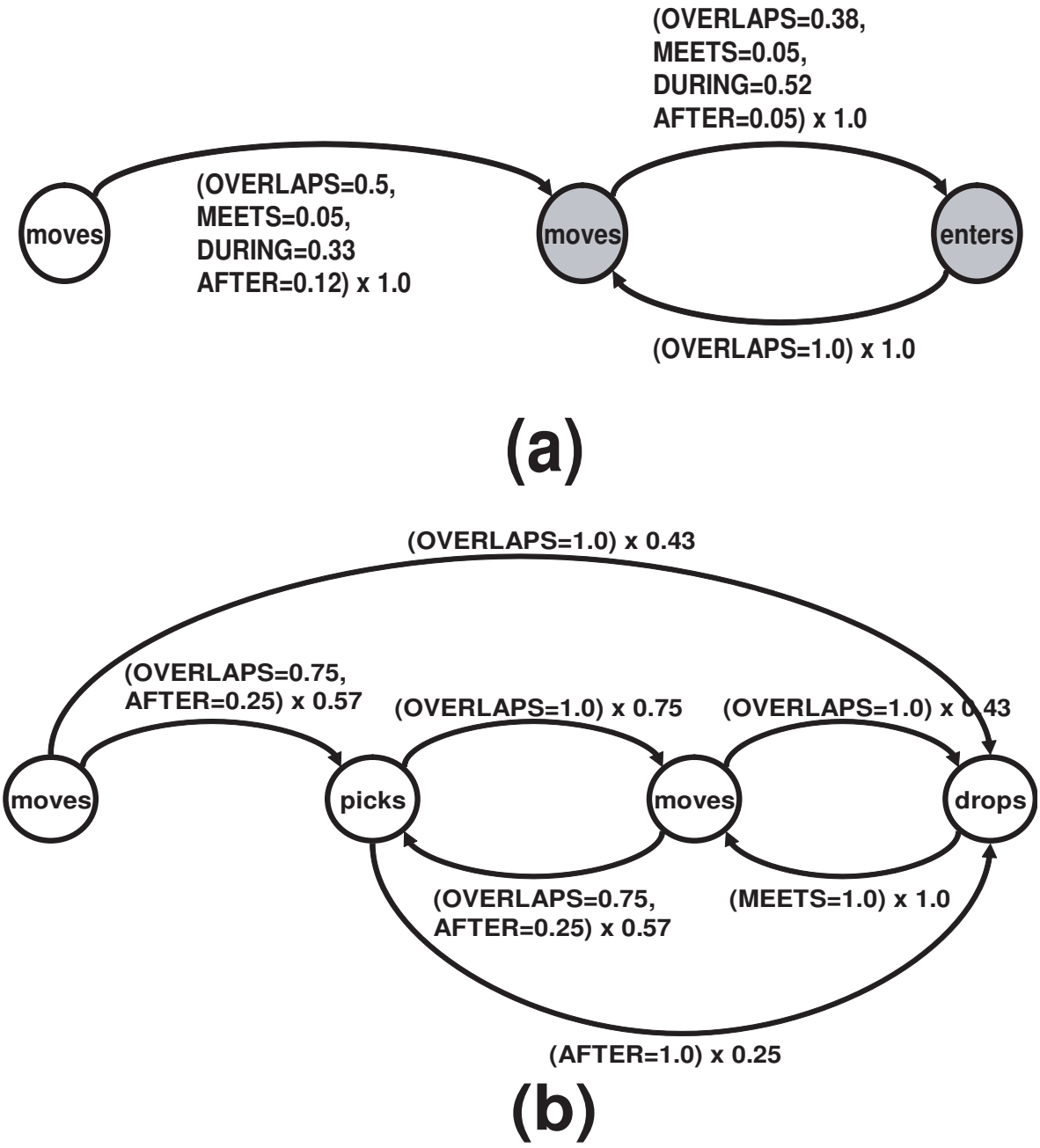


Figure 6.13: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Train enters danger-zone while gate arms moving (b) Unloading

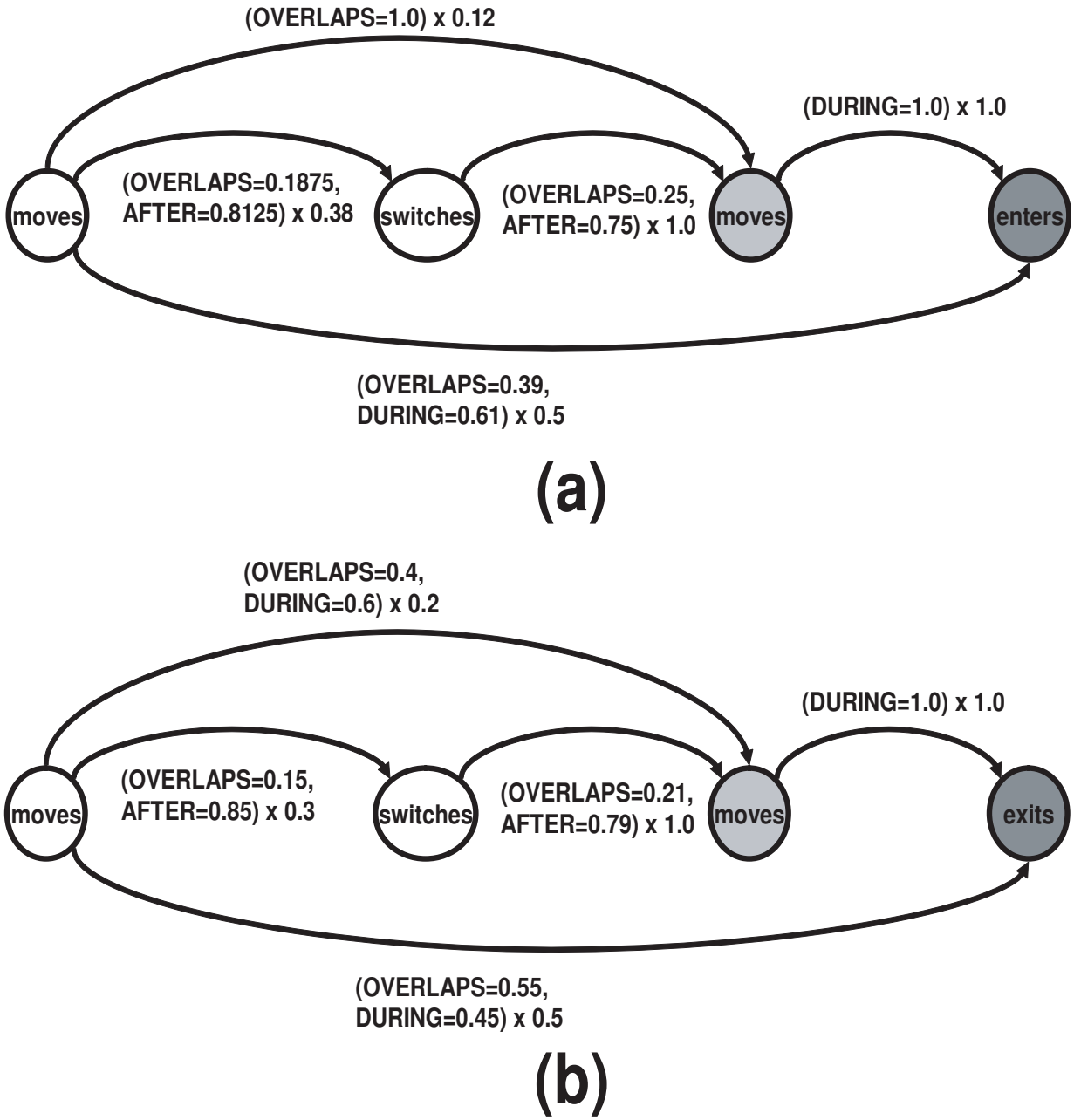


Figure 6.14: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Vehicle enters danger-zone while gate arms moving (b) Vehicle exits danger-zone while gate arms moving

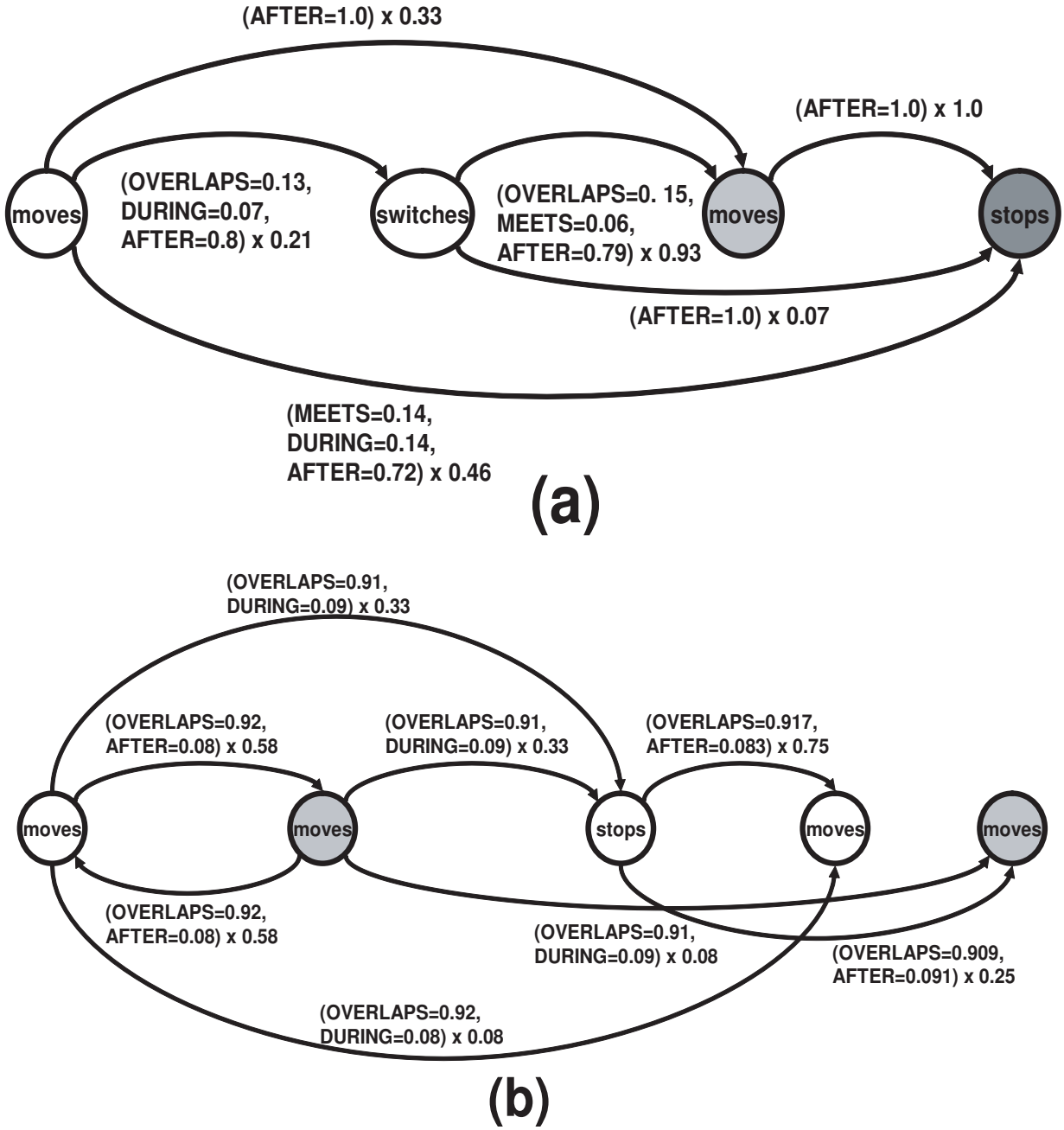


Figure 6.15: Automatically generated P-CASE representation using indexed data in Semoran system. (a) Vehicle stops outside danger-zone (b) Argument

CHAPTER 7

SUMMARY AND FUTURE WORK

The problem of detecting events in a video involving multiple agents and their interaction was identified. Event models were learnt from training videos having variations in the number of agents and the temporal order of sub-events. Event learning was formulated in a probabilistic framework, and the learnt event models were used for event detection in novel videos. Event detection was treated as a graph theoretic clustering of sub-events having high association within the event clusters and low association outside the clusters. I demonstrated my event learning and detection methods on videos in the railroad monitoring, surveillance and meeting domains. The current limitation of the event learning and detection method is that it assumes a closed world problem and thus requires the knowledge of the sub-events that may occur in an event. Future research in this area may involve removing such a constraint for event learning and detection.

Future theoretical development of the event learning and detection approaches can be strengthened via inclusion of research from teamwork theory. Saavedra *et al.* [SEV93] developed an elegant framework to understand interdependence in teams by describing a set of complex interdependencies: (a) task interdependence, (b) goal interdependence, and (c) feedback interdependence. Future work in multi-agent event learning and detection

might benefit from considering these variations in coordination. First, task interdependence describes the degree to which the individuals within a team have to interact when performing their task. Saavedra *et al.* state that this can vary along four levels: pooled, sequential, reciprocal, or team interdependence. Under pooled interdependence, each team member performs his own task and this is pooled. With regard to sequential interdependence, this is said to occur when a team member's output is required for another team member's input. In reciprocal interdependence, a given team member's output becomes another member's input and vice versa. Last, the highest form of coordinated teamwork is team interdependence, where group members jointly diagnose the problem at hand and collaborate to complete a task. Second, goal interdependence classifies individual goals (for example, minimize your own exposure to enemy observation) and team goals (for example, try to minimize the team's exposure to enemy observation). Note that in large teams, there is likely to be combinations of these types of goals, that is, both team and individual goals are issued. Third, feedback interdependence varies based upon how it is delivered to individual team members or to the team and whether individual results are provided or team results are provided, that is, it can vary along the lines of limiting the knowledge of results or changing the focus of feedback (e.g., showing only the team's overall progress but not that of the individuals).

Also, the problem of formally representing events occurring in a video sequence using measurements in terms of object labels and tracks was identified. In order to represent events, cases were added to the original CASE framework to support multi-agent/thread, temporal logic and causal relationships. An event representation was developed to cater for

the temporal variations in the sub-events, and event ontologies were automatically extracted from training videos. Experiments were performed on real sequences for the on-line generation of P-CASE for human interaction. The essence of the proposition here is that, based on the temporal relationships of the agent motions and a description of its state, it is possible to build a formal description of an event. Although the P-CASE representation can represent causality, my methods do not infer causality, and thus is an interesting future direction to this work.

I also proposed the event-based indexing and retrieval of video. To that end, the Semoran system was developed that takes in user inputs in either of the three forms for event retrieval: using pre-defined queries in P-CASE representation, using custom queries in P-CASE representation, or query by example video. The system then searches the entire database and returns the matched videos to the user. The user can then play the returned videos, look at the actions performed by various agents in the storyboard, and visualize the actions in P-CASE representation. Internally, the system performs a two-level search. At the first level, the system matches the query with the different event models indexed in the database, using maximum likelihood estimates. Once it finds a match, at the second level, it performs a thorough search with all the relevant events belonging to the matched model, using weighted Jaccard similarity.

There are several future directions to this work with possible exploratory solutions as follows:

1. **Unusual Behavior Detection**: Unusual behaviors can be detected using P-CASE by learning what constitutes 'normalcy' in each scene in terms of interaction of individuals and materials. Thus, if an event involving multiple individuals occurs, such as a crowd forming or an unusual interaction between persons with a low probability of occurrence, the video will be flagged and the event will be construed as an unusual activity. In this way, in addition to analyzing the behavior of individuals, exploration of analyzing the behavior of groups as well as to detect anomalies in a scene is an interesting direction of future research.

2. **Video Summarization**: For video summarization, the system may automatically generate the summary of the given video by extracting and stitching clips containing important events.

3. **Query by Sketch**: For query by sketch, the user may input the query in the form of motion trajectories of various individuals and the system returns the events containing that kind of motion.

4. **Query in Human Sentence**: For query in human sentences, the user may input the query in the form of a sentence and the system extracts the relevant keywords and automatically generate the P-CASE representation for retrieval of relevant events.

5. **Situation Level Indexing**: Future theoretical development of event indexing might be strengthened by adapting the situation level indexing. Comprehension models look more at the integration of information over time and they may help deal with more

complex forms of event indexing. Specifically, a complex plot unfolds in the video and viewers have little difficulty sequencing and integrating this type of input. At present, the current work is still only analogous to "sentence level" parsing, where as considering models that go beyond sentence comprehension to understand how entire sets of sentences and even paragraphs and pages of text are integrated to form a "situation model" of that text. One of the early prevalent theories is by Kinsch [K88] called the construction integration model [KV78]. These models can be ported from text and extended to videos for the detection and indexing of long complex events in videos.

APPENDIX A: SUB-EVENT DETECTION RULES

1. *Moves*(Entity)

   If $|previous\_Entity\_position - current\_Entity\_position| > Threshold$

   return *true*

2. *Stops*(Entity)

   If $|previous\_Entity\_position - current\_Entity\_position| < Threshold$

   return *true*

3. *Enters*(Agent)

   If $NOT(in\_fov(previous\_Agent\_position))$ AND $in\_fov(current\_Agent\_position)$

   return *true*

4. *Exits*(Agent)

   If $NOT(in\_fov(current\_Agent\_position))$ AND $in\_fov(previous\_Agent\_position)$

   return *true*

5. *Approaches*(Agent,Entity)

   If $current\_distance(Agent, Entity) < previous\_distance(Agent, Entity)$

   return *true*

6. *Leaves*(Agent,Entity)

   If $current\_distance(Agent, Entity) > previous\_distance(Agent, Entity)$

   return *true*

7. *Extends*(Agent,{hand})

   If $current\_distance(Agent, \{hand\}) > previous\_distance(Agent, \{hand\})$

   return *true*

8. *Holds*(Agent,Object)

   If $current\_distance(Agent, Object) < Threshold \;\; AND \;\; Moves(Agent)$

   $AND \;\; Moves(Object)$

   return *true*

9. *Picks*(Agent,Object)

   If $previous\_distance(Agent, Object) > Threshold \;\; AND \;\; Holds(Agent, Object)$

   return *true*

10. *Passes*(Agent1,Agent2,Object)

    If $Holds(Agent1, Object) \;\; AND \;\; Drops(Agent1, Object) \;\; AND \;\; Holds(Agent2, Object)$

    return *true*

11. *Drops*(Agent,Object)

    If $previous\_distance(Agent, Object) < Threshold \;\; AND$

    $current\_distance(Agent, Object) > Threshold$

    return *true*

12. *Raises*({head},{hand})

    If $current\_position(Above(\{hand\}, \{head\})) \;\; AND$

$previous\_position(Below(\{hand\}, \{head\}))$

return *true*

13. *Lowers*({head},{hand})

If $current\_position(Below(\{hand\}, \{head\}))$ AND

$previous\_position(Above(\{hand\}, \{head\}))$

return *true*

14. *Sits*(Agent)

If $previous\_y\_position(Agent) > current\_y\_positon(Agent)$ AND $NOT(Move(Agent))$

return *true*

15. *Stands*(Agent)

If $previous\_y\_position(Agent) < current\_y\_positon(Agent)$ AND $Move(Agent)$

return *true*

16. *Pushes*(Agent1,Agent2)

If $previous\_distance(Agent1, Agent2) < Threshold$ AND

$current\_distance(Agent1, Agent2) > Threshold$ AND

$Leaves(Agent1, Agent2)$

return *true*

17. *Blocks*(Agent1,Agent2,Object)

If $current\_distance(Agent1, Agent2) < Threshold$ AND

$current\_distance(Agent1, Object) > current\_distance(Agent2, Object)$

return *true*

18. *Crouches*(Agent)

If $previous\_y\_position(Agent) > current\_y\_positon(Agent) \;\; AND \;\; Move(Agent)$

return *true*

19. *Hides*(Agent,Object)

If $previous\_distance(Agent, Object) < Threshold \;\; AND \;\; Exits(Agent) \;\; AND$

$in\_fov(Object)$

return *true*

20. *Emerges*(Agent,Object)

If $current\_distance(Agent, Object) < Threshold \;\; AND \;\; Enters(Agent) \;\; AND$

$previous\_frame(in\_fov(Object))$

return *true*

21. *Collides*(Agent,Entity)

If $previous\_distance(Agent, Entity) > Threshold \;\; AND$

$current\_distance(Agent, Entity) < Threshold \;\; AND \;\; Stops(Agent) \;\; AND \;\; Stops(Entity)$

return *true*

22. *Breaks*(Agent,Entity)

If $Collides(Agent, Entity) \;\; AND$

$is\_different(previous\_shape(Agent), current\_shape(Agent))$

return *true*

23. *Switches*(Object)

If $is\_different(previous\_shape(Object), current\_shape(Object))$

return *true*

APPENDIX B: PROOF OF EQUIVALENCY FOR W AND $\tilde{W}$ BASED MINIMIZATIONS

Given W, the global criterion for minimization of Ncut function is given by:

$$\begin{aligned} Ncut(A,B) &= min[\frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)}] \\ &= min[\frac{\sum_{i\in A, j\in B} P(v_j|v_i) + \sum_{i\in A, j\in B} P(v_i|v_j)}{\sum_{i\in A, k\in I} P(v_k|v_i)} \\ &+ \frac{\sum_{i\in A, j\in B} P(v_j|v_i) + \sum_{i\in A, j\in B} P(v_i|v_j)}{\sum_{j\in B, k\in I} P(v_k|v_j)}] \end{aligned}$$

where $I = A \bigcup B$, and since W is symmetric therefore $P(v_j|v_i) = P(v_i|v_j)$. Thus the above equation is equivalent to:

$$Ncut(A,B) = min[\frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{i\in A, k\in I} P(v_k|v_i)} + \frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{j\in B, k\in I} P(v_k|v_j)}] \tag{7.1}$$

Similarly, given $\tilde{W}$, the global criterion for minimization of Ncut function is given by:

$$\begin{aligned} Ncut(A,B) &= min[\frac{\sum_{i\in A, j\in B} 2P(v_j|v_i) + \sum_{i\in A, j\in B} 2P(v_i|v_j)}{\sum_{i\in A, k\in I} P(v_k|v_i) + \sum_{i\in A, k\in I} P(v_i|v_k)} \\ &+ \frac{\sum_{i\in A, j\in B} 2P(v_j|v_i) + \sum_{i\in A, j\in B} 2P(v_i|v_j)}{\sum_{j\in B, k\in I} P(v_k|v_j) + \sum_{j\in B, k\in I} P(v_j|v_k)}] \end{aligned}$$

and since $\tilde{W} = \hat{W} + \hat{W}^T$, where $\hat{W}$ is upper triangle matrix, therefore $P(v_i|v_j) = P(v_i|v_k) = P(v_j|v_k) = 0$. Thus the above equation is reduced to:

$$Ncut(A,B) = min[\frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{i\in A, k\in I} P(v_k|v_i)} + \frac{\sum_{i\in A, j\in B} 2P(v_j|v_i)}{\sum_{j\in B, k\in I} P(v_k|v_j)}] \tag{7.2}$$

Since both equations (7.1) and (7.2) minimize the same function, thus it is equivalent to deal with W and $\tilde{W}$.

APPENDIX C: MOST LIKELY SEQUENCE OF SUB-EVENTS FOR EVENTS IN SEMARON SYSTEM

Given the P-CASE representations of events, I find the most likely sequence of sub-events by calculating the maximum likelihood estimate of all event instances using the method described in section 5.3.1. The following are the automatically extracted most likely sequence of sub-events for the events in the Semoran system:

1. **Argument:**

   [**PRED**: Moves, **AG**: Person1, **D**: hand, **FAC**: up-right, **SUB**:

   [**PRED**: Moves, **AG**: Person2, **D**: hand, **FAC**: up-left, **OVERLAPS**: Moves, **SUB**:

   [**PRED**: Stops, **AG**: Person1, **D**: hand, **OVERLAPS**: Moves, **SUB**:

   [**PRED**: Moves, **AG**: Person1, **D**: hand, **FAC**: down-left, **OVERLAPS**: Stops, **SUB**:

   [**PRED**: Moves, **AG**: Person2, **D**: hand, **FAC**: down-right, **OVERLAPS**: Moves]]]]]

2. **Chasing:**

   [**PRED**: Moves, **AG**: Person1, **FAC**: up-right, **SUB**:

   [**PRED**: Moves, **AG**: Person2, **FAC**: up-right, **OVERLAPS**: Moves, **SUB**:

   [**PRED**: Approaches, **AG**: Person2, **D**: Person1, **OVERLAPS**: Moves]]]

3. **Enter and Sit:**

   [**PRED**: Enters, **AG**: Person1, **SUB**:

   [**PRED**: Moves, **AG**: Person1, **FAC**: down-right, **OVERLAPS**: Enters, **SUB**:

   [**PRED**: Sits, **AG**: Person1, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Stops, **AG**: Person1, **OVERLAPS**: Sits]]]]

4. **Fighting:**

[**PRED**: Moves, **AG**: Person1, **FAC**: up-left, **SUB**:

[**PRED**: Pushes, **AG**: Person1, **D**: Person2, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Moves, **AG**: Person2, **FAC**: down-right, **OVERLAPS**: Pushes, **SUB**:

[**PRED**: Pushes, **AG**: Person2, **D**: Person1, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Moves, **AG**: Person1, **FAC**: down-left, **OVERLAPS**: Pushes]]]]]

5. **Loading:**

[**PRED**: Moves, **AG**: Person1, **FAC**: up-right, **SUB**:

[**PRED**: Picks, **AG**: Person1, **OBJ**: box, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Moves, **AG**: Person1, **FAC**: up-left, **OVERLAPS**: Picks, **SUB**:

[**PRED**: Drops, **AG**: Person1, **OBJ**: box, **LOC**: truck, **OVERLAPS**: Moves]]]]

6. **Object Drop:**

[**PRED**: Holds, **AG**: Person1, **OBJ**: bag, **SUB**:

[**PRED**: Moves, **AG**: Person1, **FAC**: down-right, **OVERLAPS**: Holds, **SUB**:

[**PRED**: Stops, **AG**: Person1, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Drops, **AG**: Person1, **OBJ**: bag, **AFTER**: Stops, **SUB**:

[**PRED**: Moves, **AG**: Person1, **FAC**: up-right, **OVERLAPS**: Drops]]]]]

7. **Object Exchange:**

[**PRED**: Moves, **AG**: Person2, **FAC**: down-right, **SUB**:

[**PRED**: Holds, **AG**: Person2, **OBJ**: bag, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Moves, **AG**: Person2, **FAC**: up-right, **OVERLAPS**: Holds]]]

8. **Object Passing:**

[**PRED**: Holds, **AG**: Person2, **D**: left-hand, **OBJ**: paper, **SUB**:

[**PRED**: Moves, **AG**: Person2, **D**: left-hand, **FAC**: down-left, **OVERLAPS**: Holds, **SUB**:

[**PRED**: Passes, **AG**: {Person2,Person1}, **D**: hand, **OBJ**: paper, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Holds, **AG**: Person1, **D**: right-hand, **OBJ**: paper, **OVERLAPS**: Passes, **SUB**:

[**PRED**: Moves, **AG**: Person1, **D**: right-hand, **FAC**: up-left, **OVERLAPS**: Holds]]]]]

9. **Sneaking:**

[**PRED**: Hides, **AG**: Person1, **OBJ**: bushes, **SUB**:

[**PRED**: Moves, **AG**: Person2, **FAC**: up-right, **AFTER**: Hides, **SUB**:

[**PRED**: Enters, **AG**: Person2, **OBJ**: door, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Emerges, **AG**: Person1, **OBJ**: bushes, **OVERLAPS**: Enters, **SUB**:

[**PRED**: Moves, **AG**: Person1, **FAC**: up-left, **OVERLAPS**: Emerges, **SUB**:

[**PRED**: Enters, **AG**: Person1, **OBJ**: door, **OVERLAPS**: Moves]]]]]]

10. **Stand and Leave:**

[**PRED**: Moves, **AG**: Person1, **FAC**: up-right, **SUB**:

[**PRED**: Stops, **AG**: Person1, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Moves, **AG**: Person1, **FAC**: up-left, **AFTER**: Stops, **SUB**:

[**PRED**: Exits, **AG**: Person1, **AFTER**: Moves]]]]

11. **Stealing:**

[**PRED**: Drops, **AG**: Person1, **OBJ**: box, **SUB**:

[**PRED**: Blocks, **AG**: {Person2,Person3}, **D**: Person1, **AFTER**: Moves, **SUB**:

[**PRED**: Approaches, **AG**: Person5, **OBJ**: box, **OVERLAPS**: Blocks, **SUB**:

[**PRED**: Picks, **AG**: Person5, **OBJ**: box, **OVERLAPS**: Approaches, **SUB**:

[**PRED**: Moves, **AG**: Person5, **FAC**: down-left, **OVERLAPS**: Picks]]]]]

12. **Unloading:**

[**PRED**: Moves, **AG**: Person1, **FAC**: up-right, **SUB**:

[**PRED**: Picks, **AG**: Person1, **OBJ**: box, **LOC**: truck, **OVERLAPS**: Moves, **SUB**:

[**PRED**: Moves, **AG**: Person1, **FAC**: up-left, **OVERLAPS**: Picks, **SUB**:

[**PRED**: Drops, **AG**: Person1, **OBJ**: box, **LOC**: cart, **OVERLAPS**: Moves]]]]

13. **Voting:**

    [**PRED**: Moves, **AG**: Person1, **D**: left-hand, **FAC**: up-left, **SUB**:

    [**PRED**: Raises, **AG**: Person1, **D**: left-hand, **OVERLAPS**: Moves, **SUB**:

    [**PRED**: Stops, **AG**: Person1, **D**: left-hand, **OVERLAPS**: Raises, **SUB**:

    [**PRED**: Lowers, **AG**: Person1, **D**: left-hand, **AFTER**: Stops]]]]

14. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle stops outside Zone2**

    [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

    [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

    [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

    [**PRED**: Stops, **AG**: Vehicle, **LOC**: Zone2, **FAC**: Outside, **AFTER**: Moves]]]]

15. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle enters while gate is in motion**

    [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

    [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

    [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

[**PRED**: Enters, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves]]]]

16. **train approaches ⇒ signal switches on ⇒ gate arm moves down ⇒ vehicle exits while gate is in motion**

    [**PRED**: Moves, **AG**: Train, **OBJ**: Signal, **LOC**: Zone1, **FAC**: Towards, **SUB**:

    [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **AFTER**: Moves, **SUB**:

    [**PRED**: Moves, **OBJ**: Gate, **FAC**: Down, **AFTER**: Switches, **SUB**:

    [**PRED**: Exits, **AG**: Vehicle, **LOC**: Zone2, **DURING**: Moves]]]]

17. **Person enters zone2 while signal was switched on**

    [**PRED**: Switches, **OBJ**: Signal, **FAC**: On, **SUB**:

    [**PRED**: Moves, **AG**: Person, **LOC**: Zone2, **FAC**: Towards, **OVERLAPS**: Switches, **SUB**:

    [**PRED**: Enters, **AG**: Person, **LOC**: Zone2, **AFTER**: Moves]]]

18. **Train entering zone2 while gates are in motion**

    [**PRED**: Moves, **OBJ**: Gates, **FAC**: Down, **SUB**:

    [**PRED**: Moves, **AG**: Train, **LOC**: Zone2, **FAC**: Towards, **OVERLAPS**: Moves, **SUB**:

    [**PRED**: Enters, **AG**: Train, **LOC**: Zone2, **DURING**: Moves]]

LIST OF REFERENCES

[AA01] Ali, A., and Aggarwal, J. K. 2001. Segmentation and Recognition of Continuous Human Activity. In *IEEE Workshop of International Conference on Computer Vision*, pp.28-38.

[AF94] Allen, J. F., and Ferguson, G. 1994. Actions and Events in Interval Temporal Logic. In *Journal of Logic Computation*, vol.4(5), pp.531-579.

[AS01] Ayers, D., and Shah, M. 2001. Monitoring Human Behavior from Video Taken in an Office Environment. In *Image and Vision Computing*, vol.19, pp.833-846.

[BJ98] Babaguchi, N., and Jain, R. 1998. Event Detection from Continuous Media. In *Proc. of International Conference on Pattern Recognition*, pp.1209-1212.

[B75] Badler, N. 1975. Temporal Scene Analysis: Conceptual Description of Object Movements. In *University of Toronto Technical Report No. 80.*

[BBY04] M. Balcan, A. Blum, and K. Yang. 2004. Co-training and expansion: Towards briding the- ory and practice. In *Eighteenth Annual Conference on Neural Information Processing Systems.*

[BR94] Ben-Arie, J., and Rao, K. R. 1994. Optimal Template Matching by Non-Orthogonal Image Expansion Using Restoration. In *International Journal of Machine Vision and Applications*, vol.7(2), pp.69-81.

[BWP02] Ben-Arie, J., Wang, Z., Pandit, P., and Rajaram, S. 2002. Human Activity Recognition Using Multidimensional Indexing. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.24(8), pp.1091-1104.

[B85] J. O. Berger. 1985. Statistical decision theory and Bayesian analysis. In 2nd edition, Springer-Verlag, New York.

[BM98] A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *11th Annual Conference on Computational Learning Theory.*

[BW97] Boreczky, J. and Wilcox, L. 1997. A HMM Framework for Video Segmentation using Audio and Image Features. In *International Conference on Acoustics, Speech and Signal Processing.*

[B97] Brand, M. 1997. Understanding Manipulation in Video. In *International Conference on Face and Gesture Recognition*, pp.94-99.

[BK00] Brand, M., and Kettnaker, V. 2000. Discovery and Segmentation of Activities in Video. In *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol.22(8), pp.844-851.

[CCMSZ97] Chang, S., Chen, W., Meng, H., Sundaram, H., and Zhong, D. 1998. A Fully Automated Content Based Video Search Engine Supporting Spatio-Temporal Queries. In *IEEE Transactions of Circuits Systems and Video Technology*, vol.8(5), pp.602-615.

[CRM03] Comaniciu, D., Ramesh, V. and Meer, P. 2003. Kernel-based object tracking. In *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol.25(5), pp.564-577.

[DB97] Davis, J. W., and Bobick, A. F. 1997. The Representation and Recognition of Human Movement Using Temporal Templates. In *Proc. of Computer Vision and Pattern Recognition*, pp.928-934.

[DAW00] Dimitrova, N., Agnihotri, L., and Wei, G. 2000. Video Classification based on HMM using Text and Faces. In *Proc. of European Conference on Signal Processing.*

[D04] Ding, C. 2004. Tutorial on Spectral Clustering. In *International Conference on Machine Learning.*

[F68] Fillmore, C. J. 1968. The Case for CASE. In Bach, E. and Harms, R. eds., *Universals in Linguistic Theory*, pp.1-88, New York, NY:Holt, Rinehart, and Winston.

[FMR98] Friedman, N., Murphy, K., and Russell, S. 1998. Learning the Structure of Dynamic Probabilistic Networks. In *Proc. Conference on Uncertainty in Artificial Intelligence (UAI)*, pp.139-147, Madison, WI.

[GX03] Gong, S., and Xiang, T. 2003. Recognition of Group Activities using Dynamic Probabilistic Networks. In *Proc. of International Conference on Computer Vision*, pp.742-749.

[HQS00] Haering, N., Qian, R. J., and Sezan, M. I. 2000. A Semantic Event-Detection Approach and Its Application in Wildlife Hunts. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.10(6), pp.857-868.

[HS04] Hakeem, A., and Shah, M. 2004. Ontology and Taxonomy Collaborated Framework for Meeting Classification. In *Proc. of International Conference on Pattern Recognition*, pp.219-222.

[HSS04] Hakeem, A., Sheikh, Y., and Shah, M. 2004. CASE^E: A Hierarchical Event Representation for the Analysis of Videos. In *Proc. of American Association of Artificial Intelligence (AAAI)*, pp.263-268.

[HS05] Hakeem, A., and Shah, M. 2005. Multi-Agent Event Learning, Detection, and Representation in Videos. Accepted for publication in *Proc. of American Association of Artificial Intelligence (AAAI)*.

[HL04] Harville, M., and Li, D. 2004. Fast, Integrated Person Tracking and Activity Recognition with Plan-View Templates from Single Stereo Camera. In *Proc. of Computer Vision and Pattern Recognition*, pp.398-405.

[HNB04] Hongeng, S., Nevatia, R, and Bremond, F. 2004. Video-Based Event Recognition: Activity Representation and Probabilistic Recognition Methods. In *Computer Vision and Image Understanding*, vol.96(2), pp.129-162.

[IB99] Intille, S., and Bobick, A. F. 1999. A Framework for Recognizing Multi-agent Action from Visual Evidence. In *Proc. of American Association of Artificial Intelligence (AAAI)*, pp.518-525.

[IB00] Ivanov Y. A., and Bobick A. F. 2000. Recognition of Visual Activities and Interactions by Stochastic Parsing. In *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol.22, pp.852-872.

[JSS02] Javed, O., Shafique, K., and Shah., M. 2002. A Hierarchical Approach to Robust Background Subtraction Using Color and Gradient Information. In *Workshop on Motion and Video Computing*, pp.22-27.

[JS02] Javed, O., and Shah, M. 2002. Tracking and Object Classification for Automated Surveillance. In *Proc. of European Conference on Computer Vision*, pp.343-357.

[JSC04] Javed, O., Shah, M., and Comaniciu, D. 2004. A Probabilistic Framework for Object Recognition in Video. In *Proc. of International Conference on Image Processing*.

[KS97] Katayama, N., and Satoh, S. 1997. The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries. In *Proc. of ACM SIGMOD*, pp.369-380.

[KV78] W. Kintsch, and T. Van Dijk. 1978. Toward a model of text comprehension and production. In *Psychological Review*, vol.85(5), pp.363-394.

[K88] W. Kintsch. 1988. The role of knowledge in discourse comprehension: a construction-integration model. In *Psychological Review*, vol.95, pp.163-182.

[KTF01] Kojima, A., Tamura, T. and Fukunaga, K. 2001. Natural Language Description of Human Activities from Video Images Based on Concept Hierarchy Actions. In *International Journal of Computer Vision*, vol.50, pp.171-184.

[KHN91] Koller, D., Heinze, N., and Nagel, H. H. 1991. Algorithmic Characterization of Vehicle Trajectories from Image Sequences by Motion Verbs. In *Proc. of Computer Vision and Pattern Recognition*, pp.90-95.

[MTB03] Maillot, N., Thonnat, M., and Boucher, A. 2003. Towards Ontology Based Cognitive Vision. In *Proc. of International Conference of Vision Systems*, pp.44-53.

[MJ98] Mann, R., Jepson, A. 1998. Towards the Computational Perception of Action. In *Proc. of Computer Vision and Pattern Recognition*, pp.794-799.

[MCB01] Medioni, G., Cohen, I. , Brmond, F., Hongeng S., and Nevatia, R. 2001. Event Detection and Analysis from Video Streams. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.23(8), pp. 873-889.

[MES03] Minnen, D., Essa, I., and Starner, T. 2003. Expectation Grammars: Leveraging High-Level Expectations for Activity Recognition. In *Proc. of Computer Vision and Pattern Recognition*, pp.626-632.

[MEH99] Moore, D. J., Essa, I. A., and Hayes, M. H. 1999. Exploiting Human Actions and Object Context for Recognition Tasks. In *Proc. of International Conference of Computer Vision*, pp.80-86.

[NKHR00] Naphade, M., Kozintsev, I., Huang, T., and Ramchandran, K. 2000. A factor graph framework for semantic indexing and retrieval in video. In *Workshop on Content Based Access to Image and Video Libraries, CVPR*, pp.35-39.

[NH01] Naphade, M. and Huang, T. 2001. A Probabilistic Framework for Semantic Video Indexing, Filtering, and Retrieval. In *IEEE Transactions on Multimedia*, pp.141-151.

[NN05] Natarajan, P. and Nevatia, R. 2005. EDF: A framework for Semantic Annotation of Video. In *Proc. of International Conference of Computer Vision*, pp.1876-1886.

[N89] Neumann, B. 1989. Natural Language Description of Time Varying Scenes. In Waltz, D. eds., *Semantic Structures: Advances in Natural Language Processing*, pp.167-206, Hillsdale, NJ: Lawrence Erlbaum Associates.

[NZH03] Nevatia, R., Zhao, T., and Hongeng, S. 2003. Hierarchical Language-Based Representation of Events in Video Streams. In *IEEE Workshop on Event Mining*, Madison, WI.

[NHB04] Nevatia, R., Hobbs, J., and Bolles, B. 2004. An Ontology for Video Event Representation. In *IEEE Workshop on Event Detection and Recognition.*

[NBV03] Nguyen, N. T., Bui, H. H., Venkatesh, S. and West, G. 2003. Recognising and Monitoring High-Level Behaviours in Complex Spatial Environments. In *Proc. of Computer Vision and Pattern Recognition*, pp.620-625.

[NMTM00] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. In *Maching Learning*, vol.30(2), pp.103-134.

[ORP99] Oliver, N., Rosario, and B., Pentland, A. 1999. A Bayesian Computer Vision System for Modelling Human Interaction. In *Proc. of International Conference of Computer Vision Systems*, pp.255-272.

[OLW02] Ozer, B., Lv, T., and Wolf, W. 2002. A Bottom-up Approach for Activity Recognition in Smart Rooms. In *Proc. of International Conference on Multimedia Expo*, pp.917-920.

[PP99] C. Papageorgiou and T. Poggio. 1999. Trainable pedestrian detections. In *International Conference on Image Processing*.

[PB98] Pinhanez, C., and Bobick, A. 1998. Human Action Detection Using PNF Propagation of Temporal Constraints. In *Proc. of Computer Vision and Pattern Recognition*, pp.898-904.

[PN93] Polana, R., and Nelson, R. 1993. Detecting Activities. In *Proc. of Computer Vision and Pattern Recognition*, pp.2-7.

[PN97] Polana, R., and Nelson, R. 1997. Detection and recognition of periodic, non-rigid motion. In *International Journal of Computer Vision*, vol.23(3), pp.261-282.

[RYS02] Rao, C., Yilmaz, A., and Shah, M. 2002. View-Invariant Representation and Recognition of Actions. In *International Journal of Computer Vision*, vol.50, pp.203-226.

[RHC99] Rui, Y., Huang, T., and Chang, S. 1999. Image Retrieval: Current Techniques, Promising Directions And Open Issues . In *Journal of Visual Communication and Image Representation*, vol.10(4), pp.39-62.

[RA00] Rui, Y., Anandan, P. 2000. Segmenting Visual Actions Based on Spatio-Temporal Motion Patterns. In *Proc. of Computer Vision and Pattern Recognition*, pp.111-118.

[SEV93] R. Saavedra, P. Earley, and L. Van Dyne. 1993, Complex Interdependence in Task-Performing Groups. In *Journal of Applied Psychology*, vol.78, pp.61-72.

[SK00] H. Schneiderman and T. Kanade. 2000. A statistical method for 3d object detection applied to faces and cars. In *International Conference on Computer Vision and Pattern Recognition*.

[SS05] Shafique, K., and Shah, M. 2005. A Noniterative Greedy Algorithm for Multiframe Point Correspondence. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.27(1), pp.51-65.

[SM00] Shi, J., and Malik, J. 2000. Normalized Cuts and Image Segmentation. In *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol.22(8), pp.888-905.

[SHMBE04] Shi, Y., Huang, Y., Minnen, D., Bobick, A., and Essa, I. 2004. Propagation Networks for recognition of partially ordered sequential action. In *Proc. of Computer Vision and Pattern Recognition*, pp.862-870.

[S00] Siskind, J. M. 2000. Visual Event Classification via Force Dynamics. In *Proc. of American Association of Artificial Intelligence (AAAI)*, pp.149-155, Menlo Park, CA:AAAI Press.

[SSL04] Smith, P., Shah, M., and Lobo, N. 2004. Integrating and Employing Multiple Levels of Zoom for Activity Recognition. In *Proc. of Computer Vision and Pattern Recognition*, pp.928-935.

[SG00] Staffer, C., and Grimson., W. E. L. 2000. Learning Patterns of Actvity Using Real-Time Tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.22(8), pp.747-757.

[TSKK00] P. Tsai, M. Shah, K. Keiter, and T. Kasparis. 1994. Cyclic motion detection for motion based recognition. In *Pattern Recognition*.

[VCC03] Vaswani, N., Chowdhury, A. R., and Chellapa, R. 2003. Activity Recognition Using the Dynamics of the Configuration of Interacting Objects. In *Proc. of Computer Vision and Pattern Recognition*, pp.633-642.

[WB96] Wang, Z., and Ben-Arie, J. 1996. Optimal Ramp Edge Detection Using Expansion Matching. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.18(11), pp.1092-1098.

[WAD97] Wren, C. R., Azarbayejani, A., Darrel, T., and Pentland, A. P. 1997. Pfinder: Real-Time Tracking of the Human Body. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.19(7), pp.780-785.

[XMZ03] Xu, G., Ma, Y., Zhang, H., and Yang, S. 2003. A HMM Based Semantic Analysis Framework for Sports Game Event Detection. In *Proc. of International Conference on Image Processing*, pp.25-28.

[YB98] Yacoob, Y., and Black, M. 1998. Parameterized Modeling and Recognition of Activities. In *Proc. of International Conference on Computer Vision*, pp.120-127.

[YLS04] A. Yilmaz, X. Li, and M. Shah. 2004. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.26(11).

[YJS06] Yilmaz, A., Javed, O., and Shah, M. 2006. Object tracking: A survey. In *ACM Compuing Survey*. vol.38(4)

[ZI01] Zelnik-Manor, L., and Irani, M. 2001. Event Based Analysis of Video. In *Proc. of Computer Vision and Pattern Recognition*, pp.123-130.

[ZSV04] Zhong, H., Shi, J., and Visontai, M. 2004. Detecting Unusual Activity in Video. In *Proc. of Computer Vision and Pattern Recognition*, pp.819-826.

[CAVIAR03] CAVIAR: Context Aware Vision using Image-based Active Recognition. More details at *http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/*.

[CYC01] CYC: *http://www.cyc.com*.

[MARVEL] MARVEL: Multimedia Analysis and Retrieval System. More details at *http://www.research.ibm.com/marvel/details.html*.

[PEGASUS] PEGASUS: *http://www.cs.ucf.edu/ vision/projects/pegasus/pegasus.html*.

[PETS01] PETS: Performance Evaluation for Tracking and Surveillance. More details at *http://www.cvg.cs.rdg.ac.uk/PETS-ICVS/pets-icvs-db.html*.

[VACE01] VACE: Video Analysis and Content Exploitation. More details at *http://www.informedia.cs.cmu.edu/arda/vaceI_integrate.html*.