UNDERSTANDING EVENTS IN VIDEO

GAL LAVEE
UNDERSTANDING EVENTS IN VIDEO

RESEARCH THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

GAL LAVEE

SUBMITTED TO THE SENATE OF THE TECHNION — ISRAEL INSTITUTE OF TECHNOLOGY
ELUL, 5772  HAIFA  SEPTEMBER, 2012
ACKNOWLEDGMENTS

This manuscript represents the end of a long and trying personal journey full of emotional highs and lows for me. In the end, as always in life, it is the interactions with the many wonderful people I have met throughout this period that got me through the tough parts, and that I will take with me moving forward. In this short text I would like to begin to express my gratitude to these individuals who have blessed my life, but always remain in the knowledge that I am forever indebted to them. I sincerely hope that I can somehow repay some small fraction of their contribution to my life. First and foremost, to my parents Ayelet and Zeev Lavee, who always offered their unconditional love and support, even as they bounded across the globe. To my dear sister Dagani, who at first supported me from afar and eventually became my roommate and best friend. To my darling girlfriend, Hannah, who stood by me through every imaginable hardship, and always helped to see the positive in every situation. To my one and only office-mate, Kira, who gave me endless encouragement, career advice and inspiration. Finally, to the countless colleagues, professors and friends, both in Israel and abroad with whom I was fortunate to share my time in Haifa. Thank you all for the many blessings you have brought into my life, big and small. I could not have done it without you.

THE GENEROUS FINANCIAL HELP OF THE TECHNION IS GRATEFULLY ACKNOWLEDGED
To my parents,
Ayelet and Zeev Lavee
Portions of this Thesis have been previously published in the following scientific journal articles:

<table>
<thead>
<tr>
<th>Article Name</th>
<th>Publication Name</th>
<th>Impact Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video Event Modeling and Recognition using Marking Analysis in Generalized Stochastic Petri Nets</td>
<td>IEEE Transactions on Circuits and Systems for Video Technology</td>
<td>2.55</td>
</tr>
<tr>
<td>Propagating Certainty in Petri Nets for Activity Recognition</td>
<td>IEEE Transactions on Circuits and Systems for Video Technology</td>
<td>2.55</td>
</tr>
<tr>
<td>Modeling Context using Petri Nets for Activity Recognition</td>
<td>Pattern Recognition (Submitted)</td>
<td>2.292</td>
</tr>
</tbody>
</table>

Portions of this work have also been presented at the following conferences:

<table>
<thead>
<tr>
<th>Article Name</th>
<th>Conference Name</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building Petri Nets from Video Event Ontologies</td>
<td>International Symposium on Visual Computing</td>
<td>2009</td>
</tr>
<tr>
<td>Video Event Ontologies</td>
<td>International Symposium on Visual Computing</td>
<td>2010</td>
</tr>
<tr>
<td>Propagating Uncertainty in Petri Nets for Activity Recognition</td>
<td>International Symposium on Visual Computing</td>
<td>2010</td>
</tr>
</tbody>
</table>
## Contents

Abstract .................................................. 1

1 Introduction ........................................... 3

2 What is an Event? ....................................... 7

3 Related Work ............................................. 13
   3.1 Abstraction .......................................... 14
      3.1.1 Pixel-Based Abstraction ...................... 15
      3.1.2 Object-based Abstraction ....................... 17
      3.1.3 Logic-based Abstraction ....................... 19
   3.2 Event Modeling ..................................... 20
   3.3 Pattern Recognition Methods for Event Recognition .................................. 23
   3.4 State Event Models ................................ 26
      3.4.1 Finite State Machines ......................... 27
      3.4.2 Bayesian Networks ............................ 29
      3.4.3 Hidden Markov Models ......................... 32
      3.4.4 Dynamic Bayesian Networks .................... 37
      3.4.5 Conditional Random Fields ..................... 38
   3.5 Semantic Event Models ............................ 39
      3.5.1 Grammars ....................................... 40
      3.5.2 Petri Nets ..................................... 43
      3.5.3 Constraint Satisfaction ......................... 46
      3.5.4 Logic Approaches ................................ 48

4 The Petri Net Formalism ................................ 49
   4.1 Graphical Representation .......................... 49
   4.2 Marking Analysis ................................... 51

5 The Particle Filter .................................... 53
6 Modeling Events with Generalized Stochastic Petri Nets
   6.1 Video Event Representation in SERF-PN . . . . . . . . . . . . . . 58
   6.2 Learning the System Parameters . . . . . . . . . . . . . . . . . . 59
      6.2.1 Timing Parameters . . . . . . . . . . . . . . . . . . . . . 59
      6.2.2 Marking Transition Probabilities . . . . . . . . . . . . . 59
   6.3 Constructing a PN model: Illustrative examples . . . . . . . . . 60
      6.3.1 Security Check Scenario . . . . . . . . . . . . . . . . . . 63
      6.3.2 Traffic Intersection Scenario . . . . . . . . . . . . . . . 64
   6.4 Complexity Analysis . . . . . . . . . . . . . . . . . . . . . . . . 67

7 Surveillance using SERF-PN
   7.1 Generating Video Annotation . . . . . . . . . . . . . . . . . . . 73
   7.2 Event Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
   7.3 Video Event Analysis . . . . . . . . . . . . . . . . . . . . . . . . 75
   7.4 Evaluating SERF-PN . . . . . . . . . . . . . . . . . . . . . . . . . 76
      7.4.1 Security Check . . . . . . . . . . . . . . . . . . . . . . . 77
      7.4.2 Traffic Intersection . . . . . . . . . . . . . . . . . . . . . 80
      7.4.3 CAVIAR dataset . . . . . . . . . . . . . . . . . . . . . . . 83

8 Coping with Uncertainty In a Petri Net Event Model
   8.1 Modeling activities with Petri Nets . . . . . . . . . . . . . . . . 94
   8.2 Defining a Bayesian Recursive Filter Using a Petri Net . . . . . . 95
      8.2.1 Preliminaries . . . . . . . . . . . . . . . . . . . . . . . . 95
      8.2.2 The State Space . . . . . . . . . . . . . . . . . . . . . . . 95
      8.2.3 The Observation Space . . . . . . . . . . . . . . . . . . . 96
      8.2.4 Conditional Independence . . . . . . . . . . . . . . . . . 96
      8.2.5 The Dynamic model . . . . . . . . . . . . . . . . . . . . . 97
      8.2.6 The Measurement Model . . . . . . . . . . . . . . . . . . . 98
      8.2.7 The Prior Probability . . . . . . . . . . . . . . . . . . . . 99
   8.3 Propagating Certainty . . . . . . . . . . . . . . . . . . . . . . . . 99
      8.3.1 Initialization . . . . . . . . . . . . . . . . . . . . . . . . 99
      8.3.2 Update . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100
   8.4 An Example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100
      8.4.1 Constructing the Bayes Recursive Filter Components . . . 101
      8.4.2 Propagating Certainty . . . . . . . . . . . . . . . . . . . . 103

9 Factoring Context
   9.1 Constructing the Petri Net Activity Model . . . . . . . . . . . . . 106
      9.1.1 Constructing Context Fragments . . . . . . . . . . . . . . 106
      9.1.2 Constructing Activity Fragments . . . . . . . . . . . . . . 106
      9.1.3 The Template Fragments . . . . . . . . . . . . . . . . . . . 107
List of Figures

3.1 Bird’s Eye View of the Video Event Understanding Domain. A video event understanding process takes an image sequence as input and abstracts it into meaningful units. The result of the abstraction is used by the event model to determine if an event of interest has occurred. Output of a video event understanding process may be a decision on whether a particular event has occurred or a summary of events in the input sequence. ............................. 14

3.2 A video sequence containing an event (shown as a key frame in the first column) is abstracted as a single Motion History Image in which the pixel intensity at each pixel represents the amount of motion observed at that spatial location during the video sequence. This is an example of Pixel-Based abstraction (inspired by (Bobick and Davis, 2001)) ................................................................. 17

3.3 This figure visualizes Object-Based Abstraction. In this type of abstraction scheme objects are located and tracked and the video sequence is abstracted as properties of these object. In (a) a single person is tracked and a bounding box along with a trajectory from its previous location (visualized by a line) are used to abstract the sequence at each frame. In (b) the same scheme is used for two objects (a person and a cat). (inspired by (Hongeng and Nevatia, 2001)) ................................................................. 18

3.4 FSM event model used on the "car avoids checkpoint" event a la (Medioni et al., 2001) ................................................................. 28

3.5 A naive Bayesian Network depicting an atomic event,"pedestrian crossing street" ................................................................. 30

3.6 Factorization of the state and observation space. Shaded nodes indicate observed variables. ......................................................... 35

3.7 HMM extensions to capture long-term temporal dependency ............................. 35

3.8 "Hybrid" HMM models capturing the intrinsic properties of video events. ................................................................. 36

3.9 A CRF is a discriminative undirected graphical model with structure inspired by the HMM ......................................................... 38
3.10 Example of a Object PN Model for the Left Luggage Domain.

3.11 Temporal Relations are easily modelled using Petri Net fragments. See (Borzin, Rivlin, and Rudzsky, 2007)

3.12 Example of a Plan PN model for the "Car is Parking" event.

4.1 An Example Petri Net

4.2 Reachability graph example.

5.1 An illustration of the SIS particle filtering algorithm. The algorithm proceeds by sampling from an approximation of the posterior in the prediction step. These samples are then reweighted according to their consistency with the observation.

6.1 Security Check Model (see table 6.1 for more details)

6.2 Traffic Intersection Model (see Table 6.2 for more details)

7.1 Surveillance system components

7.2 Keyframes from a typical real video sequence used in the Security Check Experiment. Ellipses denoting object locations generated by the tracker are shown.

7.3 Keyframes from a typical animation clip sequence used in the Security Check Experiment. Hollow boxes are scene objects. The solid blue gray box indicates the "Guard Post" zone.

7.4 Reduced marking graph obtained by dynamic marking analysis.

7.5 Keyframes from a typical real video sequence used in the Traffic Intersection Experiment. Ellipses denoting object locations generated by the tracker are shown.

7.6 Keyframes from a typical animation clip sequence used in the Traffic Intersection Experiment. Filled circles model cars in the scene. The solid black polygon marks the "Intersection" zone.

7.7 CAVIAR Situation Petri Net Model. See Table 7.5 for specification of transition condition.

7.8 CAVIAR Context Petri Net Model. See Table 7.6 for specification of transition condition.

8.1 Activity Petri Net Describing Bank Attack

8.2 The conditional independence relations between the state and observation variables

8.3 Activity Petri Net Describing Bank Attack

9.1 The template fragments used for the various temporal relations.
9.2 An example of chaining relations. Here we use the markings of the nested inner relation PN fragment as context for the PN relation fragment of the outer relation.

9.3 The conditional independence relations between the context, activity and observation variables. The activity state is independent of the observation given the context.

9.4 The conditional independence relations between the context and observation variables. The observation, \( i y_t \), is independent of the context factor marking, \( i x_t(m) \), given the relevant events at time \( t \), \( i x_t(e) \).

9.5 The conditional independence relations between the context and activity variables. The context at time \( t \) (denoted \( C_t \)) is a factor in determining which events occur at time \( t \) in each activity factor.

9.6 Context Fragment for the Bank Surveillance Dataset

9.7 Temporal Relation Fragment. Transition label letters correspond to place nodes in context fragment (Figure 9.6). Transition Label sub-indexes correspond to the object instantiation (1-cashier, 2-visitor).

10.1 Flowchart of System

10.2 The output of Module #1 (Tracking) and Module #2 (Event Recognition)

10.3 Example Sequence from synthetic Bank Dataset. A Bank Attack activity is taking place. Sequence should be interpreted left to right, top to bottom.

10.4 Example Sequence from ETISEO Building Sequence Dataset. A Depart By Car activity is taking place. Frames are taken from four different camera angles. Sequence should be interpreted left to right, top to bottom.

10.5 Example Sequence from Technion Parking Lot Dataset. An Arrive By Car activity is occurring simultaneously with a Car Theft activity taking place. Sequence should be interpreted left to right, top to bottom.

10.6 ROC Curve for the Synthetic Bank Dataset

10.7 ROC Curve for the ETISEO Building Entrance Dataset

10.8 ROC Curve for the Technion Parking Lot Dataset (real tracking)

10.9 ROC Curve for the Technion Parking Lot Dataset ("ideal" tracking)

10.10 ROC Curve Comparing Recognition Performance for the Bank Dataset including FPFPN

10.11 ROC Curve Comparing Recognition Performance for the ETISEO Dataset including FPFPN
List of Tables

2.1 Event Terminology .................................................. 10

3.1 An attribute grammar describing events in the parking lot scenario similar to (Joo and Chellappa, 2006). Terminals (lower-case letters) correspond to atomic sub-events and non-terminals (upper-case letters) correspond to events and composite sub-events. A location attribute(loc) is associated with each terminal. The notation \( X_i \) refers to the \( i \)th term on the right hand side of the production rule. Predefined semantic locations are defined by keywords such as BldgEnt (i.e. the entrance to the building) and FOV (i.e field of view) ........................................... 42

6.1 Elaboration on the model structure pictured in figure 6.1. Each transition is given a meaningful semantic name and associated with a condition on the properties of its input tokens. Token\( (p_x) \) denotes a token whose origin is in place \( p_x \). In situations when a transition has only one input place node, this notation is dropped and we simply write Token. In the case of multiple tokens from the same place node, we denote these Token1\( (p_x) \), Token2\( (p_x) \). Token.attributex denotes the value of attributex in the particular input token. In this example we use the attributes ”loc” (token location) and ”Appearance”, (which can take on the values {”appear”, ”visible”, ”disappear”}). See text for details of this construction. ........................................... 62

6.2 Elaboration on the model structure pictured in figure 6.2. See figure 6.1 caption for explanation of notation. In addition to the “loc” and “Appearance” attributes used in the previous example, this example also uses the “orientation” attribute which gives the direction of the object’s velocity vector. See text for details of this construction. ........................................... 66

6.3 Notation used in formal description of algorithm. \( 2^X \) denotes the powerset of set \( X \) ................. 68
6.4 Algorithm 1: detecting a particular event in a video sequence . . . 69
6.5 Algorithm 1 continued .................................................. 71

7.1 A semantic summary corresponding to the video clip in Figure 7.2 generated by the video event recognition component of SERF-PN. Events mentioned in summary correspond to transitions in the model shown in Figure 6.1 .................................................. 78
7.2 The recall and precision for the Security Check Model on real video clips and corresponding animations. The events considered were "Security Check Too Long" and "Visitor not Checked" ............... 79
7.3 A semantic summary corresponding to the video clip in Figure 7.5 generated by the video event recognition component of SERF-PN. Events mentioned in summary correspond to transitions in the model shown in Figure 6.2 .................................................. 81
7.4 The recall and precision for the Traffic Intersection Model applied real video clips and corresponding animations for the event "Safe Interaction Occurred" .................................................. 83
7.5 Each transition in the situation model structure pictured in figure 7.7 is associated with a condition on the properties of its input tokens ................................................................. 85
7.6 Each transition in the context model structure pictured in figure 7.8 is associated with a condition on the properties of its input tokens ................................................................. 86
7.7 Situation frame by frame classification ...................................... 89
7.8 Context frame by frame classification (using situation ground truth) 90
7.9 Context "last frame" classification (using situation ground truth) 90
7.10 Context frame by frame classification (using PN derived situation information) .................................................. 90
7.11 Context "last frame" classification (using PN derived situation information) .................................................. 91
7.12 Comparison ................................................................. 91

8.1 Notation Summary ........................................................... 96
8.2 The reachable states of the activity depicted in Figure 8.3 .................. 101

9.1 The reachable states of the PN fragments depicted in Figures 9.6, 9.7 ................................................................. 120

10.1 Area Under The Curve ........................................................ 141
10.2 Area Under The Curve including FPFPN .................................. 141
Abstract

Video events are those high-level semantic concepts that humans perceive when observing a video sequence. Understanding these concepts is the highest level task in computer vision. It relies on sufficient solutions to many lower-level tasks such as edge detection, optical flow estimation, object recognition, object classification and tracking. The maturity of many solutions to these low-level problems has spurred additional interest in utilizing them for higher level tasks such as video event understanding.

In this thesis we map the diverse literature in the research domain of video event understanding. First we construct a taxonomy of this research domain and apply this taxonomy to categorize many leading works. The terminology of the video event understanding research domain is often confusing and ambiguous. Many terms such as "events", "actions", "activities", and "behaviors" are often used in different ways across the literature. In this thesis we provide an in-depth discussion of this ambiguity and suggest a terminology, which we then apply throughout the remainder of the thesis, that allows unification and comparison of the various works in this research domain.

Our contribution to the research domain of video event understanding focuses on events defined by complex temporal relationships among their sub-event building blocks. We explore the representative power of the Petri Net formalism to model these events. Our early work describes an approach for modeling scenes where Petri Net place nodes represent states scene objects may take on. Petri Net transition nodes represent changes in the properties of scene objects. Petri Net tokens in this model represent scene objects. Recognition of events is achieved deterministically by tracking the properties of scene objects and propagating their representative tokens throughout the Petri Net model of the scene. This approach does allow for some variance in the duration of sub-events via the use of stochastic timed transitions.

Our later work focused on constructing a Petri Net model of an event that is robust to the various kinds of uncertainty inherent to surveillance video data. In this approach a Petri Net modeling the temporal constraints of the event is constructed by a domain expert. The Petri Net is laid out as a “plan” where token(s) are advanced from an initial place node to a final “recognized” place.
node as external sub-events are observed in a manner that is consistent with the definition of the event. In order to deal with the fact that sub-events are, in general, only observed up to a particular certainty, we define a transformation from the Petri Net definition into a probabilistic model. Within this probabilistic model, well-studied approaches, such as the Particle Filter algorithm, afford elegant reasoning under uncertainty. Furthermore, online reasoning, which is required by many of our motivating scenarios, is also enabled. In many areas of the video event understanding domain, particularly surveillance applications, we are often interested in differentiating between similar events that differ only by the configuration of their constituent sub-events. Since these events exist within the same scene, they are limited by the same physical (or context) constraints. These context constraints are independent of the constraints that define the temporal ordering of sub-events. Our most recent work has focused on applying this intuition and constructing event models which separately model context and non-context constraints. This separation, affords simpler event models, reduces the complexity of the probabilistic inference, and ultimately improves both recognition performance and efficiency.

One main contribution of this thesis to the research domain of video event understanding is the representation scheme that decouples context constraints from the temporal constraints that define the event. Another area of contribution are the recognition algorithms proposed which are constructed on top of the Petri Net representation of the event domain. These algorithms are generalized to be able to cope with the uncertainty inherent in video and afford elegant probabilistic reasoning which can be updated as new information becomes available.
Chapter 1

Introduction

Video events are those high-level semantic concepts that humans perceive when observing a video sequence. Understanding these concepts and recognizing them as they occur is the highest level task in computer vision. Solutions to this problem offer the promise of intelligent systems outfitted with inexpensive cameras enabling such applications as: active intelligent surveillance, summarization and indexing of video data, unobtrusive homecare for the elderly, and hands-free human-computer interaction. Clearly, efficient event understanding algorithms with good accuracy that are not susceptible to human failings such as slowness, subjectivity, and fatigue (classifying hours of surveillance video can be tedious), would be of great use to society.

Efficient and accurate solutions for the video event understanding task rely on sufficient solutions to lower-level tasks such as edge detection, optical flow estimation, object recognition, and object classification and tracking. Recent advances in the development of solutions to these to these low-level problems has spurred additional interest in applying these solutions for higher level tasks such as video event understanding.

The interest in video event understanding is further exhibited by the large amount of recent research projects approved in this domain including: CARETAKER (CAR, 2008), ETISEO (Nghiem et al., 2007), AVITRACK (AVI, 2006), ADVISOR (ADV, 2003), BEWARE (BEW, 2011), ICONS (ICO, 2003), VSAM (VSA, 2000), VIRAT (Oh et al., 2011), and many others.

This thesis focuses on the representation and recognition of a particular type of events we call composite events (see Chapter 2 for a more in depth discussion of the different types of events). Composite events are characterized as a set of atomic sub-events in a particular temporal configuration. This configuration is often more complex then a simple sequential configuration and can involve temporal interval relations, partial ordering, and concurrency. A bank attack in a bank surveillance application (see the description of our experiments in Chapter
10) is one example of a composite event. It consists of two scene objects, a cashier and a visitor, moving throughout the scene. Each time a scene object appears, disappears, or enters a scene zone a sub-event is generated. These sub-events are atomic and restricted to a continuous temporal interval. It is the temporal configuration of these sub-events that determines if a bank attack is occurring (the example of a bank attack event is used in chapters 8 and 9 to make our approach more concrete). Composite events are challenging to recognize as, unlike simpler atomic events, they are not characterized by appearance features such as color, edge orientation or optical flow. Nor are they characterized by object locations or trajectories. Furthermore, the particular composite events we are interested in are often uncommon (e.g. a bank attack), resulting in little, if any, available training data. The complexity of these events along with lack of training data limits the usefulness of statistical approaches for classification of time-series data, the most well known of which is the Hidden Markov Model (HMM), which have been successfully used to recognize atomic events, which can be described as a sequential ordering of appearance based primitives.

Due to the inherent uncertainty in video data, the sub-event components of our events of interest are themselves only observed up to a certainty value. One reason for this is uncertainty in the observation caused by noisy signals, ghost objects detections, and false merging/splitting of object tracks. Another reason for this uncertainty is the fuzziness of human notions which are used to define events (e.g. being inside a zone.) Furthermore, sub-event recognition approaches are often heterogenous, and each calculates the certainty associated with the sub-event in an ad hoc and often domain dependant way. For example, the certainty associated with the sub-event Cashier Moving Fast is calculated in a different way then the certainty associated with the sub-event Cashier inside Behind Counter zone.

Currently, there exists a large gap between the most successful automatic approaches for recognizing the types of events described above (Vu, 2004; Shi, Bobick, and Essa, 2006) and the ability of humans to recognize these events as they occur in video sequences. We conjecture that the reason for this gap may be that humans are equipped with a great deal of semantic knowledge about what constitutes a particular event within an event domain. While it is difficult problem to encode all human knowledge, with a powerful representational formalism we contend that it is possible to distill the knowledge needed to correctly recognize and classify events in a particular event domain, particularly simple domains such as those found in surveillance applications.

Our work explores the use of the Petri Net (PN) formalism to represent the semantic composition of an event. Petri Nets are robust to modeling partial ordering and concurrency relationships that exist in many of the events we are
interested in recognizing. We explore two main approaches to modeling events using the Petri Net formalism (Chapter 3 offers more details on these approaches.) In the Object-based PN approach a Petri Net model of the entire scene is constructed. In this approach PN allows modeling knowledge about an event domain in terms of semantic object properties and relationships. In the Plan-based PN approach a Petri Net modeling the constraints of the event is constructed. The Petri Net model is laid out as a “plan” left to right. An event is recognized when a token reaches the left-most “sink” place node of the plan.

One major contribution of our work is an event representation which decouples the representation of physical scene constraints from the temporal constraints defining the composite event. This separation simplifies the construction of the event model, and allows the estimation of the context state to be done independently of the event state. Another contribution of this thesis are the recognition algorithms that make use of the Petri Net representation of an event. To our knowledge these recognition algorithms are the first to be defined in a general setting where sub-events are only observed up to a certainty value. In this setting the recognition is agnostic to how the certainty associated with the sub-events is computed. Thus, we have defined a general approach for coping with any event that can be specified as Petri Net, independent of the event domain. The recognition algorithms propagate the uncalibrated certainty values of the recognized sub-events up to the event level and output an event recognition with its own associated certainty value. The algorithms also afford updating of the event recognition certainty as new information becomes available. This is an important quality for active applications, such as an intelligent surveillance system which fires an alert when an unauthorized event (e.g. a bank attack) is recognized. In a comparison of our event recognition approach with competing approaches, experiments show an improvement in recognition performance measured as a precision/recall tradeoff against a labeled ground truth across several varied datasets. The efficiency of our algorithms, which are based on sampling, is measured using the number of samples required to reach the best performance. As the time complexity of our approach is a function of the number of samples, reducing the number of samples needed to achieve optimal recognition results is tantamount to improving the efficiency of the recognition.

The thesis is organized as follows: In the initial chapters we map the diverse research domain of video event understanding. In Chapter 2 we unify the terminology of the field. Chapter 3 defines a taxonomy of the field, organizes the various related works into this taxonomy, and describes the varied techniques used in the literature. The remainder of the thesis describes our contribution to the field of video event understanding. We describe several approaches to representation and recognition of composite multi-threaded object-based events (see Chapter 2 for an explanation of this terminology) sometimes referred to in the
literature as activities or scenarios. Our approaches utilize the Petri Net formalism to model the complex temporal structure that often defines these types of events. Chapter 4 gives the background on the Petri Net formalism. Petri Nets can be used to model events in different ways. In Chapter 6 we describe our early work efforts to construct a Petri Net which models the entire state of the scene. Some subset of this space represents the recognition of our event of interest. This framework is deterministic in nature, but affords modeling of stochastic variance in the duration of sub-events. Chapter 7 describes our experiments for evaluating the merits of this approach across several datasets.

In Chapter 8 we describe work which applies a Petri Net to describe the semantic structure of the event we are interested in recognizing. In this chapter we consider a general event recognition problem where sub-events are observed only up to a particular certainty value and that these certainties are ad-hoc in nature and not necessarily calibrated with one another. To cope with this uncertainty we construct a probabilistic model based on the Petri Net specification of our event. This construction enables the online reasoning under uncertainty, which we achieve using the Particle filter algorithm, that is required by the motivation outlined above. Chapter 5 gives the relevant background on the Particle Filter approach.

Finally, in Chapter 9 we discuss our most recent work which combines Petri Net model of the scene (which we call context) with a Petri-Net model of the event. We show experimentally that this modeling reduces the complexity of the probabilistic model, and improves both performance and efficiency of our recognition algorithm. Chapter 10 describes our experiments evaluating the performance of the approaches described in Chapters 8 and 9 against competing state of the art approaches to event understanding across a number of heterogeneous datasets. Finally we conclude the thesis in Chapter 11.
Chapter 2
What is an Event?

In Chapter 3 we survey a large volume of works under the claim that they all pertain to video "events". However, if one were to look through all the papers in the reference section one would not necessarily encounter the word "event" in all of them. Other terms such as behavior, activity, action, scenario, gesture, primitive event and complex event are strewn about the literature. In this chapter our aim is to disambiguate these terms, relate them to one another and formulate a uniform terminology with which to discuss specific papers in subsequent sections. In the psychology discipline human understanding of their surroundings, including events occurring in their presence, have been studied in the research area dubbed perception. There are three major theories in this field: direct perception, indirect perception and internalized constraints. Direct perception is the theory put forth by James Gibson also known as ecological perception (Gibson, 1979). The main idea behind this theory is that the human system of perception makes use of inherent properties of its environment (such as gradients and optical flow) to reason about what it is observing. Gibson’s ideas included the notion of an optical array (how light is structured from a certain vantage point) rather than a retinal image being the starting point for perception processes. Another key idea in Gibsonian perception is that perception is geared towards meaning. That is, humans look for information that they are capable of interpreting rather than finding meaning in a general input of perception data. Events in direct perception are defined as disturbances in the perceiver’s optical array. Gibson lists possible event categories: object-displacements, collisions, non-rigid deformations, surface disruptions and surface deformations. Indirect perception, also know as Constructivist or Inference theory, states that rules of perception are learned from experience over time. That is, often observed co-occurrences of a set of stimuli will be grouped together and used in the future to perceive the same stimuli. This process over time forms a set of rules (also called perceptual heuristics) and eventually leads to attaching "meaning" to and
"understanding" stimulus. This theory implies that different interpretation rules of the world can be learned depending on the perceivers’ experience. The theory of indirect perception is described in (Rock, 1997). Events in this discipline are defined as the occurrence of an object in space-time.

The third theory of visual perception does not devote much attention to how the visual discrimination occurs. It does propose that there is a set of constraints governing stimuli that allow an efficient process to attaining the meaning. This theory is know as the theory of Internalized Constraints. The theory does not discuss how these constraints are arrived at (direct or indirect perception) but does argue they encode phenomena that is regularly observed over long periods (such as the laws of physics).

Newer theories of visual perception suggest that human intent, interaction with the objects and environment and scene context are also factors that may not be neglected in the building a model for human perception of events. In (Hecht, March 2000) the author evaluates the three theories of event perception discussed above and proposes that yet a more involved model need be proposed. One which "moves away from the camera model for visual perception".

Computer Vision literature has also attempted to formalize a definition for events on several occasion. In one notable paper (Nagel, 1988) Nagel characterized occurrences in image sequences in a simple hierarchy: change, event, verb and history. A “change” is defined as any deviation in the signal other than noise. This definition, is interestingly similar to Gibson’s notion of an event. Nagel’s “event” is a change that has been defined a priori (i.e. recognized). This is considered as a basic primitive for the construction of more complex activity. The term “verb” describes such a complex activity. Finally, the term “history” describes comprehensive description of activities and their relationships. In (Bobick, 1997) the authors set out to clarify the distinction between the terms movement, activity and action within the realm of video analysis work. The separation is made based on the knowledge required to classify each of these categories. A “Movement” is defined to be that event for which we can establish a clear cut template from visual cues. That is, its appearance remains similar independent of actor or other variables. An “activity” consists of a series of movements in a certain sequence. Recognition of the higher-level semantics of an activity requires more knowledge than is required to recognize a movement. Specifically, the knowledge of the representation of each of the movements as well as the order in which they are supposed to appear. An “action” is still higher on the semantics hierarchy and requires yet more knowledge than an activity. To recognize an action knowledge on the component activities as well as their relationships is required. The nature of the relationships of the activities composing the action is defined by a set of context-dependant primitives.

Unfortunately, non of these standards of terminology has been adopted uniformly.
Each new paper that comes out includes variations on these terms which it defines anew. We will review some of these terms in the following paragraphs using the common qualities to all these definitions. Namely that events have temporal qualities, they describe salient information and they have a hierarchical nature. Events in computer vision literature are, generally, given a temporal extent. (Hu et al., 2004) talks about “object behaviors”, defined as a classification of time varying feature data. This paper defines the “behavior understanding problem” as learning reference and matching techniques from training data in order to classify unlabeled examples. This is a prevalent definition. Aggarwal et al (Aggarwal and Cai, 1999) define “activity/behavior recognition” as a classification of a pattern derived from an image sequence. Also similarly, Cohn (Cohn et al., 2003b) defines “dynamic behavior” as a set of spatial states whose temporal relationships are defined by temporal logic. Hongeng and Nevatia (Hongeng and Nevatia, 2001) describe “simple events” as actions occurring in a linear time sequence. These examples illustrate the prevailing sense in the community that events occupy a certain interval of time rather than occurring instantaneously.

Another consensus on events throughout the literature is that they consider only salient(important) occurrences in the scene. Xiang and Gong (Xiang and Gong, 2006) defines “events” as localized collections of non-background pixels. These events are then composed together to form activities. (Bobick and Davis, 2001) also considers pixel-based features as sub-components for the detection of “gestures”.

Not consistent is the question of: what makes up an event? A popular alternative to pixel level events is object features and interactions. (Medioni et al., 2001) is an example of such an approach. This paper uses object trajectories to classify “behaviors”.

Another consistent property of events throughout the literature is that their hierarchical nature. This property of events is the largest cause of ambiguous terminology. The same terms are used differently to describe the granularities of the hierarchy. As an example of this we consider Howarth and Buxton (Buxton, 2002) which uses the term ”activities” to describe the basic unit of visual features. When arranged in a certain order these ”activities” from a larger semantic unit termed ”episodes”. By contrast, Medioni et al (Medioni et al., 2001) use the term ”activities” to describe higher-level concepts composed of mid and lower level semantic units. That is, the term ”activities” is akin to ”episodes” in Howarth and Buxton’s work(Howarth and Buxton, January 2000). Furthermore, in Bobick’s work (Bobick, 1997) (as previously mentioned) the term ”activities” pertains to an intermediary step in the semantic hierarchy. Hongeng and Nevatia (Hongeng and Nevatia, 2001) introduce the intuitive terms: ”simple event” and ”complex event” to model the hierarchical structure.

With these points in mind we propose the following terminology. An “event” is
defined as an occurrence of interest in a video scene. This term will be equivalent to other terms in the literature including “behavior” (Hu et al., 2004), “activity” (Howarth and Buxton, January 2000), “action” (Xiang and Gong, 2006) and “gesture” (Bobick and Davis, 2001). The term “sub-event” will be used to refer to the component parts of the event. This parallels “poses”, “actions”, “mobile event properties” and other terms, appearing elsewhere, which provide the building blocks of the occurrence of interest. Sub-events may also be composed of sub-events in a recursive definition. Complementary to sub-events we define the term “super-events” to be those semantic units formed by some combination of our events. This concept corresponds to Howarth and Buxton’s (Howarth and Buxton, January 2000) “episodes”, Mendioni and Nevatia’s “scenarios” (Medioni et al., 2001), “complex events” (Hongeng and Nevatia, 2001) and so forth. Again super-events may be defined recursively for multiple levels. We introduce prefix terms to define the temporal composition of the super-events. Borrowing from Nevatia (Hongeng and Nevatia, 2001) we will use the terms “multi-thread” and “single-thread” to describe sequential temporal relationships. Content prefix terms indicate whether the event relies on tracking and object feature information such as trajectories (Medioni et al., 2001) or uses pixel-level event information such as pixel change history (Xiang and Gong, 2006). That is, this prefix describes the base unit for composition of the event. Table 2 organizes our terminology. We call these prefix terminology as one or more prefixes can be used to describe each event.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Added Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub</td>
<td>Component of occurrence of interest</td>
</tr>
<tr>
<td>Super</td>
<td>Composed Of occurrence of interest</td>
</tr>
<tr>
<td>Atomic</td>
<td>Has no Sub-event composition</td>
</tr>
<tr>
<td>Composite</td>
<td>Has Sub-Event Composition</td>
</tr>
<tr>
<td>Pixel-Based</td>
<td>Has Pixel-Level Atomic Sub-Events</td>
</tr>
<tr>
<td>Object-Based</td>
<td>Has Object Feature Atomic Sub-Events</td>
</tr>
<tr>
<td>Single-Thread</td>
<td>Has Sequential Temporal relationships between Sub-Events</td>
</tr>
<tr>
<td>Multi-Thread</td>
<td>Has Non-Sequential Temporal relationships between Sub-Events</td>
</tr>
</tbody>
</table>

Table 2.1: Event Terminology

Our terminology and event definitions are somewhat separate from the psychology community’s definition which views event as changes in the perception such as the set defined by Gibson or object occurrences in space-time as is the view of the school of indirect perception. The main difference is that these “events” are instantaneous while our definition implies a certain duration. We can thus say that our definition subsumes the visual perception definition of
events. Our definition also departs from Bobick’s categorization based on the knowledge requirements in classifying different classes of events. The different granularity terms: movement, activity and action, are replaced with the new terms: sub-event, event and super-event. Furthermore, we can indicate temporal composition using the prefix single-thread or multi-thread (i.e. an activity is a complex single thread event, an action is a complex multi-thread event).

Our characterization is in fact similar to Nagel’s hierarchy of occurrences in video sequences. Each level being the building block of the level above it. Our sub-events correlate to Nagel’s “change” and our super-events correlate to “histories”.

Using the terminology defined above we can compare and contrast the various works in the literature. For instance we can see that Gong (Xiang and Gong, 2006) is exploring composite events constructed from pixel-based atomic sub-events. In another example we can characterize (Medioni et al., 2001) as a work in detecting composite single-thread events composed of object-based atomic sub-events which in turn can compose multi-thread super-events.
Chapter 3

Related Work

The main questions in the field of video event understanding are:

- How can the meaningful and discriminating aspects of the video sequence input be extracted?

- How can the events of interest be represented and recognized?

In this chapter we endeavor to organize the methods used in the video event understanding research domain such that their precise role becomes apparent. To achieve this, we have divided the broad research domain of video event understanding into categories. We have grouped together approaches to solving the first question above in a category called abstraction. Approaches to answer the second question aim to find a suitable formalism to both describe interesting events in the input video sequence and allow recognizing these events when they occur. These approaches are grouped together in the category of event modeling.

Both abstraction and event modeling are processes of mapping low-level to high-level information. However, we distinguish abstraction from event modeling in that abstraction molds the data into informative primitive units to be used as input to the event model. The event model may then consider spatial, compositional, temporal, logical and other types of relationships between these primitives in defining the structure of the event. That is, abstraction and event modeling are two parts of the same process.

Abstraction schemes and event models are chosen with respect to the event domain. Approaches to represent and recognize relatively simple events (single actor, known camera angle, pre-segmented event sequences) may identify a discriminating abstraction scheme and utilize a pattern recognition method for event recognition. More involved events (multiple sub-events, numerous actors, complex temporal relationships) may abstract the video sequence as a set of objects and use a semantic event model to represent and recognize the events of interest.
Figure 3.1: Bird’s Eye View of the Video Event Understanding Domain. A video event understanding process takes an image sequence as input and abstracts it into meaningful units. The result of the abstraction is used by the event model to determine if an event of interest has occurred. Output of a video event understanding process may be a decision on whether a particular event has occurred or a summary of events in the input sequence.

There have been several previous efforts to survey this area of research (Aggarwal and Cai, 1999; Buxton, 2002; Turaga et al., 2008; Hu et al., 2004). These papers touch only on a subset of the ideas considered here and often consider video event understanding as a sub-area of a related field.

3.1 Abstraction

Abstraction is the organization of low-level inputs into various constructs (sometimes called “primitives”) representing the abstract properties of the video data.
The motivation for abstraction is to provide an intermediary representation of the video sequence. Although not all papers in the literature focus on abstraction, each work must make decisions on how the low-level input will be presented to the event model (e.g. which features will be used?, will a tracker be applied?). These decisions constitute the abstraction phase (See Figure 3.1) and are an integral part of the event understanding process.

The choice of abstraction is intended to isolate salient properties of the video data especially those that allow useful discrimination between interesting events. Abstraction is thus related to the problem of feature selection. However, feature selection problems usually focus on choosing the most useful of generally simple to extract features (e.g. intensity, edges).

While an abstraction scheme can make use of simple features, many abstraction primitives are more complex aggregations of these simple features (e.g. gradient histograms (Zelnik-Manor and Irani, 2006), motion history images (Bobick and Davis, 2001)) or the output of algorithms that process these features into higher level semantic information (e.g trajectories/bounding boxes).

Abstraction may be a transformation of the low-level input or simply a way of organizing this input. Abstraction approaches may be designed to provide input to a particular event model or to construct informative atomic primitives that can serve as input to a general event model. In this section we will discuss several popular ideas for how to abstract video data.

Along with capturing the important event-discriminating aspects of the video data other main motivations in selecting a particular abstraction scheme are computational feasibility, and ability to complement the chosen event model.

In this section we will discuss three main categories of abstraction approaches: pixel-based, object-based and logic-based abstraction. Each of these approaches is named for the level at which the input is described. Pixel-based abstraction is the category of those abstraction schemes that describe the properties of pixel features in the low-level input. Object-based abstraction approaches describe the low-level input in terms of semantic objects (and their properties). Logic-based abstraction approaches organize the low-level input into statements of semantic knowledge. The following subsections expand on each of these categories with examples from the literature.

### 3.1.1 Pixel-Based Abstraction

In this category we have grouped all those abstraction schemes that rely on pixel or pixel group features such as color, texture and gradient. This scheme is distinguished from other abstraction schemes as it does not attempt to group pixel regions into blobs or objects, but simply computes features based on the salient pixel regions of an input video sequence.
A popular pixel-based abstraction is the organization of low-level input into vectors in an N-dimensional metric space (Bobick and Wilson, 1997; P. Ribeiro, 2005; Zelnik-Manor and Irani, 2006; Shechtman and Irani, 2007; Bobick and Davis, 2001; Zhong, Shi, and Visontai, 2004; Kim, Wong, and Cipolla, 2007). These kind of representations are used to model many different types of data and methods for classification problems on this kind of abstraction are well understood.

The abstract space may be a selection of image features observed directly from the video frames. In this case evaluation may be done to determine which set of the various possible sets of features is the most useful for discriminating between events of interest (using a particular event model) (P. Ribeiro, 2005).

A vector representation may also be the result of the application of some mathematical tool such as dynamic time warping (DTW) (Bobick and Wilson, 1997). A benefit of the vector representation choice of abstraction is that it is fairly general and can be used as input to numerous event models. One drawback of this abstraction approach is that, in some cases, it does not allow for a straightforward semantic interpretation. That is, with the exception of some vector abstractions which allow a meaningful visualization (e.g MHI), the initial abstraction of the video input is meaningless to a human without being processed by an appropriate event model.

The choice of pixel-based abstraction usually follows a researcher’s intuition about what are those pixel feature properties that are important for describing video events. Examples of this abstract video input as intuitively meaningful features such as histograms of spatio-temporal gradients (Zelnik-Manor and Irani, 2006), spatio-temporal patches (Dollár et al., 2005; Laptev and Pérez, 2007; Niebles, Wang, and Fei Fei, 2006), “self-similarity surfaces” (Shechtman and Irani, 2007), and “actionlets” (Wang et al., 2012).

One subclass of these intuitive abstractions that is especially worth noting is that of Motion History Images (MHI). Originally proposed by Bobick and Davis (Bobick and Davis, 2001), the MHI is an intensity image that indicates the spatial location of the most recent pixel motion with higher intensity values. Thus allowing a simple abstraction of a the video input that is also easy to visualize (see Figure 3.2). Other works have expanded on this idea of abstraction (Gong and Ng, 2001; Gong and Xiang, 2003b; Zhong, Shi, and Visontai, 2004).

Pixel-based abstraction approaches are used in a wide variety of event domains including: aerobic exercises (Bobick and Davis, 2001), single person movements (Gong and Ng, 2001; Zelnik-Manor and Irani, 2006; P. Ribeiro, 2005), multi-person activities (Gong and Xiang, 2003b) and traffic monitoring (Zhong, Shi, and Visontai, 2004).
Figure 3.2: A video sequence containing an event (shown as a key frame in the first column) is abstracted as a single Motion History Image in which the pixel intensity at each pixel represents the amount of motion observed at that spatial location during the video sequence. This is an example of Pixel-Based abstraction (inspired by (Bobick and Davis, 2001)).

3.1.2 Object-based Abstraction

Object-based abstraction is an approach based on the intuition that a description of the objects participating in the video sequence is a good intermediate representation for event reasoning. Thus the low-level input is abstracted into a set of objects and their properties. These properties include: speed, position, and trajectory.

Examples of object-based abstractions such as bounding boxes and blobs can be found throughout the literature (Hongeng and Nevatia, 2001; Ghanem et al., 2004a; Tran and Davis, 2008; Borzin, Rivlin, and Rudzsky, 2007; Oliver, Rosario, and Pentland, 2000c). An illustration of this abstraction is shown in figure 3.3.
Figure 3.3: This figure visualizes Object-Based Abstraction. In this type of abstraction scheme objects are located and tracked and the video sequence is abstracted as properties of these objects. In (a) a single person is tracked and a bounding box along with a trajectory from its previous location (visualized by a line) are used to abstract the sequence at each frame. In (b) the same scheme is used for two objects (a person and a cat). (inspired by (Hongeng and Nevatia, 2001))

Object-based abstraction is usually obtained by making use of existing technologies for object detection and visual tracking. These areas are the focus of much attention in the computer vision community and are outside the scope of this work. Interested readers are referred to (Forsyth and Ponce, 2002) for more information.

Silhouettes are a popular object-based abstraction used by many event understanding works(Blank et al., 2005; Schuldt, Laptev, and Caputo, 2004). Many works further manipulate silhouette data using binning (Wang and Suter, 2007),
PCA (Goldenberg et al., 2005), and directionality features (Singh, Basu, and Mandal, 2008).

Trajectories are another very popular object-based abstraction approach prevalent in the literature (Eickhorst, Agouris, and Stefanidis, 2004; Ikizler and Forsyth, 2007; Cuntoor and Chellappa, 2007; Ng and Chua, 2012; Zhou, Wang, and Tang, 2012). Trajectory abstraction are often coupled with pattern recognition methods which allow learning a classifier in an unsupervised fashion from training data (Patino et al., 2008; Piciarelli and Foresti, 2006; Piciarelli, Foresti, and Snidaro, 2005; Piciarelli and Foresti, 2007; Piciarelli, Micheloni, and Foresti, 2008; Gaffney and Smyth, 1999; Porikli, 2004). Several works study the dimensionality reduction of trajectory abstractions using methods such as PCA (Anjum and Cavallaro, 2007), ICA (Antonini and Thiran, 2006) and Fourier transform (Khalid and Naftel, 2005).

Trajectory abstractions are used in event domains including: metro surveillance (Patino et al., 2008), sign language recognition (Bashir, Khokhar, and Schonfeld, 2007), and parking lot surveillance (Piciarelli, Foresti, and Snidaro, 2005).

### 3.1.3 Logic-based Abstraction

A type of abstraction which we have dubbed logic-based abstraction is motivated by the observation that the world is not described by multi-dimensional parameterizations of pixel distributions, or even a set of semantic objects and their properties, but rather by a set of semantic rules and concepts, which act upon units of knowledge. Thus it aims to abstract low-level input into statements of semantic knowledge (i.e. assertions) that can be reasoned on by a rule based event model.

As in other abstraction schemes we have considered the choice of abstraction is motivated by intuition on what are the important properties of the video input that help discriminate between events.

An example of logic-based abstraction can be seen in work by Siskind (Siskind, 2000). Low-level input is abstracted into line segments associated by kinematic stability concepts such as grounding and support. This abstraction is motivated by the biological model.

Another example is provided in work by Cohn et al (Cohn et al., 2003a). The chosen abstraction scheme focuses mainly on the spatial aspects of the event. A set of qualitative spatial relations is applied to the video sequence and relates important image regions to one another.

The advantage of the logical abstraction is that the representation space after the abstraction is much smaller than the original space of all possible inputs. Also the influence of uncertainty errors on quantitative abstraction approaches is

19
reduced substantially.

In this section we have reviewed three categories of abstraction: pixel-based, object-based and logic-based. Abstraction is the critical first part of the event understanding process. It is an essential precursor to the event models discussed later in this chapter.

However, because of the fact that abstraction approaches are closely coupled with event models in the literature not many works provide meaningful evaluation on what is the most powerful abstraction scheme. For example, as discussed above many papers make use of silhouette abstraction coupled with different event models (e.g. Nearest Neighbor, HMM, CRF, etc..) to attempt to classify the same class of events. This analysis, while useful for evaluating the particular event model, tells us nothing as to how influential the choice of the abstraction scheme is. A useful comparative evaluation of different abstraction schemes using the same event model to classify the events within the same domain is difficult to find in the literature.

An exception to this, is found in the sub-set of event understanding often called "action recognition". This event domain generally contains a set of single person events captured from a known camera angle. Recently, the investigation of this domain has gained popularity partly because of the availability of public databases (Blank et al., 2005; Schuldt, Laptev, and Caputo, 2004; Weinland, Özuysal, and Fua, 2010), but largely because of the need to compare and contrast different abstraction schemes and their effect on event recognition performance (Liu, Ali, and Shah, 2008; Natarajan and Nevatia, 2008; Rodriguez, Ahmed, and Shah, 2008; Souvenir and Babbs, 2008; Thurau and Hlavac, 2008; Banerjee and Nevatia, 2011; Song, Morency, and Davis, 2012).

It is intuitive that the more complex the choice of abstraction, the more useful it will be when it comes time to be reasoned on by an event model. However, the computation of some very complex abstractions is prohibitive. These include heavy tracking approaches, multi-dimensional histograms, and others. Thus, for papers that propose full systems of event understanding we often see that more light-weight tracking approaches such as background subtraction are used in contrast to particle filter or other more computation intensive trackers.

### 3.2 Event Modeling

Event modeling is the complementary problem to abstraction (discussed in the previous section). This discipline seeks formal ways to describe and recognize events in a particular domain given the choice of an abstraction scheme. A particular event model is chosen based on both its capacity for representation of salient events in a particular domain and its capacity for recognition of those
events as they occur in the video sequence input.

Many works in event understanding literature focus on this problem of finding a suitable modeling formalism, an event model, to describe the occurrences of interest in a particular domain.

Event models have many aspects and, hence, they may be categorized in several different ways. One such categorization, is the distinction between deterministic and probabilistic event models. Examples of event models in the deterministic category include Finite State Machines (FSM), Grammars and Petri-Nets, while Bayesian Networks, Hidden Markov Models (HMMs), Conditional Random fields (CRF) and Stochastic grammars all associate a probability score with an event occurrence.

Some event modeling formalisms can be expressed graphically. That is, they can be defined using the mathematical notion of a graph, which utilizes such concepts as "nodes" and "arcs". Yet others, known as "probabilistic graphical models" (or sometimes simply "graphical models"), comply with a stricter definition that requires a graph in which nodes represent stochastic variables and arcs represent dependencies. Petri-Nets and FSM event models lie in the former category while Bayesian nets, HMMs and CRFs are in the latter. There are also models which cannot be fully represented graphically, including multi-variable Support Vector Machines (SVM) and Grammar models.

Another distinction that can be drawn among event models is between generative and discriminative models. In a statistical (probabilistic) framework, discriminative models are defined as those models that directly model the posterior probability for use in classification tasks. This is in contrast to so-called generative models which model and learn the joint distribution over all possible observations and labels and then use this information to extract the posterior probability (i.e. the distribution over class labels given the observations). Intuitively, a discriminative model is only concerned with the decision problem, while a generative model attempts to capture the phenomenon that "generates" both events and observations. Outside of a statistical framework, the distinction between generative and discriminative models is less well-defined.

Works contrasting the application of discriminative and generative of models (Ng and Jordan, 2001) have shown that discriminative models have lower error rates as the number of training examples becomes very large. However, they also found that generative models tend to converge to their optimal performance much quicker (with less training examples) than their discriminative counterparts. That is, if a smaller body of training data is available the generative approach might be favorable. Additionally, generative models have been found to be more flexible to incomplete training data and new classes and may be better suited to learning complex patterns (Ulusoy and Bishop, 2005).

The above categorizations are useful in many ways, but they do not fully
capture the diversity of event modeling approaches in the event understanding literature. For this reason, this chapter proposes a categorization that the authors feel best allows us to describe the domain of event modeling.

We have chosen to organize event models into three different categories: "Pattern Recognition Methods", "State Models", and "Semantic Models". These are related in some ways to the model categories discussed above, but there is not a direct one to one relationship. Not every event modeling approach in the literature necessarily falls exclusively into one of these categories, but we believe they represent the general spirit of approaches within recent event modeling research. This categorization is closely related to the categorization of event models by temporal relationship complexity proposed in (Bremond, 2007).

"Pattern Recognition Methods" do not generally address the event representation aspect of event modeling and approach the event recognition problem as a traditional pattern recognition/classification problem. Accordingly, traditional approaches to these problems such as Support Vector Machines, Neural Networks, Nearest Neighbor Classifiers, etc. are applied to the abstraction scheme. Minimal semantic knowledge is needed in building the event classifiers in this category. Often they may be fully specified from training data.

More recent approaches attempt to model video events using semantic information. The first class of these models, we have named "State Models" for the reason that they concentrate on specifying the state space of the model. Often this state space is reduced or factorized using semantic knowledge. This class of approaches includes FSMs which conceive video events as fully observable processes of a set of known states. These states may have semantic meaning taken from domain knowledge. The transitions between these states are also specified using knowledge of the domain. This class also includes the set of probabilistic graphical model approaches. These approaches factorize the state into variables (the structure of the graph) according to some semantic knowledge of the domain. The existence (under some structural assumptions) of efficient algorithms for the learning of parameters from training and the inference of hidden node values also motivate the choice of probabilistic graphical models to model video events. This enables some degree of automated learning of abstract parameters. However, as previously stated, the structure of this type of model is usually specified with some knowledge of the event domain. This knowledge is usually implicit and imposed on the model indirectly (e.g. assigning a semantic label to each state). This class of models largely consists of approaches fully specified using a combination of knowledge based structure specification and automatic parameter learning from training data. In addition to their flexibility, generative models also exhibit adaptive properties. This is in contrast to discriminative models which must be completely relearned for each new decision class added. These properties are probable reasons why the work in this category of event modeling largely favors
generative models, although newer work proposes discriminative approaches (e.g. CRF).

Higher-level semantics include sub-event ordering information (including partial ordering), complex temporal, spatial and logical relations among sub-events. Also important is the ability to express and recognize partial events. These properties become important when the event domain includes high-level events which are best expressed in qualitative terms and natural language. To this end a group of modeling formalisms we have named "Semantic Models" have been proposed which enable explicit specification of these complex semantic properties. Among these are Petri-Nets and Grammar models as well as constraint satisfaction and logic based approaches. These models are usually fully specified using domain knowledge and are not usually learned from training data.

In the following sections we will take a more in depth look at the three categories of Event Modeling and explore the various formalisms contained within each category with examples from the literature. Particularly we will discuss the representational strengths and recognition efficiency of each category. We will also provide discussion on the event types and domains typically modelled by the approaches in each of our categories.

3.3 Pattern Recognition Methods for Event Recognition

The class of techniques in this section are not quite event models, in the sense that they do not consider the problem of event representation. Instead they focus on the event recognition problem, formulated as a traditional pattern recognition problem. This class includes well studied approaches including Nearest Neighbor, Support Vector Machines, Neural Networks.

The main advantage of the classifiers in this category is that they may be fully specified from a set of training data. These approaches are usually simple and straightforward to implement. This simplicity is afforded by excluding semantics (i.e. high-level knowledge about the event domain) entirely from the specification of the classifier. The representational considerations are usually left to the abstraction scheme associated with the event recognition method in the particular event understanding application.

Nearest neighbor (NN) is a well-studied pattern matching technique for classification. An unlabeled example is labeled using its "nearest" labeled neighbor in the database. $K$-NN is a variation of nearest neighbor where the $K$ nearest neighbors vote on the label of the test example. The notion of closeness is defined by a distance measure decided upon during the model specification. (Bishop, 2006)

Nearest Neighbor models can naturally be coupled with a vector abstraction
of the data as well as other abstractions assuming an appropriate distance metric between abstraction has been defined.

The choice of the distance measure in the various works utilizing a NN event model for event understanding is usually chosen with respect to the abstraction of the video input. Clever choices of distance measures allow for better event discrimination. Choices range from simple distance measure such as Euclidean (Blank et al., 2005; Masoud and Papanikolopoulos, 2003) and Chi-Squared (Zelnik-Manor and Irani, 2006) to more complex choices such as Linear programming based distance (Jiang, Drew, and Li, 2006). Some choices for distance measures are event-domain dependent such as spatio-temporal region intersection (Ke, Sukthankar, and Hebert, 2007) and gradient matrix of motion field (Shechtman and Irani, 2005).

So called template matching techniques (Bobick and Davis, 2001; Gong and Ng, 2001) also utilize a NN event model.

The NN model does not inherently allow representation of important properties of video events. However, the more dense the coverage over the space of possible abstraction the better the model performs in classification tasks. Unfortunately, as the database of examples grows so does the recognition time of the algorithm. Because the abstraction of video events is often high-dimensional, a sufficiently dense NN event model is still intractable for recognition. This is especially true when a complicated distance measure is utilized. Furthermore, in many domains it is not possible to collect or store such a database of examples and many approaches utilizing NN use a sparse database of examples to increase the efficiency of both storage and recognition.

Nearest Neighbor event models have been used in event domains including: aerobic exercises (Bobick and Davis, 2001), single actor actions (Zelnik-Manor and Irani, 2006; Blank et al., 2005), and Ice skating maneuvers (Shechtman and Irani, 2005).

Support vector machines (SVM) (Cristianini and Shawe-Taylor, 2000; Burges, 1998) are a group of models designed to find the optimal hyperplane separating two classes in a multi-dimensional space.

Each SVM may be associated with a kernel function to transform the multi-dimensional space (which is possibly non-separable) into a higher dimensional space which is more likely to be separable.

The kernel function is usually chosen using knowledge of the domain. Additionally, abstraction schemes coupled with SVM event models are often chosen to be quite complex. In addition to a classification from a SVM model we may also get a confidence score. This score is equivalent to the distance in the decision space between a particular example and the decision boundary.

The separating hyperplane is determined in a training phase using labelled
training data by solving an optimization problem maximizing the distance between examples with different labels in this higher dimensional space.

The two class SVM can be generalized to a multi-class decision problem. Pittore et al. (Pittore, Basso, and Verri, 1999) uses such a scheme for the classification of object-based atomic events in the event domain of office surveillance.

In the event understanding literature SVMs utilize a wide variety of kernel functions including: linear (Pittore, Basso, and Verri, 1999), tree-kernel (Fleischman, Decamp, and Roy, 2006), polynomial (Cao et al., 2004), Radial basis function (RBF) (Xu and Chang, 2007).

Like the nearest neighbor model the SVM is reliant on the abstraction scheme associated with it to represent the salient properties of video events. In and of itself it does not have the capacity to capture these properties. Furthermore, the high-dimensional "decision space" representation created using the kernel function, while allowing better separation of the examples, is abstract and not meaningful outside the context of the SVM decision.

SVMs have been shown to be very powerful for many classification tasks. However, the training phase of SVMs (i.e. calculating the separating hyperplane) can be lengthy in high-dimensional problems.

Additionally, because of its inability to represent event properties such as temporal evolution, an SVM event model is often coupled with complex abstraction schemes which may themselves be difficult to compute efficiently.

SVM classifiers are coupled with various abstraction schemes including both pixel-based (Pittore, Basso, and Verri, 1999) and object-based (Piciarelli, Foresti, and Snidaro, 2005) abstractions. SVM classifiers are used in such event domains as parking lot surveillance (Piciarelli, Foresti, and Snidaro, 2005), single person actions (Cao et al., 2004), kitchen activities (Fleischman, Decamp, and Roy, 2006), facial expressions (Simon et al., 2010), and violence detection in sports video (Nievas et al., 2011).

The general approach to combining classifiers to enhance the accuracy of the final classification is called boosting (Freund and Schapire, 1999). Boosting is not in itself an event recognition technique but rather is often applied to pattern recognition classifiers to yield an improved event classifier. We discuss this topic here because boosting approaches are often applied to yield event classifiers for video event understanding.

Most of the popular boosting methodologies are used to construct event models in the literature of video event understanding including: AdaBoost (Truyen et al., 2006; Laptev and Pérez, 2007; Minnen, Westeyn, and Starner, 2007) and LPBoost (Nowozin, Bakir, and Tsuda, 2007). Other, less well-known, boosting methodologies such as "Knapsack Boost" (Lv et al., 2004) are also in use.
In some cases the properties of video events have guided the choice of boosting scheme (i.e. how to combine classifiers). TemporalBoost (Smith, da Vitoria Lobo, and Shah, 2005; Ribeiro, Moreno, and Santos Victor, 2007; Canotilho and Moreno, 2007) uses the idea of dependency between frames in a video sequence.

Boosting based classifiers are efficient for recognition, but require a lengthy training phase. Additionally, these type of event models do not capture intrinsic properties of video events and thus depend on their associated abstraction scheme to capture these.

Boosted event classifiers have been used in event domains such as single person actions (Smith, da Vitoria Lobo, and Shah, 2005; Ribeiro, Moreno, and Santos Victor, 2007), metro surveillance (Canotilho and Moreno, 2007), and office surveillance (Lv et al., 2004).

Another well known pattern recognition technique is the Neural Network. This type of classifier simulates the biological system by linking several decision nodes in layers. This approach has been used to solve complex decision problems and recognize patterns in many domains. Early work in video event understanding explored the use of Neural Networks as an event classifier (Vassilakis, Howell, and Buxton, 2002).

The strength of the pattern recognition approaches to event recognition is that they are well understood and are straightforward to both formalize mathematically and implement practically. They also allow learning of the model from the data. In cases where a sufficient amount of training data is not available such as recognition of rare events, these methods are naturally less effective. Furthermore, when events become more complex and are defined by their semantic content, that high-level knowledge that humans use to define video events, pattern recognition approaches also fall short due to their limited ability to capture aspects of events such as: meaningful state space factorization, temporal extent, and composition in both space and time. For these, these models are most frequently applied to the recognition of atomic events. The limitations of pattern recognition event methods are addressed by other classes of event models we will explore in subsequent sections.

3.4 State Event Models

“State” event models are a class of formalisms which are chosen using semantic knowledge of the state of the video event in space and time. Reasonable assumptions about the nature of video events have lead to the development of each of these formalisms. Each of these capture an important aspect or property of video events through their use.
State event models improve on pattern recognition methods in that they intrinsically model the structure of the state space of the event domain. This is used for capturing both the hierarchical nature and the temporal evolution of state, that are inherent to video events. This modeling capacity generally increases the ability of these event models to represent different types of events, even when coupled with a simple abstraction scheme.

Modeling formalisms in this category, not only capture some inherent properties of video events, but are also well studied and mathematically well formulated to allow for efficient algorithms and sound formulations of problems such as parameter learning and event recognition.

In most, but not all, cases the semantic information associated with the model structure makes this structure difficult to learn from training data. However, once the model structure is specified model parameters can often be learned from the training data. This aspect of state models contributes to their popularity allowing them to combine human intuition about the event structure (semantics) and machine learning techniques.

A category closely associated with State Models is that of generative models from statistics and machine learning. This is the class of models that attempts to model the phenomenon that "generates" the observation. However, as we shall see not all State event modeling formalisms are generative.

State modeling formalisms include: Finite State Machines (FSMs), Bayesian Networks (BN), Hidden Markov Models (HMM), Dynamic Bayesian Networks (DBN), and Conditional Random Fields.

### 3.4.1 Finite State Machines

Finite State Machines (FSM) (Gill, 1962), also known as Finite State Automata, are a formalism useful for modeling the temporal (especially sequential) aspects of video events. This formalism extends a state transition diagram with start and accept states to allow recognition of processes. FSMs are traditionally deterministic models and provide computationally efficient solution to reasoning about event occurrences.

The strengths of the FSM model are in its ability to model sequential aspects of video events, its model simplicity and its ability to be learned from training data. FSMs are also a well studied formalism which allows for straightforward analysis of running time complexity.

FSMs appearing in the literature naturally model single-thread events formed by a sequence of states. Event domains for which FSM event models are utilized include hand gestures (Jo, Kuno, and Shirai, 1998), single actor behavior (Lv and Nevatia, 2007), multiple person interaction (Hongeng and Nevatia, 2001), and aerial surveillance (Medioni et al., 2001).
Figure 3.4: FSM event model used on the "car avoids checkpoint" event a la
(Medioni et al., 2001)

The inherent ability of the FSM formalism to capture sequence allows it to be
associated with different abstraction types including pixel-based (Jo, Kuno, and
Shirai, 1998) and object-based abstraction (Hongeng and Nevatia, 2001; Medioni
et al., 2001; Bremond and Thonnat, 1997; Bremond, Thonnat, and Zuniga, 2006).

The FSM assumption of a fully observable state is not present in other State
event modeling formalisms. It allows the event recognition problem to be reduced
to accepting/rejecting the process representing the event. Additionally, because
all states, input symbols and state transitions are fully observable, an FSM model
may be learned from training data (Hong, Huang, and Turk, 2000). Some work
has also been done on inferring an FSM event model from user specification
(Bremond and Thonnat, 1997; Bremond, Thonnat, and Zuniga, 2006).

FSMs are a useful tool in video event understanding because of their simplicity,
ability to model temporal sequence and their learnability.

Extensions to the FSM have been proposed to capture the hierarchal prop-
erty of video events (Park, Park, and Aggarwal, 2003; Mahajan et al., 2004; Lv
and Nevatia, 2007; Yuan and Xu, 2011). Uncertainty in video events has also
been addressed through the introduction of probabilities into the FSM framework
(Mahajan et al., 2004). It should be noted that in some areas of the event under-
standing literature the terms of "HMMs" (see section 3.4.3) and "probabilistic
FSMs" are used interchangeably. The main distinction is that FSMs assume a
fully observable state while HMMs assume a hidden state variable.

These extensions to the FSM formalism have attempted to introduce aspects
such as hierarchy and uncertainty. These methods have largely been applied to
specific event domains and have not been embraced as general solutions. This is largely because of the availability of other formalisms that are well adapted to such aspects (e.g. the HMM for uncertainty).

3.4.2 Bayesian Networks

In order to deal with the inherent uncertainty of observations and interpretation which exists in video events, Bayesian Network event models utilizing probability as a mechanism for dealing with uncertainty, have been proposed.

Bayesian Networks (BN) (also known as probabilistic networks, Bayesian Belief networks, or independence diagrams) are a class of directed acyclic graphical models. Nodes in the BN represent random variables which may be discrete or continuous. Conditional independence between these variables are represented by the structure of the graph.

The structure of the BN allows specification of the joint probability over all variables in a succinct form with few parameters, using the notion of conditional independence. For further details readers are referred to (Jensen, 2001; Pearl, 1988).

Having such an expression of the joint probability allows us to reason about any node in the network using known values. Often BN event models will model the event as an unknown or "hidden" binary variable (event has/hasn’t occurred) and the observations (abstraction primitives) as known variables. The BN structure (nodes and arcs) and parameters (conditional and prior probabilities) can be used to estimate the distribution of unknown variables given the value of known variables.

While the general inference (i.e. the estimation of hidden variables given observed variables) problem in BNs is NP-hard, efficient algorithms for inference exist under certain BN structure assumptions (Pearl, 1988). BN model parameters may also be learned from training data. Additionally, network structure learning has also been explored in the literature. (Jordan, 1998)

As an example of the standard approach to event modelling using BN we consider the "pedestrian crossing street" event. We will choose this event to be atomic, that is having no sub-event composition. Therefore we will have the event decision based on abstraction primitives. We will construct a simple BN as pictured in Figure 3.5 in which the top node will represent the occurrence of the event (i.e. discrete binary variable). The bottom nodes in the figure correspond to abstraction primitives. The first node corresponds to location of the person and the second to direction of the person. For simplicity we will consider these nodes to be discrete. The first node can take on the values “inside intersection” or “outside intersection”. The second node can take on the values “not moving” “moving towards sidewalk” or “moving not towards sidewalk”. As we observe the
abstraction of video input, the value of these nodes will be available at each frame. We still must determine the parameters for the BN. These parameters can be set using semantic domain knowledge (i.e., we know that during "pedestrian crossing street" the person is "inside crosswalk" with a high probability). The parameters can also be learned from training data. In the training data we will have knowledge of the event variable's value which will allow us to set the parameters according to observed frequency. In the case of a more complex BN with multiple hidden variables the expectation maximization (EM) algorithm is used for parameter estimation. Once the parameters are estimated we can observe test data and determine whether our event is occurring or not by calculating the joint probability and then marginalizing out other variables to obtain the marginal probability over our event variable.

This method achieves a probability score indicating how likely the event is to have occurred given the input.

BN models do not have an inherent capacity for modelling temporal composition which is an important aspect of video events. Solutions to this problem include single-frame classification (Buxton and Gong, 1995) and choosing abstraction schemes which encapsulate temporal properties of the low-level input (Intille and Bobick, 1999; Lv et al., 2006). Naive Bayesian networks, such as the
one pictured in Figure 3.5, appear often throughout the event understanding literature. This model is sometimes called an "agent" architecture because several Bayesian "agents" are applied to objects of interest within the video sequence input. This structure is also well adapted to the hierarchical composition inherent to many video events. This is because the probability output of the top node in a sub-event network can be easily integrated as an "observation" node in a higher-level event model.

Agent architectures have been used in event domains such as aerial surveillance (Buxton and Gong, 1995; Higgins, 2005), and indoor surveillance of people (Lv et al., 2006). More complex BNs have been used in event domains such as parking lot surveillance (Remagnino, Tan, and Baker, 1998) and recognizing American football plays (Intille and Bobick, 1999). Although these networks are large they retain a structure that allows for efficient inference.

Inference algorithms, such as belief propagation, for calculating the marginal distribution (i.e. belief) of a particular node run in time polynomial in the number of states per node in the BN under these structural assumptions.

The BN agent approaches in the literature are more commonly associated with object-based abstractions (Buxton and Gong, 1995; Higgins, 2005; Lv et al., 2006; Intille and Bobick, 1999; Remagnino, Tan, and Baker, 1998).

Modeling the hierarchy of video events is straightforward within the BN framework. Hongeng and Nevatia (Hongeng, Nevatia, and Bremond, 2004) model the semantic hierarchy using BN layers. Each layer corresponds to a higher-level semantic units.

A more recent group (Niebles, Wang, and Fei Fei, 2006; Wong, Kim, and Cipolla, 2007; Wang, Ma, and Grimson, 2007) of works make use of Bayesian Network models adapted from the text and image mining communities. These approaches are also known as Probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 1999) and consider variables representing documents, words and topics which, in the event understanding domain, correspond to video sequences, abstraction primitives, and events, respectively. These types of approaches are most commonly associated with pixel-based abstractions (e.g. "cuboids" (Dollár et al., 2005; Niebles, Wang, and Fei Fei, 2006))

Bayesian networks are powerful tool in factorizing the state space into variables using semantic knowledge of the domain and specifying a joint distribution over all possible values of these variables succinctly. This formalism naturally models the hierarchical and semantic state of video events. The probabilistic output of BNs is useful for addressing uncertainty. This formalism also allows computationally tractable solutions for inference. The main shortcoming of the BN model is in modelling the temporal aspects of video events.
3.4.3 Hidden Markov Models

The benefits of a temporal evolution model (like FSM) and a probabilistic model (like BN) are combined within the framework of the Hidden Markov Model event model.

Hidden Markov Models (HMM) are a class of directed graphical models extended to model the temporal evolution of the state. The HMM has a specific graph structure associated with it. One variable, representing the hidden state, and one variable, representing the observation, comprise a single "time slice". The "time slices" represent the evolution of the process (event) described by the model over time. Intra-slice arcs indicate the dependence of the observation variable on the state variable. Inter-slice arcs connect the state variable in the previous slice to the state variable in the current slice. This structure describes a model where the observations are dependent only on the current state. The state is only dependent upon the state at the previous "time slice" (the Markov assumption). This structure (see Figure 3.6a) is imposed to allow efficient inference algorithms.

Since the HMM structure is fixed and repetitive we can define the likelihood of long sequence of states (and corresponding observations) by specifying the following parameters: the initial (distribution over initial state values), the transition (distribution over the state given the previous state), and the emission (distribution over observations given the state) probabilities. The number of parameters required to specify these probabilities depends on the number of states and observation symbols, which are usually determined empirically.

There exist well-studied polynomial (in the number of hidden states) time algorithms for evaluation, inference and learning in HMMs. For further details regarding HMMs the reader is referred to (Rabiner, 1990; Ghahramani and Jordan, 1995).

A common use for HMMs in modeling video events is as follows. An HMM event model is defined by observation symbols related to the chosen abstraction scheme. The states of the HMM are usually not semantically meaningful and their number is chosen empirically. The parameters of the HMM model may be learned from training data or specified manually using knowledge of the event domain. To discriminate between events, such an HMM event model is trained for each event under consideration. Test examples are then evaluated to determine how likely they are to have been generated by each of the HMM models. The event model that yields the highest likelihood score is used to label the test example.(Gong and Buxton, 1992)

A number of early works in the literature employ this approach in the event domains of tennis stroke recognition (Yamato, Ohya, and Ishii, 1992), Sign Language and gesture recognition (Starner and Pentland, 1995; Schlenzig, Hunter,
and Jain, 1994), single-person actions (e.g. "walking", "kneeling") (Ogale et al., 2004). The events recognized in these works are mostly a few seconds in length. Furthermore, these methods are generally dependent on adequate segmentation of the video sequence into event clips. That is, before we can classify the event in a given video sequence we must be given a clip known to contain an event (and only one event).

In more recent work the HMM model has been extended in several ways to adapt to the challenges of modelling video events. One such challenge is the representation of the state and observation spaces within one variable, respectively. As the number of states and observations grow this representation requires a great deal of parameters to be estimated and therefore a large set of training data (often larger than what is available). To deal with this challenge, solutions factorize the observation space into multiple variables or alter the network topology (Figure 3.6).

Multi-Observation Hidden Markov Models (MOHMM) (Xiang and Gong, 2006) use multiple variables to represent the observation. The variables are casually dependent on the state variable, meaning they are conditionally independent of one another given the state. This model reduces the number of parameters to be learned and thus makes parameter estimation from a finite set of training data more likely to produce good results. Parameter estimation of MOHMMs is similar to that of HMMs except that additional emission probabilities for each additional observation variable must be defined.

Another approach to reducing the parameters to be learned is altering the network topology (specifically which states are reachable from which other states) (Brand and Kettnaker, 2000). For certain events, those composed of an ordered sequence of states, a fully connected transition model has unnecessary parameters. An HMM topology which only allows transitions from one state to the next state in the sequence (without skipping states) would greatly reduce the number of parameters (all parameters not fitting these constraints would be set to zero). This kind of topology is called a casual or left-right HMM (with no-skip constraint).

Often the event would be more naturally (from a semantic perspective) modelled with two or more state variables, forming state chains over time. Factorizing the state space into these multiple state chains is another way to simplify the event model. These multiple chains could correspond to simultaneous sub-events in a composite event or multiple objects interacting within an atomic object-based event. Of course, some way of merging the output likelihoods of these chains while taking into account the dependencies between them is needed.

Several event models with variations on this approach exist. In Parallel Hidden Markov Models (PaHMM) (Vogler and Metaxas, 2001) the multiple chains of the state are modelled as separate HMMs each with its own observation sequence.
Coupled Hidden Markov Models (CHMM) (Brand, Oliver, and Pentland, 1997; Brand, 1997; Oliver, Rosario, and Pentland, 2000b) where the hidden process chains are coupled in such a way that the current state in a particular chain depends on the previous state of all chains. Dynamic Multi-Level Hidden Markov Models (DML-HMM) (Gong and Xiang, 2003a) extend the coupling concept by attempting to learn the dependencies between the hidden state chains. That is, the state space is reduced by both separating the state into multiple variables and simplifying the topology.

As expected, these extensions are used in event domains where there are several elements participating in the events of interest including sign language (Vogler and Metaxas, 2001), Tai-Chi gestures (Brand, Oliver, and Pentland, 1997), multiple person interactions (Oliver, Rosario, and Pentland, 2000b), and airport tarmac surveillance (Gong and Xiang, 2003a).

The multiple chains also allow a relaxation of the linear temporal order of the states. That is, more complex temporal relationships between the state sequences can be modeled in this way.

However, as the topology becomes more complex the efficient exact algorithms associated with the "pure" HMM structure are no longer applicable and must be replaced by approximation algorithms. An experimental comparison of MOHMMs, PaHMMs, CHMMs and DML-HMMs can be found in (Xiang and Gong, 2006).

Another extension to the basic HMM structure is motivated by the long-term temporal dependence of state variables within a video event. That is, the Markov assumption, that the current state depends only on the state at a previous time, is not necessarily valid. The reason for this may be inherent long term dependencies or occlusions and other phenomena that cause errors in state estimation. Figure 3.7 visualizes some of these approaches.

N-order hidden Markov models deal with this problem by amending the Markov assumption to consider the N previous states. Variable Length Markov Models (VLMM) (Galata, Johnson, and Hogg, 2001; Galata et al., 2002) calculates the optimal level of temporal dependence using a divergence criterion. Hidden Semi-Markov Models (HSMM) (sometimes called Semi-HMMs) (Hong-geng and Nevatia, 2003a) allow each state to emit multiple observations. That is along with the state variable at each time there will also be a duration variable (observation length).

Parameterized HMMs (Wilson and Bobick, 1998) introduce extra parameters to model events that may have variance that does not affect the event classification. These parameters may have a semantic meaning and may or may not be measurable. These parameterizations prevent classification errors due to variance in this parameter. Secondly, in estimating this parameter extra information about the event is obtained.
Several HMM extensions have been proposed to incorporate the inherent hierarchical composition of video events into the event models. In Hierarchical Markov Models (HHMM) (Fine, Singer, and Tishby, 1998; Nguyen et al., 2005), each possible state is represented by a lower-level HMM. In a similar approach Oliver et al. (Oliver, Horvitz, and Garg, 2002) uses a Layered HMM (LHMM) event model in the event domain of office surveillance.

Several efforts have been made to integrate the various classes of extensions to the HMM event model into a single formalism. That is, an event model
Figure 3.8: "Hybrid" HMM models capturing the intrinsic properties of video events.

that models long-term temporal dependence, hierarchical composition and factorization of the state space into multiple variables. These include the Switching Hidden Semi-Markov Model (S-HSMM) (Duong et al., 2005), Hierarchical Semi-Parallel Hidden Markov Models (HSPaMM) (Natarajan and Nevatia, 2007b) and the Coupled Hidden semi-Markov models (CHSMMs) (Natarajan and Nevatia, 2007a). Figure 3.8 illustrates some of these "hybrid" HMMs.

Hidden Markov Models are among the most popular formalisms for modelling video events. As the event being modeled becomes more complex and has interesting properties such as long term dependence and hierarchical composition, the basic HMM has evolved complicated variations. These extensions attempt to introduce more and more semantics into the formalisms. Unfortunately these semantically enhanced models often come at the cost of tractability. The structural constraints that afford tractability are the original motivation for adopting the HMM and must be adhered to in order to have a practical event model. This
means finding a balance between a model that captures the properties of video events well and a model that is realistic for application.

### 3.4.4 Dynamic Bayesian Networks

As we have seen in the previous section, event models in some cases benefit from a meaningful factorization of the state and observation space. An event modeling formalism which allows such general factorization while still capturing the temporal evolution of state is the Dynamic Bayesian network.

Dynamic Bayesian networks (DBN) generalize Bayesian Networks (BN) with a temporal extent. They can be described formally by intra-temporal dependencies and inter-temporal dependencies. The former is described as a "static" BN and the latter as a special two-slice BN. In this specification the Markov assumption is preserved. HMMs are a special case of DBNs in which the structure is restricted to provide efficient algorithms for learning and inference. All HMM variants previously discussed are also special cases of the DBN. The strength of the general DBN in comparison to HMM is its ability to factorize the state-space of the model in semantically meaningful or classification performance enhancing ways. This, however, often comes at the cost of computational tractability. Approximation techniques are usually used to perform learning and inference on general DBNs.

Because exact solutions are not available, general DBNs are not often used for modelling video events. Instead specific DBNs with structural assumptions that yield computationally tractable algorithms (such as the HMM and its variants) are often used.

Because of the rich information about the structure of the event contained in the event model, a relatively simple pixel-based abstraction scheme is coupled with many DBN event models (Oliver and Horvitz, 2005; Shi et al., 2004; Shi and Bobick, 2003; Muncaster and Ma, 2007).

DBN approaches have been applied in event domains such as the office environment (Oliver and Horvitz, 2005), assisted living (Shi et al., 2004; Shi and Bobick, 2003; Laxton, Lim, and Kriegman, 2007), and surveillance of people (Moenne-Loccoz, Bremond, and Thonnat, 2003; Muncaster and Ma, 2007).

To overcome the computationally hard inference in DBNs many of the works in the literature make simplifying assumptions such as restricting the temporal links in the graph (Moenne-Loccoz, Bremond, and Thonnat, 2003), restricting state transition topology (Laxton, Lim, and Kriegman, 2007; Shi et al., 2004; Shi and Bobick, 2003).

Apart from learning the DBN parameters from the data many recent work attempt to learn aspects of the model such as model structure (Oliver and Horvitz, 2005), abstraction scheme (Shi, Bobick, and Essa, 2006), and the number of states each variable takes on (Muncaster and Ma, 2007).
Figure 3.9: A CRF is a discriminative undirected graphical model with structure inspired by the HMM

Dynamic Bayesian Networks in their general form appear less often as event modeling formalism in the literature. Special constrained cases of the DBN (most notably the HMM), however, are quite popular as event models throughout the event understanding community.

3.4.5 Conditional Random Fields

One drawback of generative models in general and HMMs in particular, is their dependence on the availability of a prior on the observations (abstraction primitives). This prior is not always known and frequently estimated using assumptions that will yield efficient computation, such as independence between observations given the state (a la HMM). In the domain of video events this is often an invalid assumption. In a discriminative statistical framework only the conditional distribution is sought (modeled) and as such there is no need for such restrictive assumptions. The adoption of conditional random fields as event models is based on this idea.

Conditional Random Fields (CRF), recently introduced in (Lafferty, McCallum, and Pereira, 2001), are undirected graphical models that generalize the Hidden Markov Model by putting feature functions conditioned on the global observation in the place of the transition probabilities. The number of these functions may be arbitrarily set. Existing known algorithms for HMM problems of inference and evaluation can be extended to CRFs. Learning of CRF parameters can be achieved using convex optimization methods such as conjugate gradient descent (Sutton and McCallum, 2006).

In event modelling, CRFs have consistently been shown to outperform HMMs for similar event recognition tasks (Sminchisescu et al., 2005; Wang et al., 2006). This is attributed to the ability to choose an arbitrarily dependent abstraction
scheme. Furthermore, in a CRF, unlike in the HMM, abstraction feature selection does not have to be limited to the current observation but can also consider any combination of past and future observations. A major disadvantage of CRF models in comparison to HMMs is their parameter learning time. The optimization procedures like conjugate gradient descent take a significantly longer time than the training of HMMs.

Several more recent works have attempted to introduce additional structure into the CRF formalism using knowledge of the event domain (Wang and Suter, 2007; Morency, Quattoni, and Darrell, 2007; Ning et al., 2008). These extensions to the original CRF structure to better capture some inherent properties of the event domain are similar to those extensions for HMMs discussed in Section 3.4.3.

CRFs are a recently popular event modeling formalism that is straightforward to apply in cases where HMMs have been applied before and achieve better event recognition result. The tradeoff incurred by this is a significantly longer training time.

3.5 Semantic Event Models

While many events can be described as a sequence of a number of states, an interesting subset of events are those defined by the semantic relationships between their composing sub-events. For instance, a “Bank Attack” event may be partially defined as ("robber enters zone" during "cashier at position") before ("cashier at safe" during "robber at safe") (Vu, Brémont, and Thonnat, 2003). This is an example of an event definition which makes use of temporal and spatial semantic relationships. To allow these kind of semantic relationships to be represented and recognized several event modeling formalisms have been proposed. We have grouped these in the category of "Semantic Event Models".

The class of Semantic Event Models contains event modelling approaches that do not aim to define the entire state space of the event domain as in "State Model" approaches. Semantic knowledge is still used to construct the event model. However, the event model is defined in terms of semantic rules, constraints and relations. That is, there is a large degree of overlap between how humans describe what constitutes an event and how it is defined within these modelling formalisms. Recognizing an event as it occurs becomes a problem of “explaining” the observation using the available semantic knowledge.

This type of approach allows the event model to capture high-level semantics such as long-term temporal dependence, hierarchy, partial ordering, concurrency and complex relations among sub-events and abstraction primitives. Additionally, "incomplete" events, those observations that do not constitute a recognized event, can contribute meaningful information. For instance, answering the question of
“how far?” is the completion of an event of interest.

Because of the high-level nature of this class of models they often must be manually specified by a domain expert. That is, learning model structure and/or parameters is generally infeasible/ill defined. Furthermore, the formalisms in this category of event models are largely deterministic and the convenient mechanism of probabilistic reasoning to handle uncertainty (both in observation and interpretation) is generally unavailable.

The semantic event models are usually applied in event domains where the events of interest are relatively complex and a particular event has large variance in its appearance (Vu, Brémond, and Thommat, 2003; Tran and Davis, 2008; Borzin, Rivlin, and Rudzsky, 2007; Bobick and Davis, 2001).

In the following sections we will explore such semantic event modelling formalisms including: Grammars, Petri-Nets, Logics and Constraint Satisfaction approaches.

The commonality of all these approaches is that the event model is fully specified in terms of high-level semantic concepts.

### 3.5.1 Grammars

Language is a basic mechanism used by humans to define and describe video events. It is therefore intuitive that formal notions of language, as defined by grammar models would be natural to model the inherently semantic properties of video events.

Grammar models (Aho and Ullman, 1972) are well studied and have been used in several domains including Speech Recognition (Jelinek, 1998) and Computer Vision (Chanda and Dellaert, 2004).

Grammar models specify the structure of video events as sentences composed of words corresponding to abstraction primitives. The grammar formalism allows for mid-level semantic concepts (parts of speech in language processing). In the event model context, these mid-level concepts are used to model composing sub-events. This formalism naturally captures sequence and hierarchical composition as well as long-term temporal dependencies.

When we discuss semantic grammar models in this section we are referring to those approaches that infuse semantics into the grammar rule description rather than those grammars that are simply an equivalent representation of a finite state machine (or HMM).

Formally, a grammar model consists of three components: a set of terminals, a set of non-terminals and a set of production rules.

In the domain of video event modeling, grammars are used as follows: Terminals correspond to abstraction primitives. Similarly, non-terminals may correspond to semantic concepts (i.e. sub-events). Production rules in an event model
correspond to the semantic structure of the event. A semantic grammar event model makes use of these components to represent a particular event domain.

The recognition of an event, is reduced to determining whether a particular video sequence abstraction (sequence of terminals) constitutes an instance of an event. In formal grammar terminology, this process is called parsing. The particular set of production rules used in recognizing the event is called the parse.

For the classes of regular and Context Free Grammars (as defined by Chomsky’s hierarchy of grammar models (Chomsky, 1957)) efficient polynomial time algorithms exist for parsing (Earley, 1970).

Deterministic semantic grammar models have been used in several event domains including object manipulations (Brand, 1996) and two-person interactions (Ryoo and Aggarwal, 2006).

A straightforward extension allows probabilities to be associated with each production rule. Grammar models utilizing this extension, called stochastic grammars (or sometimes probabilistic grammars), can give a probability score to a number of legal parses. This extension provides this formalism a mechanism to deal with the uncertainty inherent in video events. The parsing algorithm for deterministic grammars has been extended to work for stochastic grammars with the same asymptotic time complexity (Stolcke, 1995).

Stochastic grammars have been used in event domains such as parking lot surveillance (Ivanov and Bobick, 2000), card game surveillance (Moore and Essa, 2001), complex task recognition (e.g. Japanese tea ceremonies ) (Yamamoto et al., 2006; Minnen, Essa, and Starner, 2003), complex motions (Lymberopoulos et al., 2006; Zhang, Huang, and Tan, 2008), and human actions (Ogale, Karapurkar, and Aloimonos, 2005).

It is interesting to observe that the event domains in which semantic grammar models are utilized, in general, contain more complex events whose variance in appearance is very large. As we have previously noted, it is the insertion of semantic knowledge into the structure of the event model that allows representation and recognition of these complex events.

Attribute grammars, introduced by Knuth (Knuth, 1968), formally associate conditions with each production rule. Each terminal has certain attributes associated with it, and the use of each production rule in a parse is conditioned upon these attributes. The conditions on each production rule introduce additional semantics into the event model and are specified using knowledge of the domain.

In video event understanding, attribute grammars have been used (Joo and Chellappa, 2006) to classify single-thread atomic object-based events in a parking lot surveillance event domain. If the production rule predicates are chosen to be probabilistic, attribute grammars can be considered a special case of stochastic grammars.
Table 3.1: An attribute grammar describing events in the parking lot scenario similar to (Joo and Chellappa, 2006). Terminals (lower-case letters) correspond to atomic sub-events and non-terminals (upper-case letters) correspond to events and composite sub-events. A location attribute(loc) is associated with each terminal. The notation \( X_i \) refers to the \( i \)th term on the right hand side of the production rule. Predefined semantic locations are defined by keywords such as BldgEnt (i.e. the entrance to the building) and FOV (i.e. field of view).

Table 3.1 shows an example attribute grammar describing events in a parking lot scenario. The start symbol is the non-terminal "PARKING_LOT". Two non-terminals corresponding to events are "DROPOFF" and "PICKUP". Terminals represent atomic sub-events. Examples of these are "person_appear", "car_disappear", and "carstart". A set of attributes is associated with each of these terminals. We can then condition each of the production rules on these attributes’ values. An example of this can be seen in the rule for "DROPOFF". The production rule requires the sequence: "CARSTAND" (a non-terminal representing the composite sub-event of a car standing in place), "person_appear", "person_disappear", "CARSTART" (a non-terminal representing the composite sub-event of car starting to move). Aside from enforcing this sequence the production rules also enforce the semantic constraint that the location where the car is standing be near the location where the person appeared. This is done by comparing the "loc" attribute associated with each component of the right hand side of the production (using the "Near" predicate). Although non-terminals do not have attributes explicitly associated with them it is straightforward to derive attributes from their composing terminals.

Due to the inherent non-sequential temporal relationships in many video events, particularly those defined using semantics, many works have attempted to introduce these relations into the grammar event models (Ivanov and Bobick, 2000; Moore and Essa, 2001; Ryoo and Aggarwal, 2006; Zhang, Huang, and Tan, 2006; Zhang, Huang, and Tan, 2008).

Learning of semantic event models including grammar models is a challenging problem. Although several works have explored the problem of automatically learning a grammar model for video event representation (Cho, Cho, and Um,
2004; Cho, Cho, and Um, 2006; Guerra-Filho and Aloimonos, 2006; Kitani, Sato, and Sugimoto, 2007), the event description and recognition in semantic terms afforded by grammar approaches can, generally, only be achieved through manual specification of the model using expert domain knowledge.

Grammar models are well adapted to represent sequence and hierarchical composition in video events. Long-term dependence is also straightforward to express in a grammar model. Stochastic grammars allow reasoning with uncertainty and error correction methods. Attribute grammars allow further semantic knowledge to be introduced into the parsing process. Temporal relations other than sequence are not naturally represented by the grammar model, though there have been extensions to allow capturing these relations to some extent. However, the representation of these complex relations is not straightforward in the grammar formalism and are more naturally represented in other semantic event modelling formalisms such as Petri Nets (see next section).

3.5.2 Petri Nets

The non-sequential temporal relations that define many video events require a formalism that captures these relations naturally. Furthermore, as we have seen with BNs and HMMs, graphical formalisms allow a compact visualization of our event model.

The Petri Nets (PN) formalism allows such a graphical representation of the event model and can be used to naturally model non-sequential temporal relations as well as other semantic relations that often occur in video events. A more detailed introduction to the Petri Net formalism is given in Chapter 4.

In video event understanding, PN event model approaches can generally be categorized into two classes: Object-based PNs and Plan-based PNs (Lavee et al., 2007). We distinguish these approaches by the design choices made in constructing the event model.

The Object-based PN event model is used to describe single and multi-thread composite object-based events. Tokens in the Object PN model correspond to video objects and their properties. Place nodes in the Object-based PN represent object states. Transition nodes represent either a change in an object state or the verification of a relation. The enabling rules of conditional transitions are conditioned only on the object properties of the tokens (representing objects) in their immediate input place nodes. Particular transition nodes can represent events of interest. Events are recognized when these particular transitions fire. The Object-based PN model has been used in the event domains of traffic (Ghanem et al., 2004a; Ghanem, 2007) and people (Borzin, Rivlin, and Rudzsky, 2007) surveillance. Figure 3.10 illustrates an example of the Object-based PN model applied to a left luggage scenario. Each token corresponding to a detected object is inserted
Figure 3.10: Example of an Object PN Model for the Left Luggage Domain.

into the model at the "root" node and propagated onward according to enabling rules on the transition nodes. Several events of interest in this domain are represented as transition nodes in this model. Some of these events may be viewed as sub-events as they must necessarily occur to reach a state that allows recognizing another event. An example of this is the "person_went_away_from_luggage" event which is a prerequisite for the "person_abandoned_luggage" event in the figure. Gray rectangles in the figure indicate stochastic timed transition whose parameters can be estimated from training data.

Plan-based Petri Nets are another approach to event modelling that represents each event as a "plan" of sub-events. Each event is represented as a plan,
a number of sub-events connected in such a way as to enforce the temporal relations between them. Each sub-event is represented by a place node and can be considered to be occurring when a token is in this place node (these nodes only have a one token capacity). Transitions between sub-events are conditioned on general abstraction properties instead of on specific properties linked to input tokens (objects) as in the Object PNs. An event is recognized when the “sink” transition of the plan fires. Unlike in Object-based PNs, Plan-based PNs require a separate model for each event.

Plan-based Petri-Net event models have been applied to several event domains including: parking lot surveillance (Castel, Chaudron, and Tessier, 1996), people surveillance (Albanese et al., 2008a), and complex gesture recognition (Nam, Wohn, and Lee-Kwang, 1999).

An example of a Plan-based PN model is illustrated in Figure 3.12 using the “Car is Parking” event. Our sub-events will be “Car enters parking lot”, “Car stops in parking spot”. These sub-events will be represented by place nodes. An initial transition node T1, intended to detect the first sub-event, requires an object to be a car and to be located in the parking lot, to fire. Once this sub-event is detected (the conditions on transitions T1 are met) a token is placed in the place node corresponding to the “Car enters parking lot” sub-event. The state of the PN model now indicates which part of the observation must be queried for the next sub-event, namely the properties described in the next transition node’s enabling rule, the proximity of the car to the parking spot and the car’s speed. Once the conditions on these properties (transition T2) are met a token will be placed in the place corresponding to the sub-event “Car stops in parking spot”. Once this sub-event occurs we can trivially declare that our event, “Car is Parking” has occurred. To maintain the reference to the object we utilize an extra “condition” place. This place ensures we are referencing the same object in each transition. A logical AND fragment enforces this constraint on the “sink” transition of the event model. It is worthwhile to note that even before the “sink” transition of the plan is reached, there is a semantic notion of how “close” the event is to completion.

In most known works employing PN models for the representation and recognition of video events an object-based abstraction is used (Ghanem et al., 2004b; Borzin, Rivlin, and Rudzsky, 2007; Albanese et al., 2008a; Castel, Chaudron, and Tessier, 1996). The high-level semantics captured by PN event models include temporal, spatial and logical composition, hierarchy, concurrency and partial ordering. PN event models are usually specified manually using knowledge of the domain. Sub-event temporal and logical relationships are related through known PN fragments which correspond to Allen’s temporal relations (Figure 3.11) and the three logical relations (AND, OR, NOT). The manual construction allows meaningful semantic concepts to be associated with the place and transition
nodes of the PN event model. The semantic nature of PN models makes learning these models from training data ill defined. This raises concerns about the scalability of this approach to larger problems than those illustrated in the various other works. Initial research has been done on translating standard knowledge specification formats for video events into PN event models (Lavee et al., 2007).

Another disadvantage of PN event models is their deterministic nature. A recurring criticism of the PN formalism for video event understanding is their reliance on an error-free ”perfect abstraction”, in contrast to probabilistic formalisms (e.g. BN) that can use their probabilistic mechanism to correct for these errors.

3.5.3 Constraint Satisfaction

Another approach to representation and recognition of a particular event domain in terms of semantic concepts and relations is to represent the event as a set of semantic constraints on the abstraction and to pose the problem of recognition as one of constraint satisfaction. An event is recognized by determining whether a particular video sequence abstraction is consistent with these constraints. One advantage of this approach is that the constraints can be formulated as an ontology for a particular event domain and reused in different applications.
Early work in constraint recognition introduced the notion of chronicles, undirected constraint graphs describing the temporal constraints of atomic sub-events (Dousson, Gaborit, and Ghallab, 1993; Ghallab, 1996). The event recognition task in these approaches is reduced to mapping the set of constraints to a temporal constraint network and determining whether the abstracted video sequence satisfies these constraints. While known algorithms exist to solve this problem, it is, in general, computationally intractable (NP-hard in the number of constraints). However, several event models have been proposed which approximate the solution by such methods as reducing the domain of each node (representing a sub-event) in the temporal constraint network (Pinhanez and Bobick, 1998) and eliminating arcs in the network with less relevance to the solution (Vu, Brémond, and Thonnat, 2002; Vu, Brémond, and Thonnat, 2003).

Vu et al (Vu, Brémond, and Thonnat, 2002; Vu, Brémond, and Thonnat, 2003) achieve a speed up of the algorithm that allows it to be used in real-time surveillance applications. Their method, coupled with an object-based abstraction, has been evaluated extensively in several event domains including airport surveillance (Fusier et al., 2007), home care applications (Zouba, Bremond, and Thonnat, 2008), and others (Vu, 2004).

In addition to temporal constraints, more recent work incorporates semantic knowledge about atemporal constraints pertaining to the properties of objects participating in the scene (Chelq and Thonnat, 1996; Rota and Thonnat, 2000a; Rota and Thonnat, 2000b; Vu, Bremond, and Thonnat, 2002; Vu, Brémond, and Thonnat, 2003). Description logics (Terzic, Hotz, and Neumann, 2007; Neumann and Möller, 2008) offer a very rich framework for representing video events including compositional hierarchy specification as well as semantic relationships. Learning of these description logic models has also been explored (Hartz and Neumann, 2007).
An object-based abstraction is often coupled with the constraint satisfaction event models (Chelq and Thonnat, 1996; Rota and Thonnat, 2000a; Rota and Thonnat, 2000b; Vu, Bremond, and Thonnat, 2002; Vu, Brémond, and Thonnat, 2003; Dousson, Gaborit, and Ghallab, 1993). Other works in constraint satisfaction event models assume a higher-level abstraction where a video sequence is described in terms of atomic sub-events (Pinhanez and Bobick, 1998; Dousson, Gaborit, and Ghallab, 1993).

3.5.4 Logic Approaches

The formalization of knowledge using logic is a well studied topic in artificial intelligence. The AI literature has proposed several works discussing how to specify the semantic knowledge of an event domain using well-studied logic. Many of these works discuss the specification of “event calculus” (Shanahan, 2000; Shanahan, 1990).

Only recently, however, have logic-based event models been introduced for video event understanding. In this type of event model knowledge about an event domain is specified as a set of logic predicates. A particular event is recognized using logical inference techniques such as resolution. These techniques are not tractable in general but are useful as long as the number of predicates, inference rules, and groundings (usually corresponding to the number of objects in the video sequence) are kept low.

Initial work applying the first-order logic framework of Prolog to recognition in the event domain of parking lot surveillance (Shet, Harwood, and Davis, 2005). All semantic relations may be modeled as predicates and their relationships may be specified using the structure of the event inference rules. This specification may be considered an event domain independent part of the model specification. That is, one part of the event model describes predicates and inference rules that define semantic relations, and another part defines predicates that use these semantic relations to describe the events in a particular event domain.

To cope with the uncertainty inherent in video events some extensions to logics approaches have been proposed including multi-valued logics (Shet, Harwood, and Davis, 2006), and Markov logics (Tran and Davis, 2008).

Logic approaches are a promising direction in event understanding. These kind of event models provide a robust representation and a well-understood recognition technique. It is has not been studied how this class of event models will scale up to problems with many inference rules. Furthermore, semantic relations must be modeled as part of the knowledge base and are not an intrinsic part of the model.
Chapter 4

The Petri Net Formalism

Petri Nets (PN) are a formalism useful for specifying complex temporal relations that has recently generated much interest in video event understanding research (Albanese et al., 2008a; Castel, Chaudron, and Tessier, 1996; Ghanem et al., 2004a; Lavee et al., 2010; Perše et al., 2010). Petri Nets have previously been used in modeling and analysis of complex systems.

4.1 Graphical Representation

The Petri Net model can be represented as a graph that has two node types (place and transition nodes) and two arc types (regular and inhibit arcs). In a graphical representation, the places are drawn as circles and the transitions are drawn as squares or rectangles. Arcs are drawn connecting place nodes to transition nodes (input arcs) or transition nodes to place nodes (output arcs). Regular arcs are drawn with arrow heads. Inhibit arcs are drawn with dot heads. Arcs are associated with a weight, also called the arc’s multiplicity. Places connected to a transition by input arcs are called that transition’s input places (or input set). Similarly, places connected to a transition by output arcs are called that transition’s output places (or output set). A place node may contain a number of tokens (another graph component). Tokens are visualized as black dots within the place node which contains them. Figure 4.1 illustrates a simple Example Petri Net.

At any given time, the state of the PN model is defined by the number of tokens in each of the model’s place nodes. This configuration is called a “marking”. The marking can be easily represented as a vector in which the $i$th entry corresponds to the number of tokens in the $i$th place.

Transition from one marking to another occurs when one or more transition nodes “fire”. Transition nodes can only fire once they become “enabled”. A transition’s “enabling rule” is considered satisfied when (1) all the input places
connected to the transition by regular arcs contain at least as many tokens as the multiplicity of the connecting arc and (2) all the input places connected to the transition by inhibit arcs contain a number of tokens strictly less than the multiplicity of the connecting arc. When the enabling rule is satisfied a transition is considered “enabled”. In extensions to the PN formalism, an enabling rule can be modified to contain conditions on the properties of tokens in the input places.

Once a transition node becomes enabled it may fire. The modification to the state of the PN model resulting from the firing of a transition is defined by a “firing rule”. A transition’s firing rule is defined as (1) removing a number of tokens from each of the transition’s input set places equal to the multiplicity of the connecting arc, and (2): creating a number of new tokens in each of the output set places equal to the multiplicity of the connecting arc. “Conditional transition” nodes have an enabling rule applied to them which imposes additional external conditions on the enabling of the transition.

Occasionally, more than one transition may become enabled at the same time. Depending on the PN structure, this situation can create a conflict, as firing of one transition may immediately disable another transition. Conflicts may be resolved in a controllable way by adding a priority parameter to each transition node. In case of conflict, the higher priority transition will always fire before the

Figure 4.1: An Example Petri Net
lower priority one.

One extension to the Petri Net formalism is the addition of timed transitions which have an associated duration parameter. A timed transition may only fire if the period of time since it became enabled is greater than this duration parameter (Marsan et al., 1995).

Another extension to the PN model that adopts timed transitions with stochastic distributions over the duration parameter is known as a Stochastic Petri Net (SPN). Each timed transition duration is modeled using the negative exponential probability density function. A parameter indicating the rate of the distribution (the inverse of the mean) is associated with each individual transition. A Generalized Stochastic Petri Net (GSPN) utilizes both immediate transitions and timed transitions (Marsan et al., 1995).

### 4.2 Marking Analysis

Prior to involving any PN model dynamics, it is possible to compute the set of all markings reachable from the initial marking and all the paths that the system may follow to move from state to state. The set of all reachable markings define the “reachability set” of the PN graph. The reachability graph consists of nodes representing reachable markings and arcs representing possible transition from one marking to another. An example of reachability graph is illustrated on Figure 4.2.

The reachability set represents all possible model states and the reachability graph represents all possible state transitions. However, the reachability graph is only finite if the model and the number of involved tokens are finite. In practice, this means that it is possible to calculate the full reachability graph only if we
know the maximum number of the expected tokens in the system. This information is not always available which limits the extent a reachability graph of this sort can be used.
Chapter 5

The Particle Filter

In a Bayesian approach to analyzing dynamic systems, the goal is to estimate the posterior distribution $P(x_t|y_{1:t})$ over the system state at time $t$, denoted $x_t$, taking into account all observations up to the current time, denoted $y_{1:t}$. The Bayesian Recursive Filter (BRF) is an approach appropriate for online problems in which $P(x_{t-1}|y_{1:t-1})$, the previous estimation of the posterior, is used in a recursive fashion to derive an updated estimation with each new observation.

The BRF framework consists of two major components. The dynamic model, denoted $P(x_t|x_{t-1})$, describes the evolution of states over time. The measurement model, denoted $P(y_t|x_t)$, specifies the relationship between state and observation variables. After initialization a BRF approach proceeds in two stages, prediction and correction. In the prediction stage, the previous estimation of the posterior is used to arrive at an estimation of possible future states.

Particle filters (PF), also known as sequential Monte Carlo methods, are techniques for probability density estimation within the BRF framework based on sampling. Particle filter algorithms have previously been used in many application fields including visual tracking of objects (Isard and Blake, 1998).

The PF approach maintains a set of $N$ hypotheses of the current state called particles, denoted $X_t = \{x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(N)}\}$. Each particle $x_t^{(i)}$ is associated with a weight, denoted $w_t^{(i)}$. The weights are used to approximate the posterior distribution as follows:

$$P(x_t|y_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \delta(x_t, x_t^{(i)}) \tag{5.1}$$

where $\delta$ denotes the Dirac delta function.

The commonly used Sequential Importance Sampling (SIS) algorithm (illustrated in Figure 5.1) of the particle filter proceeds as follows: In the prediction phase the algorithm samples a new set of particles using the current particle
An illustration of the SIS particle filtering algorithm. The algorithm proceeds by sampling from an approximation of the posterior in the prediction step. These samples are then reweighted according to their consistency with the observation.

Each particle at time $t$, denoted $\mathbf{x}_t^{(i)}$, is sampled from the proposal (or importance) distribution, denoted $\pi(\mathbf{x}_t^{(i)} | \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t})$, used as an approximation to the posterior distribution, which is often difficult to sample from directly. In the correction stage, the algorithm uses the most recent observation, denoted $\mathbf{y}_t$, to adjust the weights of particles sampled in the prediction phase, increasing (decreasing) the weight of states consistent (inconsistent) with the observation. The weights of each particle value are multiplied by the ratio of the posterior distribution (evaluated at the particle value) to the proposal distribution as follows:

$$\hat{w}_t^{(i)} = w_{t-1}^{(i)} \cdot \frac{p(\mathbf{y}_t | \mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)} | \mathbf{x}_{1:t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)} | \mathbf{x}_{1:t-1}^{(i)}, \mathbf{y}_{1:t})}. \quad (5.2)$$

The proposal distribution is often chosen to both allow simplification of this ratio and to allow straightforward sampling. One common choice for the proposal distribution is the dynamic model, $P(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)})$, which simplifies the above equation to:

$$\hat{w}_t^{(i)} = w_{t-1}^{(i)} \cdot p(\mathbf{y}_t | \mathbf{x}_t^{(i)}) \quad (5.3)$$

The sum over the weights of particles in a particular state is the probability mass of that state. The probability mass of all states is normalized to sum to one,
and thus defines a probability distribution over the state space (approximating the posterior distribution).

In the commonly used Sequential Importance Resampling particle filter algorithm, the particles are resampled from the probability distribution implied by the probability mass of each of the states, if the effective number of particles (the inverse of the sum of squared weights), becomes too small.

The Particle Filter framework is used in our approach to event recognition and representation in Chapters 8 and 9 to enable probabilistic online reasoning when available sub-event are only observed up to some level of certainty.
Chapter 6

Modeling Events with Generalized Stochastic Petri Nets

In this chapter we narrow our focus from mapping the space of video event understanding to describe our own contribution to this space. Our early work was focused on extending the Object-based PN paradigm for modeling variance in surveillance video events.

Multi-thread complex events are difficult to define by appearance features. Rather, they are defined by atomic sub-events restricted to temporal intervals and the interval temporal relationship between those sub-events. In essence it is semantic knowledge that defines these events. Here we propose the SERF-PN (Surveillance Event Recognition Framework using Petri Nets) framework for both modeling semantic knowledge, and utilizing this knowledge in recognition of events of interest in unlabeled video. Using the statistics gathered from observing a set of training data, SERF-PN is also able to make assertions on the likelihood of future events.

The major novelties of this approach are extensions to both the modeling and the recognition capacities of Object-based PN paradigm (See section 3.5.2). We first extend the representational capacities by introducing stochastic timed transitions to allow modeling of events which have some random variance in duration. A second representational novelty is the use of a single PN to represent the entire event domain, as opposed to previous approaches which have utilized several networks, one for each event of interest. A third contribution is the capacity to probabilistically predict future events by constructing a discrete time Markov chain model of transitions between states. Of course, it is important that any algorithmic approach to recognition of events in video be computationally tractable, to this end we provide a formalization and complexity analysis of our proposed algorithm.


6.1 Video Event Representation in SERF-PN

Our representation model follows the Object-Based PN paradigm. Within this framework, PN components have the following roles:

- Tokens represent actors or static objects detected in the video sequence. Each token has a set of properties associated with it corresponding to object properties.
- Places represent the possible object states. Each place containing more than one token indicates a group of objects in the same state.
- Transitions represent video events that define dynamics of the event model. Transition node firing can be equivalent to the object state change in the real scene or can be the result of a satisfied relation constraint.

In SERF-PN, tokens are associated with properties such as tracking and classification information provided by an intermediate video processing layer. Enabling rules may require specific values for token properties (e.g. movement, position, appearance, etc.) or enforce relations between tokens participating in the transitions. More exactly, a transition is enabled if the following is satisfied:

- Each input place connected to the transition by a regular arc contains token(s) that satisfy enabling rule conditions and the number of such tokens is greater than or equal to the multiplicity of the arc connecting that place to the transition.
- Each input place connected to the transition by an inhibit arc contains a number of tokens is strictly less than the multiplicity of the arc connecting that place to the transition.

SERF-PN does not modify the firing rule definition. That is, firing transitions delete from each input place as many tokens as the input arc multiplicity and then add to each output place as many tokens as the output arc multiplicity. State changes in SERF-PN are usually the result of the initiation or completion of some event or the verification of a condition. SERF-PN utilizes a Generalized Stochastic Petri Net (GSPN) model which includes immediate transitions as well as stochastic timed transitions whose firing delays are distributed according to the negative exponential distribution. Generally, time consuming events corresponding to real world occurrences will be modeled as timed transitions while condition or relation verifications will be modeled as immediate transitions. Examples of these modeling practices are given in the section 6.3.
6.2 Learning the System Parameters

In SERF-PN, the structure of the model is static and designed manually using knowledge of the domain and of the PN formalism by a knowledge engineer. Parameters of the system, however, are dynamic and may be learned to assist us in predicting events based on the current system state. One set of these parameters include duration parameters (i.e. exponential distribution rates) for each of the timed transition in the PN model. Another set of parameters are the transition likelihoods between directly reachable markings.

To learn these parameters we require a set of examples which cover the typical occurrences in the domain of interest as well as a specification of our PN event model. In most surveillance tasks the existence of such a set of examples is a realistic assumption due to the repetitive nature of many events in the surveillance domain. Note also, that the knowledge used by the engineer in constructing the static structure of the model is also based on experience with typical events.

6.2.1 Timing Parameters

The probability density of the firing delay of each timed transition is given by the PDF function:

\[ D_n = 1 - e^{-\frac{t_n}{\mu_n}} \]  

where \( t_n \) is an enabling period of timed transition \( n \) and \( \mu_n \) is an average delay of timed transition \( n \). During the training process we wish to estimate the \( \mu_n \) parameters for each transition \( n \). This is achieved by not allowing timed transitions to fire and calculating the average time each of these transitions is enabled over the course of the training sequence. Once these parameters are learned the system can model the occurrence of a timed event relative to the training data.

6.2.2 Marking Transition Probabilities

The combination of the PN model and the initial marking allows the construction of a reachability graph for the domain. Each marking node in this reachability graph represents a legal state of the system. An outgoing link from marking node, \( M_1 \), to another marking node, \( M_2 \), indicates that marking \( M_2 \) is directly reachable from the marking \( M_1 \).

The reachability graph defines the space of possible states within the model. The probability mass in this space, however, is not uniformly distributed. In fact, the majority of the probability mass is concentrated in a just a few markings relative to the large number of possible markings. In other words, while the
theoretical number of possible markings is large, in practice only a small subset of those markings are observed.

More specifically, the possible number of markings for a PN model with \( n \) place nodes and \( t \) tokens is given by the recursion formula:

\[
\Pi(t, n) = 1 + \sum_{i=1}^{t} \Pi(i, n - 1)
\]

(6.2)

where the base case is:

\[
\Pi(t, 1) = 1
\]

(6.3)

The probability distribution over the marking set can be estimated using observed data, which may be viewed as a sample from this distribution. This is contingent upon the sample being representative of the underlying distribution, which we have claimed is a reasonable assumption in surveillance applications.

Another simplifying assumption made at this stage is the Markov assumption, which states that at each discrete time slice the current marking depends only on the previous marking. This allows us to construct a discrete time Markov Chain (DTMC) representing the joint probability of a sequence of particular markings. This formulation may be used to predict future states or answer queries on the probabilistic distance from a particular marking.

Our proposed approach is to construct this DTMC dynamically from the training data. Again, we assume that the training sequence is a representative sample of the underlying probability distribution over event patterns. Thus the training sequence covers the significant reachable markings in this domain. Each encountered marking in the training sequence corresponds to a state the Markov chain may take on. The probabilities for state transitions in the Markov chains are then estimated directly from the training data according to the formula:

\[
\lambda_{n,k} = \frac{N_{n,k}}{N_n}
\]

(6.4)

where \( \lambda_{n,k} \) is probability to move to marking \( M_k \) from marking \( M_n \), \( N_{n,k} \) is number of detected transitions from marking \( M_n \) to marking \( M_k \), and \( N_n \) is number of \( M_n \) marking occurrences.

### 6.3 Constructing a PN model: Illustrative examples

In this section we provide two comprehensive examples of using the qualities of the GSPN formalism described in previous sections to construct a model describing the semantics in a particular event domain. Our construction remains
firmly within the Object PN paradigm. That is, each token corresponds to a scene object, each place node corresponds to an object state and each transition corresponds to a change in object state or a relation verification.

It is important to note that a PN model structure along with the conditions defined on each of the transitions within the model is a reduction of a human decision process in a particular event domain. As such, it may be possible to define several different models that provide a reasonable semantic summary of an event domain. Determining if a particular model is sufficiently accurate can be done empirically by comparing the results of the model applied to a particular video interpretation to the interpretation provided by a human observer.
<table>
<thead>
<tr>
<th>Transition</th>
<th>Name</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Person_Appeared</td>
<td>Token.Appearance='appear'</td>
</tr>
<tr>
<td>T2</td>
<td>Person_Disappeared</td>
<td>Token.Appearance='disappear'</td>
</tr>
<tr>
<td>T3</td>
<td>Guard_Post_Manned</td>
<td>Token.loc overlaps &quot;Guard Post&quot; Zone</td>
</tr>
<tr>
<td>T4</td>
<td>Guard_Post_Unmanned</td>
<td>Token.loc not overlaps &quot;Guard Post&quot; Zone</td>
</tr>
<tr>
<td>T5</td>
<td>Visitor_Not_Checked</td>
<td>Token.loc overlaps &quot;Not Checked&quot; Zone</td>
</tr>
<tr>
<td>T6</td>
<td>Visitor_Entered</td>
<td>Token.loc not overlaps &quot;Guard Post&quot; Zone</td>
</tr>
<tr>
<td>T7</td>
<td>Guard_Met_Visitor</td>
<td>distance(Token(p5).loc,Token(p6).loc) &lt; T_{meeting_start}</td>
</tr>
<tr>
<td>T8</td>
<td>Meeting_Is_Over</td>
<td>distance(Token1(p0).loc,Token2(p0).loc) &gt; T_{meeting_end}</td>
</tr>
<tr>
<td>T9</td>
<td>Security_CheckToo_Long</td>
<td>distance(Token1(p0).loc,Token2(p0).loc) &lt; T_{meeting_end} for a period exceeding a value sampled from D_0</td>
</tr>
<tr>
<td>T10</td>
<td>Meeting_Is_Over(after Delay)</td>
<td>distance(Token1(p0).loc,Token2(p0).loc) &gt; T_{meeting_end}</td>
</tr>
<tr>
<td>T11</td>
<td>Guard_Post_Manned(After Check)</td>
<td>Token.loc overlaps &quot;Guard Post&quot; Zone</td>
</tr>
<tr>
<td>T12</td>
<td>Visitor_Continues(After Check)</td>
<td>Token.loc not overlaps &quot;Guard Post&quot; Zone</td>
</tr>
<tr>
<td>T13</td>
<td>Person_Disappeared(After Check)</td>
<td>Token.Appearance='disappear'</td>
</tr>
</tbody>
</table>

**Table 6.1:** Elaboration on the model structure pictured in figure 6.1. Each transition is given a meaningful semantic name and associated with a condition on the properties of its input tokens. Token(p_x) denotes a token whose origin is in place p_x. In situations when a transition has only one input place node, this notation is dropped and we simply write Token. In the case of multiple tokens from the same place node, we denote these Token1(p_x), Token2(p_x). Token.attributex denotes the value of attributex in the particular input token. In this example we use the attributes "loc" (token location) and "Appearance", (which can take on the values {"appear","visible","disappear"}). See text for details of this construction.
6.3.1 Security Check Scenario

In this section we consider the scenario where a single person guards the entrance to a particular location and is tasked with checking the bags of all those who enter the area in a time-efficient manner. We are particularly interested in two main events:

- A visitor has entered the area without being checked
- The duration of the security check is excessively long.

To describe this scenario using the PN formalism we define two semantic areas within our camera view: the ”Guard Post” zone and the ”Not Checked” zone. The former zone is rather self-explanatory, it is the area in the camera view taken by the guard. The latter zone refers to an area in the camera view to which a visitor should only gain access after being checked.

All objects detected by the system are mapped to tokens and each of the object’s attributes are associated with the corresponding token. In the remaining part of this discussion we will refer to objects and tokens interchangeably. Initially tokens are placed in the “Root” place node. The transition “Person Appeared” becomes enabled when any tokens that meet its enabling condition are present in the “Root” place node. Specifically, the “Person Appeared” transition is conditioned on the ”Appearance” attribute of its input tokens and will be allowed to fire if this attribute takes on the value ”appear”. Upon firing of this transition the corresponding input tokens will be removed from the “Root” place node and placed in the “P3” place node (see figure 6.1).

The “Guard Post Manned” transition is similarly conditioned on the “loc” (object location) attribute of its input tokens. If an input token located in the “Guard Post” Zone exists, the transition will be allowed to fire. The transitions “Visitor Entered”, and “Visitor Not Checked” are similarly defined by conditions tied to the location attribute.

The “Guard Met Visitor” transition requires input tokens in two separate place nodes (representing the guard and the visitor). It also has a condition on the distance between these two tokens’ locations. If this distance is below a certain threshold , denoted $T_{meeting,start}$, this transition is allowed to fire. In firing, it moves both of its input tokens to its output place node, “P0”.

Transitions “Meeting Is Over” and “Meeting Is Over(After Delay)” are defined similarly with a maximum distance threshold, denoted $T_{meeting,end}$.

Transition “Security Check Too Long” is a stochastic timed transition with parameter $\mu_9$ which indicates the average firing delay of the transition. This transition becomes enabled when two tokens are in input place “P0’ (the two token requirement is captured by assigning a multiplicity of two to the input arc of this transition). Once the enabling condition is met the transition may only fire
once a period of time, sampled from the distribution $D_9$ (which is determined by parameter $\mu_9$), has elapsed. Note also that this transition becomes enabled at the same time as the transition “Meeting Is Over”. The fact, that firing one of these transitions will disable the other, is called a conflict in Petri Net terminology. Our condition on the length of the meeting uses this. That is, once the security check has begun we will only allow a specific time interval (determined by parameter $\mu_9$) to elapse before alerting that the security check has gone on too long. Of course if the meeting ends before this interval elapses, we do not need to provide such an alert.

The remaining transitions “Guard Post Manned(After Check)”, “Visitor Continues(After Check)”, “Person Disappeared(After Check)” are immediate conditional transitions conditioned on the “loc” and “Appearance” attributes of their input tokens.

The PN graph corresponding to this model is illustrated in Figure 6.1. The transition names and conditions are enumerated in Table 6.1.

### 6.3.2 Traffic Intersection Scenario

In this section we consider a four-way intersection with no traffic light or stop sign. Cars must mind oncoming traffic before entering the intersection. More specifically, cars that wish to turn in the face on oncoming traffic must do so a reasonable time before the oncoming traffic enters the intersection. We have generalized this model to also include cars crossing the intersection.

Of primary concern to us in this model:

- An interaction between two cars during which one of the cars is directly in the path of the other.

We shall use the orientation attribute of the objects to determine if their trajectories are on a collision course (i.e. one car is in the path of another). Additionally, we shall model a “reasonable time” using a stochastic timed transition. Furthermore, we shall define five zones of interest, an “Approach” zone from each direction surrounding the intersection and the “Intersection” zone. These zones will describe which part of the camera view is taken up by the associated semantic area.

Similarly to the first example, all detected object tokens are placed in the “Root” place node initially. The “Car Appeared” transition will move these tokens into the “P1” place node if their “Appearance” attribute takes on the value “appear”. The “Car Approaching” set of transitions (T2{a-d} in the figure) will compare the ”loc” attribute (i.e. object location) to each of the four “Approach” Zones defined and fire if the ”loc” value is found to be within one of these.
The stochastic timed transition “Safe Interaction Occurred” becomes enabled when there exists a token in each of the two place nodes “P2” and “P3” (representing a car inside the intersection and a car approaching the intersection, respectively.) If this transition remains enabled for a an interval of time, whose length is sampled from distribution $D_5$ (determined by parameter $\mu_5$), it will allowed to fire. Again, we place our timed transition in conflict with an immediate transition (two in this case). In this model, the firing of either the the “Car Entered Intersection” transition or the “Car Leaving Intersection” will disable the timed transition. These transitions check if the “loc” attribute is inside/outside the “Intersection” zone.

The “Unsafe Interaction Occuring” transition has an input arc of multiplicity two. This means at least two tokens must exist in its input place node, “P3”, for this transition to become enabled. Furthermore it has a condition on the difference in “orientation” attribute values of each of the tokens participating in the enabling. The “orientation” attribute values must be perpendicular (a difference of $90 \pm 15$ degrees) for this transition to fire.

If the “reasonable” amount of time has elapsed between the time a particular car has entered the intersection in the path of another oncoming car (i.e. the transition “Safe Interaction Occurred” fires), our knowledge of the event domain states that an unsafe situation is less likely to occur and hence we can allow perpendicular trajectories within the intersection (e.g. a car is existing to the
<table>
<thead>
<tr>
<th>Transition</th>
<th>Name</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Car_Appear</td>
<td>Token.Appearance=&quot;appear&quot;</td>
</tr>
<tr>
<td>T2{a,b,c,d}</td>
<td>Car_Approaching_Intersection</td>
<td>Token.loc overlaps one of 4 &quot;Approach&quot; Zones</td>
</tr>
<tr>
<td></td>
<td>(from {North,South,East,West})</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>Car_Entered_Intersection</td>
<td>Token.loc overlaps &quot;Intersection&quot; Zone</td>
</tr>
<tr>
<td>T4</td>
<td>Unsafe_Interaction_Occuring</td>
<td>Token1(p3).orientation is perpendicular to Token2(p3).orientation</td>
</tr>
<tr>
<td>T5 (timed)</td>
<td>Safe_Interaction_Occurred</td>
<td>Token(p2).orientation is perpendicular to Token(p3).orientation for a period exceeding a value sampled from $D_5$</td>
</tr>
<tr>
<td>T6</td>
<td>Car_Leaves_Intersection</td>
<td>Token.loc not overlaps &quot;Intersection&quot; Zone</td>
</tr>
<tr>
<td>T7</td>
<td>Car_Continues (After Safe Interaction)</td>
<td>Token.loc not overlaps &quot;Intersection&quot; Zone</td>
</tr>
<tr>
<td>T8</td>
<td>Car_In_Intersection (After Safe Interaction)</td>
<td>Token.loc &quot;Intersection&quot; Zone</td>
</tr>
<tr>
<td>T9</td>
<td>Car_Leaves_Intersection (After Safe Interaction)</td>
<td>Token.loc not overlaps &quot;Intersection&quot; Zone</td>
</tr>
<tr>
<td>T10</td>
<td>Car_Disappears</td>
<td>Token.Appearance=&quot;disappear&quot;</td>
</tr>
</tbody>
</table>

**Table 6.2:** Elaboration on the model structure pictured in figure 6.2. See figure 6.1 caption for explanation of notation. In addition to the “loc” and “Appearance” attributes used in the previous example, this example also uses the “orientation” attribute which gives the direction of the object’s velocity vector. See text for details of this construction.

The remaining transitions “Car_Continues(After Safe Interaction)”, “Car_Leaves_Intersection(After Safe Interaction)” and “Car_Disappears” are conditioned on the “loc” and “Appearance” attributes.

The PN graph corresponding to this model is illustrated in Figure 6.2. The transition names and conditions are enumerated in Table 6.2.
6.4 Complexity Analysis

In this section we aim to give a formal treatment to our algorithm for event recognition using the Petri Net modeling formalism. To this end we must define some notations and concepts. The notations described in the following are based on notations in (Albanese et al., 2008b; Marsan et al., 1995).

First, we will define a video sequence as the set of frames \( F \), the function \( \rho \) which maps a particular frame to a set of objects in the video, and the function \( l \) which maps a particular set of objects to a set of features. An example of possible features would be the CAVIAR format features which include continuous valued variables \( h, w, x, y \) (respectively representing the height, width, and centroid coordinates of the object bounding box) as well as discrete finite-domain categorical variables such as movement, role, situation, and context. Essentially the \( l \) function allows us to retrieve the semantic properties of each object (or set of objects).

A Petri Net event model can be described using \( P \), the set of all places and \( T \) the set of all transitions. Additionally, the function \( \text{flow} \) indicates the arc flow of the network. That is, if \( y \in \text{flow}(x) \) there is an arc from node \( x \) to node \( y \). The \( \delta \) function describes the condition associated with each transition, by associating each transition \( t \) with a set of possible feature values. That is a set of objects, \( p \), whose labels \( l(p) \) are a subset of \( \delta(t) \) can be said to fulfill the condition of transition \( t \). Function \( r \), returns the arity of a particular transition. The arity is defined as the number of tokens to which the transition’s condition is applied. For example, a transition which verifies a single object(token) such as (appearance = ‘visible’) has an arity of one. Similarly a transition that verifies a relationship between two objects (tokens) , such as (distance(\( t1, t2 ) > \text{thresh} ) has an arity of two. Finally, the variable rootnode indicates the ‘root’ place node in which tokens corresponding to new objects are initially placed.

Table 6.3 summarizes the notation used in this section.

The algorithm for detecting a particular event in a video sequence is presented in pseudo code in Algorithm 1. The algorithm accepts as input the video sequence and PN model parameters, as well as the variable event transition, which indicates which transition corresponds to the particular event we are interested in detecting. After initialization of temporary variables, the algorithm proceeds to loop over all frames in the video. At each frame the algorithm assigns tokens in the network to each object, determines which transitions are enabled, and loops over these enabled transitions to determine if their conditions for firing have been met. In order to make this determination, the algorithm must check each possible combination of tokens of size \( y \) in the input set of each transition, where \( y \) is the arity of the transition’s condition. Thus, we have a loop over all possible (size \( y \)) combinations of tokens nested inside a loop over all enabled transitions, nested
inside a loop over all frames in the video sequence. The number of outer loops is bounded by $|F|$, the number of enabled transitions is bounded by the total number of transitions $|T|$ which is a known small constant relative to the number of frames in the average input sequence. The number of token combinations of size $y$ is given by $\binom{k}{y}$, where $k$ is the number of tokens in the transition’s input set. Clearly $k$ is bounded by $|K| = |O|$, the number of objects/tokens in our net. $y$ in this expression is bounded by $Y = \max_{t \in T} \{r(t)\}$. Thus the worst case complexity of this algorithm is given by $O(|F| \cdot \binom{|K|}{Y})$. That is, the running time is dependent on the number of objects and the maximal arity of the transition conditions. In practice, however the number of objects simultaneously appearing in a frame is very small compared to the number of frames in the video sequence. In our experiments the maximal number of objects in a single frame was 7. On average this number is significantly less than this. Similarly, when

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>set of places</td>
</tr>
<tr>
<td>$T$</td>
<td>set of transitions</td>
</tr>
<tr>
<td>$S$</td>
<td>set of features</td>
</tr>
<tr>
<td>$flow : P \cup T \rightarrow 2^{P \cup T}$</td>
<td>function indicating the direction of arc flow</td>
</tr>
<tr>
<td>$\cdot x = {y</td>
<td>x \in flow(y)}$</td>
</tr>
<tr>
<td>$x = {y</td>
<td>y \in flow(x)}$</td>
</tr>
<tr>
<td>$\delta : T \rightarrow 2^S$</td>
<td>associates each transition with a set of features values needed to fire</td>
</tr>
<tr>
<td>$r : T \rightarrow \mathbb{N}$</td>
<td>function from a transition to their condition arity</td>
</tr>
<tr>
<td>$\text{rootnode}$</td>
<td>place in which tokens are initially placed</td>
</tr>
<tr>
<td>$F$</td>
<td>set of frames</td>
</tr>
<tr>
<td>$O$</td>
<td>set of objects</td>
</tr>
<tr>
<td>$\rho : F \rightarrow 2^O$</td>
<td>set of objects in frame</td>
</tr>
<tr>
<td>$l : 2^O \rightarrow 2^S$</td>
<td>function from set of objects to labels</td>
</tr>
<tr>
<td>$K$</td>
<td>set of tokens</td>
</tr>
<tr>
<td>$\mu : P \rightarrow 2^K$</td>
<td>mapping from p to a set of tokens</td>
</tr>
<tr>
<td>$g : K \rightarrow O$</td>
<td>function from tokens to objects</td>
</tr>
<tr>
<td>$f : O \rightarrow K$</td>
<td>function from objects to tokens</td>
</tr>
<tr>
<td>$G(i, X) = {x_{r_1}, x_{r_2}, ..., x_{r_i}</td>
<td>x_{r_1}, x_{r_2}, ..., x_{r_i} \in X, r_1 &lt; r_2 &lt; ... &lt; r_i}$</td>
</tr>
</tbody>
</table>

Table 6.3: Notation used in formal description of algorithm. $2^X$ denotes the powerset of set $X$. 

68
### Algorithm

**Input:**  
- $F, I, \rho$ //video and object labeling  
- $P, T, flow, \delta, r, rootnode$ //PN model  
- $\text{event}_\text{transition}$ //transition corresponding to particular event we are interested in

**Output:** `true` if event occurs in video sequence, `false` otherwise

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:</td>
<td>//Initialization:</td>
<td></td>
</tr>
<tr>
<td>02:</td>
<td>$K \leftarrow \emptyset$</td>
<td></td>
</tr>
<tr>
<td>03:</td>
<td><strong>for all</strong> places $p$ in $P$ <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>04:</td>
<td>$\mu(p) \leftarrow \emptyset$</td>
<td></td>
</tr>
<tr>
<td>05:</td>
<td><strong>end</strong></td>
<td></td>
</tr>
<tr>
<td>06:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07:</td>
<td><strong>for all</strong> frames $f$ in $F$ <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>08:</td>
<td>//Add tokens for appeared objects:</td>
<td></td>
</tr>
<tr>
<td>09:</td>
<td><strong>for all</strong> objects $o$ in $\rho(f) \setminus (K \cap \rho(f))$ <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>$k \leftarrow \text{new token}$</td>
<td></td>
</tr>
<tr>
<td>11:</td>
<td>$K \leftarrow K \cup {k}$</td>
<td></td>
</tr>
<tr>
<td>12:</td>
<td>$g(k) \leftarrow o$</td>
<td></td>
</tr>
<tr>
<td>13:</td>
<td>$f(o) \leftarrow k$</td>
<td></td>
</tr>
<tr>
<td>14:</td>
<td>$\mu(rootnode) \leftarrow \mu(rootnode) \cup {k}$</td>
<td></td>
</tr>
<tr>
<td>15:</td>
<td><strong>end</strong></td>
<td></td>
</tr>
<tr>
<td>16:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:</td>
<td>//Determine enabled transitions:</td>
<td></td>
</tr>
<tr>
<td>18:</td>
<td>$\text{enabled}_\text{transitions} \leftarrow \emptyset$</td>
<td></td>
</tr>
<tr>
<td>19:</td>
<td><strong>for all</strong> transitions $t$ in $T$ <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>20:</td>
<td>$\text{enabled} \leftarrow true$</td>
<td></td>
</tr>
<tr>
<td>21:</td>
<td><strong>for all</strong> places $p$ in $\cdot t$ <strong>do</strong></td>
<td></td>
</tr>
<tr>
<td>22:</td>
<td>if $\mu(p) = \emptyset$</td>
<td></td>
</tr>
<tr>
<td>23:</td>
<td>$\text{enabled} \leftarrow false$</td>
<td></td>
</tr>
<tr>
<td>24:</td>
<td><strong>end if</strong></td>
<td></td>
</tr>
<tr>
<td>25:</td>
<td><strong>end</strong></td>
<td></td>
</tr>
<tr>
<td>26:</td>
<td>if $\text{enabled} = true$</td>
<td></td>
</tr>
<tr>
<td>27:</td>
<td>$\text{enabled}<em>\text{transitions} \leftarrow \text{enabled}</em>\text{transitions} \cup t$</td>
<td></td>
</tr>
<tr>
<td>28:</td>
<td><strong>end if</strong></td>
<td></td>
</tr>
<tr>
<td>29:</td>
<td><strong>end</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Table 6.4:** Algorithm 1: detecting a particular event in a video sequence
designing our PN event models we seldom used transition conditions greater than two (e.g. Guard_Met_Visitor). Most often, we made use of transition conditions with an arity of one (e.g Car_Entered_Intersection). Thus, under the assumption that both \(|K| = |O|\) and \(Y\) are such small numbers compared with the length of the video sequence in frames we can make the claim that the algorithm’s time complexity is given by \(O(|F|)\). That is, the algorithm is linear in the length of the video sequence. Such assumptions has been previously made for complexity analysis in other works in the domain of video event analysis (Vu, 2004).

The complexity of related task including learning the timing parameters, learning marking transitions, detecting multiple events of interest, and counting events of interest can be accomplished in the same time complexity as above.

For the learning of timing parameters task we add a counter that keeps track of how long each timed transition is enabled. Instead of firing timed transitions when their conditions are met (line 41 in the algorithm) we increment this counter. The few additional lines of code needed to average these duration values would also have linear complexity in the number of frames in the worst case (i.e. the transition cannot be activated more times than there are frames in the video sequence).

The learning of marking transitions can be done by adding a few additional lines to our algorithm to keep track of visited markings. We would then add some lines after the firing of an enabled transition to determine if the resulting marking is new and update the transition probabilities from the previous marking to the current marking. This requires us to loop over all markings we have previously seen. This number is bounded by the total number of markings which is given by \(|P|^{|K|}\) where \(|P|\) is the number of places in the net and \(|K|\) is the number of tokens. This makes this problem complexity exponential in the number of tokens/objects in the scene. However, again applying our assumptions on a low \(|K|\) relative to \(|F|\) brings us back to linear time complexity in \(|F|\).

To check for more than one event we can generalize the event_transition input variable to a set of transitions and modify line 53 accordingly. To count the number of occurrences of a particular event(set of events) we simply add a counter (counters) variable and increment this counter in line 54 (instead of return true). We would also modify the final line to return this counter (counters) variable.
Algorithm ctd.

30: for all $t$ in enabled_transitions do
31:   $Q = \mu(t)$ // all tokens in $t$’s input set
32:   $H = \emptyset$
33:   for all $q$ in $Q$ do
34:     $H \leftarrow H \cup g(q)$ // $H$ is the set of all
35:     // objects corresponding to tokens in $t$’s input set
36:   end
37:   $y \leftarrow r(t)$ // arity of transition $t$’s condition
38:   for all subsets of objects $p$ in $G(y, H)$ do
39:     if $l(p) \subseteq \delta(t)$ // objects meet transition condition
40:       // fire transition
41:         for all objects $o$ in $p$ do
42:           $k \leftarrow f(o)$
43:             for all places $m$ in $\cdot t$ do
44:               if $k \in \mu(m)$
45:                 $\mu(m) \leftarrow \mu(m) \setminus \{k\}$
46:                 break
47:             end if
48:           end
49:         end
50:       end
51:     end
52:   end
53:   if $t = event\_transition$ // fired transition corresponds to event we are interested in
54:     return true
55:   end if
56: end
57: end
58: return false

Table 6.5: Algorithm 1 continued
Chapter 7

Surveillance using SERF-PN

The SERF-PN system described in the previous chapter was designed to aid in the human task of monitoring surveillance video. As such, given a description of the events of a surveillance domain (in a PN format) and an unlabeled (annotated) video sequence, the system should be able to notify when these events have occurred in the video sequence as well as predict the probable next event (or alternately provide the probability of an undesirable event). For this process to occur we must allow our system to (1) accept video sequences annotated with mid-level semantic information by an intermediate video processing layer. (2) Allow the user to input an event description in a PN format. (3) Train the specific parameters of the representation to apply to a particular scenario from training data (section 6.2). (4) Run on unlabeled (annotated) video data and give descriptions of events as they occur as well as predict future events. These processes are described below. A schematic of the SERF-PN components is shown in Figure 7.1.

7.1 Generating Video Annotation

The event analysis of the SERF-PN system is predicated upon video which has been annotated with mid-level semantic concepts such as object locations, trajectories and classes. Many works in object detection, tracking, and classification exist that focus on achieving this annotation from raw video data. In this work we have put the focus on event analysis and representation and have made use of existing low-level methods to create this annotation from real video sequences. This is referred to as the intermediate video processing unit.

To make our system modular to the process that created the video annotation we have made the video event recognition component of SERF-PN fully compliant with the CAVIAR (Fisher, 2004b) standard for annotation of video. Thus, the CAVIAR dataset (which has been manually annotated) and any other dataset in
this format can be easily evaluated by our system.

The standardization of the video annotation format also allows the use of a synthetic video tool called Video Scene Animator. This module takes object parameters as input and generates a CAVIAR compatible annotated video. This tool has been used to efficiently simulate real video sequences captured by a video camera, allowing us to extend the available dataset of real video clips. Furthermore, to train the parameters of the event model a large volume of ”normal” video sequences are needed. Video Scene animator, running in scenario mode, allows efficiently generating a number of similar scenes whose variance in object location and other properties is defined by a set of parameters. This feature may be used to generate the data required to train the system parameters from a small set of representative sequences (along with real videos). A large body of data also allows us to better evaluate the system on combined sets of real and synthetic annotated video sequences.

The synthetic videos are created by fully specifying all the properties of each object (including height, width, coordinates of bounding box, appearance, movement, and the remaining categorical properties) at each frame in the animations. The Video Scene Animator tool provides an interface for making this process more efficient and for adding certain randomization to this process. For example, one may specify a random (uniformly distributed) value within a certain range for the duration of time that an object takes to travel from one point to another. This is somewhat different, however, from the approach taken to generate synthetic data taken in (Oliver, Rosario, and Pentland, 2000a) which creates object agents
and allows them to “choose” their own behavior by reacting to other objects in the scene.

Despite the presented advantages, improper synthetic sequence usage can be misleading and produce unreasonable conclusions. It is user’s responsibility to ensure that synthetic scenes are modeled after real scenes before proceeding with the model training and event analysis.

The developed Video Scene Animator is a stand alone application. A visualization of an animated scene created using Video Scene Animator is seen in Figures 7.3 and 7.6.

7.2 Event Modeling

The event modeling component allows the knowledge engineer to construct the PN event model using a graphical interface. All place and transition nodes are specified according to knowledge of the domain along with enabling rules corresponding to each transition. Stochastic timed transition parameters may be specified or left to be learned from the data using the training mode of the Video Event recognition component.

A visualization of a model created using the event modeling component of SERF-PN is seen in Figures 6.1 and 6.2. The place and transition nodes and their connections can be seen in the figure. Associated enabling rule conditions are listed in the corresponding tables 6.1, 6.2, respectively. These condition specify the properties of the input tokens necessary for that transition to become enabled. This information, along with the model structure defines the full event model.

7.3 Video Event Analysis

The Video event recognition component of the SERF-PN system is used to aid a human analyst in monitoring video sequences online or used to annotate video sequences offline for content based retrieval.

The input to this system is the PN event model, along with the CAVIAR format annotated video sequence. All objects appearing in the video annotation are entered as tokens into the PN event model in a specially marked “root” place node. From there, transitions which become enabled according to their enabling rules are allowed to fire. As the video sequence processing progresses the properties of tokens corresponding to objects are updated to reflect the properties of the object. Transitions which represent events of interest (indicated as such in event model) are output to the video analyst (or stored as annotation for the video sequence in an offline application). This annotation includes information
on the event occurred (transition), the objects involved (tokens) and the frame number in the sequence.

If available, marking analysis data can be used to predict what is the next likely state of the system or what is likelihood of an abnormal or suspicious event given the current configuration. Currently the most probable next system state is output along with the event annotation.

The results of the interpretation are presented in the log window of the graphical user interface and may then be stored in a text file. To learn the parameters of the models the video event recognition module may run in training mode. In training mode this module disables timed transitions and learns their parameters by observing their average enabling time. It also constructs the dynamic reachability graph and collects statistics on transitions between markings. (see Section 6.2)

7.4 Evaluating SERF-PN

In this section we will evaluate the merits of our approach in a real world video event domain. This is achieved by studying how the interpretation results output by the system based on a particular PN event model compare to a human observer generated interpretation. Furthermore, we will provide a comparison with other approaches used for recognition in the domain of video event understanding.

For our first two experiments we captured a number of real video clips containing our events of interest. We then applied a tracker, recently developed in our lab (Leichter, Lindenbaum, and Rivlin., 2007), to obtain the tracking information. This tracker makes use of constancy of color and color edge features. As expected the tracking results were good but not without error. Object tracks were occasionally lost due to occlusions, lighting variation and other reasons.

In parallel we also generated a set of animated clips corresponding to the video events captured on camera. This allowed us to increase the volume of data available to test our system and to introduce unusual events that are improbable in real video (but which are usually the object of interest). In generating these animated clips some randomness was added in the object sizes and locations, mimicking the variability observed in the real videos.

As stated previously, the ground truth video data was obtained by allowing a human observer to determine whether a specific event is occurring in a video sequence. The system output for a particular clip is then checked for this particular event.

The third experiment is intended to compare our approach to other representation and recognition approaches. In this experiment we used available ground truth information in place of the tracker to supply low-level object information
such as location and bounding box properties. We used high-level ground truth information to evaluate the classification results of our model.

### 7.4.1 Security Check

For the security check experiment we had a total of 19 real video clips and 100 animations, each containing events of interest. Clip lengths ranged between 11-48 seconds. The mean and median lengths over all clips used in this experiments were 23 seconds. In this experiment we have defined two events of interest among the possible events: "Security Check Too Long" and "Visitor not checked". Not all clips include one of these event of interest. Some clips included multiple events of interest. More specifically in the real video clips both the "Security Check Too Long" event and the "Visitor not checked" event occurred 5 times. They occurred within the same clip twice. In the animations the "Security Check Too Long" event occurred 43 times and the "Visitor not checked" event occurred 14 times. The events occurred together in the same synthetic video sequence 7 times. We built the model as described in detail in section 6.3 and calculated precision and recall statistics on our events of interest. In Table 7.2 we report the results.
<table>
<thead>
<tr>
<th>Frame #</th>
<th>Transition / Event</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>#4757</td>
<td>Person Appeared</td>
<td>0</td>
</tr>
<tr>
<td>#4757</td>
<td>Transition 'Guard Post Manned'</td>
<td>0</td>
</tr>
<tr>
<td>#4798</td>
<td>Person Appeared</td>
<td>1</td>
</tr>
<tr>
<td>#4798</td>
<td>Visitor Entered</td>
<td>1</td>
</tr>
<tr>
<td>#4798</td>
<td>Guard Met Visitor</td>
<td>0, 1</td>
</tr>
<tr>
<td>#4913</td>
<td>Person Appeared</td>
<td>2</td>
</tr>
<tr>
<td>#4913</td>
<td>Visitor Entered</td>
<td>2</td>
</tr>
<tr>
<td>#4973</td>
<td>Visitor not checked</td>
<td>2</td>
</tr>
<tr>
<td>#4973</td>
<td>Guard Post Manned</td>
<td>2</td>
</tr>
<tr>
<td>#4976</td>
<td>Guard Post Unmanned</td>
<td>2</td>
</tr>
<tr>
<td>#4998</td>
<td>Person Disappeared</td>
<td>2</td>
</tr>
<tr>
<td>#5298</td>
<td>Security Check Too Long</td>
<td>0, 1</td>
</tr>
</tbody>
</table>

**Table 7.1:** A semantic summary corresponding to the video clip in Figure 7.2 generated by the video event recognition component of SERF-PN. Events mentioned in summary correspond to transitions in the model shown in Figure 6.1.

**Figure 7.3:** Keyframes from a typical animation clip sequence used in the Security Check Experiment. Hollow boxes are scene objects. The solid blue gray box indicates the "Guard Post" zone.
<table>
<thead>
<tr>
<th></th>
<th>Security Check Too Long</th>
<th>Visitor Not Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>Real</td>
<td>.92</td>
<td>1.0</td>
</tr>
<tr>
<td>Animated</td>
<td>.98</td>
<td>.91</td>
</tr>
<tr>
<td>Combined</td>
<td>.97</td>
<td>.92</td>
</tr>
</tbody>
</table>

Table 7.2: The recall and precision for the Security Check Model on real video clips and corresponding animations. The events considered were "Security Check Too Long" and "Visitor not Checked".

This experiment was carried out on both the set of real videos with tracking info, the set of animations only and the combined data set. The length of the average activation was manually set using knowledge of the event domain ($\mu_0 \approx 20$ sec).

It is important to note that treating this model as an event occurred/hasn’t occurred classifier was done for evaluation purposes and that the output of the system is, in fact, a semantic description of the events in a particular video sequence. Such a semantic description, corresponding to the event shown in Figure 7.2, is shown in Table 7.1. Qualitative assessment of this description shows it to be quite similar to a possible human description of the same video sequence.

As the table indicates the results are quite good in both the animation set, which corresponds to perfect tracking and detection, and in the real video dataset with the automatic tracker applied which is less than perfect. Our experiments showed that minor tracking error has little affect the high-level analysis results. While an object completely lost by the tracker cannot be reasoned upon by the high-level system, the system can still provide an analysis up to the point of object loss. That is, we can still get meaningful semantic description from only partially correct tracks.

Prediction using the mechanism of marking analysis can also be illustrated using this example. We applied a dynamic marking analysis using our dataset of real and animated clips. The marking graph generated with meaningful names for each marking and associated transition probabilities is shown in Figure 7.4. This Figure reduces the full marking graph by consolidating markings with similar semantic meanings and neglecting markings which are reachable with very low probability. Note that in the full marking graph outgoing arrows must sum to one. Hence, we renormalize all remaining outgoing arcs from the same marking node to sum to one. A marking node with no outgoing arrows indicates that the system did not observe a transition from this marking to another during the training process.

Using this information to construct a Markov process as described in section 6.2.2 allows answering such queries as what is the next most likely state given the
current state and what is the probability of a particular state given the current state. For example, note that given that a check of a visitor has started there is only a small probability that the check will be too long and a moderate possibility of an unchecked visitor. However, once the check has gone over the time limit, the probability of an unchecked visitor entering the secure area increases.

It is also interesting to note that although the theoretical number of markings is exponential in the number of tokens, by using the dynamic marking analysis we can concentrate on the markings in which most of the probability mass is concentrated. In our example, if we assume three actors only, out of a possible 120 markings (obtained by plugging \( \Pi(3, 8) \) into equation 6.2 ) our marking analysis observed only 18 markings in our training data which we reduced to 6 with semantic meaning and significant probability mass.

### 7.4.2 Traffic Intersection

The Traffic Intersection scenario is slightly more complex semantically than the previously discussed security check. We built the model as described in detail in section 6.3. The parameter of the stochastic timed transition modelling a safe interaction between two vehicles with collision course trajectories may not be obvious even to an expert in the domain. To this end we apply our timed parameter training approach as discussed in section 6.2.1. We applied this method to several subsets of available data (animations and real) using 4-fold cross validation. That is, we divided the available labeled data (including real and synthetic data) into 4 roughly equal sets. We then trained the timing parameters using one of these sets, and applied the trained event model to the remaining three sets. We repeated this process for each of the sets (each time training on one and testing on the rest) and averaged the accuracy results over all four sub-experiments to
Figure 7.5: Keyframes from a typical real video sequence used in the Traffic Intersection Experiment. Ellipses denoting object locations generated by the tracker are shown.

<table>
<thead>
<tr>
<th>Frame #</th>
<th>Transition</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>#61350</td>
<td>Car Appeared in int</td>
<td>0</td>
</tr>
<tr>
<td>#61355</td>
<td>Car Approaching Intersection(East)</td>
<td>1</td>
</tr>
<tr>
<td>#61394</td>
<td>Car Approaching Intersection(South)</td>
<td>2</td>
</tr>
<tr>
<td>#61410</td>
<td>Safe Interaction Occured</td>
<td>0, 2</td>
</tr>
<tr>
<td>#61414</td>
<td>Car Enters Intersection</td>
<td>2</td>
</tr>
<tr>
<td>#61418</td>
<td>Safe Interaction Occured</td>
<td>1, 2</td>
</tr>
<tr>
<td>#61418</td>
<td>Car Approaching Intersection(North)</td>
<td>3</td>
</tr>
<tr>
<td>#61437</td>
<td>Car Leave Int After Safe Turn</td>
<td>2</td>
</tr>
<tr>
<td>#61463</td>
<td>Car Enters Intersection</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 7.3: A semantic summary corresponding to the video clip in Figure 7.5 generated by the video event recognition component of SERF-PN. Events mentioned in summary correspond to transitions in the model shown in Figure 6.2
Figure 7.6: Keyframes from a typical animation clip sequence used in the Traffic Intersection Experiment. Filled circles model cars in the scene. The solid black polygon marks the "Intersection" zone.

Once this parameter is obtained we again take the approach of viewing our summary output as a class/nonclass classifier for a particular event, in this case the event "Safe Interaction Occurred". The results are reported in Table 7.4. As can be observed from the table, the precision and recall rates are slightly lower than those of the previous example. This can be attributed to the increased semantic complexity. In inspection of the results we found that those clips marked by humans as safe are also marked as safe by the model. However, the model also has a tendency to mark human "not safe" events as safe. Of course this can be remedied by reducing the semantic definition into relations that fall into the vocabulary of the Petri Net formalism and adding them to the model (i.e. change
Table 7.4: The recall and precision for the Traffic Intersection Model applied real video clips and corresponding animations for the event "Safe Interaction Occurred"

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>.91</td>
<td>.75</td>
</tr>
<tr>
<td>Animated</td>
<td>.80</td>
<td>.97</td>
</tr>
<tr>
<td>Combined</td>
<td>.85</td>
<td>.85</td>
</tr>
</tbody>
</table>

For this experiment we made use of 48 real video clips and 77 animations. Each clip had at least one "safe" or "not safe" event. Of the real clips we captured, 44 include a "safe" event and the remaining 4 included "unsafe" events. Of the synthetic video clips 38 clips included "safe" events and the remaining 39 include "unsafe" events. The maximum and minimum clip lengths used in this experiment are 11 and 3 seconds, respectively. The mean length length over all clips is about 6 seconds. The median length is 7 seconds. The animations were particularly useful in simulating "not safe" scenarios which were difficult to come across in the real video sequences.

At this point, we would like to note that we did not take an approach (similar to the one proposed in (Oliver, Rosario, and Pentland, 2000a)) of training on the synthetic data and testing on the real data. The reason for this is that the training data is created by specifying semantically meaningful parameters including event timing. Were we to train the timing parameters of the model using only this synthetic data (as we have in the "animated only" part of the experiment), we would simply recover the original parameters we put into the algorithm for creating the synthetic data. This is equivalent to manually setting the timing parameters using domain knowledge. This is unlike the approach described in (Oliver, Rosario, and Pentland, 2000a) which attempts to learn abstract (i.e. not semantically meaningful) parameters from the synthetic data. Since these parameters are abstract they cannot be easily specified using semantic knowledge and, hence, must be learned from synthesized example(s) of the event.

7.4.3 CAVIAR dataset

In our third experiment, we evaluated our approach to modeling events on a portion of the CAVIAR dataset (Fisher, 2004b), that includes surveillance video of a shopping center in Lisbon, for the purpose of comparing our approach to others tested on the same dataset. The CAVIAR(Context Aware Vision using Image-based Active Recognition) dataset covers the potential events that can
occur in a shopping mall surveillance context. It defines a strict semantic hierarchy of events. In the CAVIAR ground truth (Fisher, 2004c), each object is assigned a movement, role, situation, and context. Each of these variables can take on a value from a discrete set of values. Also each of these categories represents a more specific/general semantic level of information. Movement, is the lowest level of information and can take on the states running, walking, active, inactive or unknown. Role, describes the role an object plays in a higher level event (Situation or Context). Role can take on such states as walker, fighter, and left object. Situation is a higher semantic level and can take on values such as moving, browsing, shop enter, shop exit and so on. Context can be described as the overriding semantic purpose of the object in question. In the CAVIAR ground truth each object is assigned only one context throughout the duration of its appearance in the scene. Context values may be shop enter, shop exit, shop re-enter, walking, windowshopping, etc. CAVIAR also defines each particular context using a number of situations. For instance the windowshopping context may consist of the situations moving, browsing, and then moving again. That is, we can define situations using the movement information, and contexts according to the situation information.

Contrary to our previous experiments, where we attempted to classify whether an event has occurred or not (i.e. a particular transition has fired), in this experiment we attempt to classify the state of each object at every frame (i.e. the place node the object token is residing in). This is an equivalent problem as it is the events (i.e. transitions) that cause the changes in object states (place nodes containing tokens). We have chosen to present our results in this fashion to allow a more straightforward comparison with the available ground truth.

Thus to evaluate our PN representation approach we built two PN Models, one to evaluate the situations based on object location and movement information. In principle, this information can be given by a tracker, as in our previous experiments, but for comparison purposes we have chosen to use the ground truth information provided as done in other approaches. The second PN model evaluates the context of each object using the object information along with the situation label obtained from the ground truth.

The situation network built for this experiment is shown in Figure 7.7. In this network the transition conditions are based on all object properties available in the ground truth except for the situation and context labels. In the majority of cases, we have found it sufficient to use the appearance, movement and location properties. Additionally, we have defined some important zones in the scene including "shop exit", "shop enter", and "windowshop" zones. These zone definitions allow us to condition transitions on whether an object location is inside/outside a particular zone. The place node names of our network correspond to possible situation values. We made use of this in the experiment to compare
the ground truth label of each object at each frame with the label of the place in which the corresponding token is residing. A detailed description of the conditions on each transition is given in Table 7.5. In the table $\text{Token.$attributex$}$ denotes the value of $\text{attributex}$ in the particular input token. In this example we use the attributes "loc" (token location), "Movement" (which can take on the values {"walking","running","active","inactive"}) and "Appearance", (which can take on the values {"appear","visible","disappear"}). Note that the same transition may appear multiple times, sometimes with a temporal duration constraint attached to the condition.

The context network is an event model, with the events being the changes in the context classification of the particular object. Like in the situation network we condition the transitions in this net on the properties of objects including location, appearance, and movement. Because this network models a higher semantic level than the situation network we also allow the use of the situation property in these transition conditions. Again, we have made use of the definitions of important zones in the scene including the "Shop Enter" "Shop Exit" and "windowshop" zones. The place node names corresponds to the various values the context property may take on according to the CAVIAR specification, to allow comparison with the ground truth on a frame by frame basis.

A detailed description of the conditions on each transition is given in Table 7.6. In this table we use the attributes "loc" (location), "Movement" (which can take on the values {"walking","running","active","inactive"}) , "Appearance" (which can take on the values {"appear","visible","disappear"}) and situation (which can take on the values {"moving","inactive","browsing","shop enter","shop exit"}).

The situation of an object may change throughout the scene, and so a frame by frame evaluation of the resulting situation against the situation label given

<table>
<thead>
<tr>
<th>Transition Name</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Situation: Shop Exit</td>
<td>$\text{Token.loc overlaps &quot;Shop Exit&quot; zone}$</td>
</tr>
<tr>
<td>Situation: Moving</td>
<td>$\text{Token.Appearance='visible' and Token.Movement='walking'}$</td>
</tr>
<tr>
<td>Situation: Browsing</td>
<td>$\text{Token.Appearance='visible' and Token.Movement='active' and Token.loc overlaps &quot;windowshop&quot; zone}$</td>
</tr>
<tr>
<td>Situation: Inactive</td>
<td>$\text{Token.Appearance='visible' and Token.Movement='inactive'}$</td>
</tr>
<tr>
<td>Situation: Shop Enter</td>
<td>$\text{Token.Appearance='visible' and Token.Movement='active' and Token.loc overlaps &quot;Shop Enter&quot; zone}$</td>
</tr>
<tr>
<td>Stops in Place</td>
<td>$\text{Token.Movement='active' or Token.Movement='inactive'}$</td>
</tr>
<tr>
<td>Disappear</td>
<td>$\text{Token.Appearance='disappear'}$</td>
</tr>
</tbody>
</table>

Table 7.5: Each transition in the situation model structure pictured in figure 7.7 is associated with a condition on the properties of its input tokens.
Figure 7.7: CAVIAR Situation Petri Net Model. See Table 7.5 for specification of transition condition.

<table>
<thead>
<tr>
<th>Transition Name</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context: Shop Exit</td>
<td>Token.situation=&quot;Shop Exit&quot;</td>
</tr>
<tr>
<td>Context: Window Shopping</td>
<td>Token.situation='browsing' and Token.loc overlaps &quot;windowshop&quot; zone</td>
</tr>
<tr>
<td>Context: Walking</td>
<td>Token.situation='moving'</td>
</tr>
<tr>
<td>Context: Immobile</td>
<td>Token.situation='inactive'</td>
</tr>
<tr>
<td>Context: Shop Enter</td>
<td>Token.situation='shop enter'</td>
</tr>
<tr>
<td>Context: Shop Re-Enter</td>
<td>Token.situation='shop enter'</td>
</tr>
</tbody>
</table>

Table 7.6: Each transition in the context model structure pictured in figure 7.8 is associated with a condition on the properties of its input tokens.
in the ground truth is warranted. To this end we compared the situation label against the label of the place node in which the token corresponding to the object resided. Of course, each of the place node labels in this network corresponds to a possible situation value.

Context may also be measured in this frame by frame manner. However, when considering some types of contexts it is not reasonable to expect their detection from the first appearance of the object in the scene. For example, a context such as enter store can only be detected (by a human or an automatic system both) when the object enters the store. Thus, when compared to the oracle-like nature of the ground-truth context on a frame by frame basis even the best system would
produce low classification performance on these types of context.

For this reason it is beneficial to use the frame by frame situation classification in a more sophisticated way to determine the context for the entire duration of the object’s existence within the scene. In our experiments we report results on both the frame by frame classification of the context and a simple but effective method of looking at the entire sequence to determine the overall context, using the last frame context classification.

For comparison we offer two approaches with results described in (Fisher, 2004b). The first is that of a rule based approach. This approach used semantic rules on both the role and movement classifications in the ground truth to arrive at situation and context classifications. This approach is somewhat similar to the use of semantic knowledge in our approach, but is missing the basic formalism (i.e. Petri - Net ) to allow us to model these rules efficiently and determine if they apply in a straightforward manner. This rule base also lacks the notion of states intrinsic to Petri net markings. All this means that rules applied are less powerful and robust and less likely to capture the semantics of the event domain appropriately.

Another approach is that of a probabilistic model with parameters that can be tuned to the event domain and offers a probabilistic interpretation of the uncertainty in the event classification. Another strength of the probabilistic approach is its ability to accept soft probabilistic evidence as opposed to hard evidence expected by semantic rule based approaches including our approach. The particular probabilistic graphical model explored for the CAVIAR context interpretation problem is the Hidden Semi-Markov Model (HSMM) (Tweed et al., 2005; Honggeng and Nevatia, 2003b). HSMMs extend the standard Hidden Markov model with an explicit duration model for each state. Using the situation values as the possible values of hidden states, and training a separate HSMM for each context model, the model is able to answer questions like what is the most likely state sequence given a particular observation, and also which is the most likely context model. Apart from the results offered in (Fisher, 2004a) the HSMM approach for event detection is described in (Tweed et al., 2005). Essentially the probabilities are used to determine which is the most likely context given a sequence of situations.

We ran our experiments on a total of 26 CAVIAR sequences ranging in length from 10 to 123 seconds. The mean length of these sequences was about 47 seconds and the median length about 49 seconds. Overall we considered 235 objects, 5 possible situations, and 7 possible contexts. There were several sequences where no objects were visible for a significant portion of the sequence, as well as several sequences where a number of objects appeared simultaneously. We built two PN event models, one to classify situations and one to classify contexts.
For our first experiment we evaluated the PN model’s ability to classify situations per frame. Each object’s situation was set to be equal to the label of the place node in the PN model in which the token corresponding to this particular object is residing. The accuracy of this classification is obtained by comparing this label to the situation label in the ground truth.

The confusion matrix for this experiment is shown in Table 7.7. Overall the frame by frame situation classification accuracy achieved is 91%. Another observation that can be made by examining the table is that some situations are better detected than others. Specifically, ”shop enter” is often confused with other situations such as ”walking”. However, this error is subjective to interpretation and as we shall see does not greatly effect the eventual higher level context results that use these situation classifications.

The context evaluations are shown both using a frame by frame approach similar to that of the situation evaluations, and using a simple evaluation over the series of situations, namely selecting the last frame in the object life-span and using the context evaluation in that frame as the context of the object. In this evaluation, we thus evaluate the context of each object against its ground truth, as opposed to the frame by frame approach which evaluates each frame of each object. For this reason the total contexts evaluated in this ”last frame” approach are significantly less than those shown in the frame by frame approach.

It can be seen however that this ”last frame” approach has a large improvement of context classification results over those of the frame by frame approach. This is attributed to the fact that many contexts (e.g. ”shop enter”, ”shop reenter”) only become apparent towards the ends of the object’s life in the scene.

Table 7.7: Situation frame by frame classification

<table>
<thead>
<tr>
<th></th>
<th>moving</th>
<th>inactive</th>
<th>browsing</th>
<th>shop enter</th>
<th>shop exit</th>
<th>unknown</th>
<th>Total</th>
<th>Overall Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>moving</td>
<td>103883</td>
<td>347</td>
<td>1349</td>
<td>72</td>
<td>3085</td>
<td>301</td>
<td>109017</td>
<td>0.9529</td>
</tr>
<tr>
<td>inactive</td>
<td>77</td>
<td>5695</td>
<td>1397</td>
<td>0</td>
<td>214</td>
<td>226</td>
<td>7609</td>
<td>0.7485</td>
</tr>
<tr>
<td>browsing</td>
<td>182</td>
<td>1118</td>
<td>9173</td>
<td>0</td>
<td>211</td>
<td>78</td>
<td>10762</td>
<td>0.8524</td>
</tr>
<tr>
<td>shop enter</td>
<td>1224</td>
<td>0</td>
<td>438</td>
<td>45</td>
<td>62</td>
<td>1798</td>
<td>7609</td>
<td>0.2476</td>
</tr>
<tr>
<td>shop exit</td>
<td>2123</td>
<td>0</td>
<td>0</td>
<td>1390</td>
<td>3</td>
<td>3516</td>
<td>132673</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Tables 7.8 and 7.9, show the results for context evaluation, for frame by frame and ”last frame” approaches respectively. These results are based on the application of the context PN model to the ground truth situation information. For the frame by frame approach we achieve an overall accuracy of 69%. For the ”last frame” classification we achieve a significantly better 86%.

The next experiment is designed to evaluate the effect of the errors in the situation evaluation on the final context classification. To this end we repeated the experiment, this time replacing the ground truth situations with those yielded by the situation network. As can be seen in Tables 7.10 (frame by frame) and 7.11...
Table 7.8: Context frame by frame classification (using situation ground truth)

<table>
<thead>
<tr>
<th></th>
<th>browsing</th>
<th>immobile</th>
<th>walking</th>
<th>windowshop</th>
<th>shop enter</th>
<th>shop exit</th>
<th>shop reenter</th>
<th>unknown</th>
<th>Total</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>browsing</td>
<td>0</td>
<td>0</td>
<td>2323</td>
<td>4054</td>
<td>0</td>
<td>433</td>
<td>0</td>
<td>0</td>
<td>7505</td>
<td>0.69</td>
</tr>
<tr>
<td>immobile</td>
<td>0</td>
<td>3290</td>
<td>8865</td>
<td>224</td>
<td>0</td>
<td>232</td>
<td>0</td>
<td>0</td>
<td>12611</td>
<td>0.2609</td>
</tr>
<tr>
<td>walking</td>
<td>0</td>
<td>0</td>
<td>45993</td>
<td>1103</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>0</td>
<td>47154</td>
<td>0.9754</td>
</tr>
<tr>
<td>windowshop</td>
<td>0</td>
<td>0</td>
<td>4804</td>
<td>15078</td>
<td>79</td>
<td>150</td>
<td>0</td>
<td>36</td>
<td>20147</td>
<td>0.7484</td>
</tr>
<tr>
<td>shop enter</td>
<td>0</td>
<td>0</td>
<td>17402</td>
<td>0</td>
<td>1533</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>18960</td>
<td>0.9869</td>
</tr>
<tr>
<td>shop exit</td>
<td>0</td>
<td>0</td>
<td>371</td>
<td>0</td>
<td>0</td>
<td>25166</td>
<td>0</td>
<td>0</td>
<td>25537</td>
<td>0.9855</td>
</tr>
<tr>
<td>shop reenter</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>663</td>
<td>96</td>
<td>0</td>
<td>759</td>
<td>759</td>
<td>0.1265</td>
</tr>
<tr>
<td>Total</td>
<td>132673</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Acc:</td>
<td>0.69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.9: Context "last frame" classification (using situation ground truth)

<table>
<thead>
<tr>
<th></th>
<th>browsing</th>
<th>immobile</th>
<th>walking</th>
<th>windowshop</th>
<th>shop enter</th>
<th>shop exit</th>
<th>shop reenter</th>
<th>unknown</th>
<th>Total</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>browsing</td>
<td>0</td>
<td>59</td>
<td>2642</td>
<td>3804</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7505</td>
<td>0.69</td>
</tr>
<tr>
<td>immobile</td>
<td>0</td>
<td>3073</td>
<td>6872</td>
<td>1103</td>
<td>872</td>
<td>3073</td>
<td>0</td>
<td>73</td>
<td>47154</td>
<td>0.8914</td>
</tr>
<tr>
<td>walking</td>
<td>0</td>
<td>0</td>
<td>42033</td>
<td>1103</td>
<td>872</td>
<td>3073</td>
<td>0</td>
<td>73</td>
<td>47154</td>
<td>0.8914</td>
</tr>
<tr>
<td>windowshop</td>
<td>0</td>
<td>0</td>
<td>6173</td>
<td>13746</td>
<td>26</td>
<td>178</td>
<td>8</td>
<td>16</td>
<td>20147</td>
<td>0.6823</td>
</tr>
<tr>
<td>shop enter</td>
<td>0</td>
<td>0</td>
<td>17372</td>
<td>11</td>
<td>694</td>
<td>784</td>
<td>52</td>
<td>48</td>
<td>18960</td>
<td>0.9997</td>
</tr>
<tr>
<td>shop exit</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>25529</td>
<td>0</td>
<td>1</td>
<td>25537</td>
<td>0.9997</td>
</tr>
<tr>
<td>shop reenter</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>752</td>
<td>7</td>
<td>0</td>
<td>759</td>
<td>0.0092</td>
</tr>
<tr>
<td>Total</td>
<td>132673</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Acc:</td>
<td>0.64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.10: Context frame by frame classification (using PN derived situation information)

("last frame"), the classification of the context is downgraded a smaller amount then the actual error in classification (10%).

For comparison purposes we evaluate the "rule-based" and HSMM algorithms for evaluating both situation and context. Table 7.12 shows the results reported in (Fisher, 2004a) on the CAVIAR dataset as compared to our results on the same set. As is shown in the table our frame by frame situation evaluation is a significant improvement over both the "rule-based" and the HSMM situation results. Although a PN model can be considered a type of rule- based model based on semantic knowledge, the framework it provides allows us to keep track of object states and model relationships, thus allowing for a more robust rule set and hence the improvement in results. With additional semantic information specifically orientation and speed information these results possibly may be improved.

In the frame by frame context evaluation we show results comparable to those
Table 7.11: Context "last frame" classification (using PN derived situation information)

Table 7.12: Comparison

of the HSMM context classifier. We should note however that the HSMM classifier does not take a naive approach to classifying each frame and in fact performs inference on all the situation values at each frame to determine what is the most likely context. If we apply a similar approach, although admittedly much simpler, of simply taking the last frame of each object and using the context label as the context label for the entire sequence, our "last frame" classification strategy, we achieve significantly better classification using the PN model approach. This due to the fact that the classification are made based on a series of rules reflecting semantic knowledge of the domain as opposed to a set of abstract state transition parameters (as in the HSMM) with no inherent semantic value.
Chapter 8

Coping with Uncertainty In a Petri Net Event Model

The previous two chapters have shown that the expressiveness of Petri Nets can successfully be used to model the semantics of a video event domain, and afford recognition of events in this domain. One question that still remains is how to elegantly deal with uncertainty inherent in the video data. Probabilistic State Models (see Section 3.4) have well-studied methods of dealing with uncertainty, but are not able to robustly model the semantics of the video event domain.

This chapter proposes an approach that models events with the Petri Net formalism. This semantic model is then mapped into a probabilistic model. This construction allows the application of well-studied methods for dealing with uncertainty, while still taking advantage of the Petri Net Structure for modeling events.

The terminology in this chapter deviates from the terminology used in previous chapters. The reason for this is that, unfortunately, our proposed terminology for Chapter 2 has not been widely adopted, and we wish to use terminology consistent with the works which we are comparing against. However, using the proposed terminology in Chapter 2 we can more exactly define our terms. Thus, in the parlance of the proposed terminology, we define the term activity to mean multi-thread complex event, and the term event is redefined to mean an atomic sub-event of the activity restricted to a temporal interval.

The probabilistic model we construct will be cast into the Bayesian Recursive Filter (BRF) framework. Within the proposed framework, each event is associated with a certainty. These certainties are treated as features in the measurement model. This construction allows application of the Particle Filter algorithm, enabling reasoning on uncertain event observations independent of how observation certainty measures are computed. Furthermore, the recursive framework maintains a system state estimation at each frame of the video, allowing dynamic
alerts when activities are recognized.

Our approach to activity recognition takes two inputs: (1) an activity definition, specified as a Petri Net, and (2) a video sequence specified as a list of time-stamped events with associated certainties.

Given these inputs we first compute the Bayesian Recursive Filter components from the activity definition. Our recognition algorithm, based on the Particle Filter, then processes each frame in the video sequence in chronological order, updating the certainty in the recognition of the activity after each new observation, and taking into account the certainty of all observed events.

8.1 Modeling activities with Petri Nets

The approach to modeling video activities with Petri Nets we adopt in this framework differs from the approach in previous chapters. This approach involves connecting fragments corresponding to the composing events in such a way that enforces the temporal and logical relationships between them. Using the Petri Net formalism, we are able to model all temporal interval relations as defined in (Allen and Ferguson, 1994). Place nodes are “waypoints” which indicate the progress throughout the activity. Special place nodes indicate the start and end of the activity. The “start” place node will hold a token when the recognition of the activity is still in its initial state. The “end” place node will hold a token when the activity recognition has been completed. Otherwise, one or more of the intermediate place nodes will hold a token (e.g. the token configuration pictured in Fig. 8.1 indicates the activity state: the teller is present, but the visitor has yet to appear). Each transition node in the figure has an associated event label. For a transition to fire, this event must be observed while the transition is enabled.

Consider the Petri Net definition of an activity depicted in Figure 8.1. This is part of the Bank Attack activity used in our experiments which is closely modeled after an activity Petri Net proposed in (Albanese et al., 2008a). The Petri Net activity model in the figure may be formalized as a tuple $< P, T, C, events, \delta, S, F >$.
where \( P = \{P_1, P_2, ..., P_8\} \) is the set of places, \( T = \{T_1, T_2, ..., T_9\} \) is the set of transitions, \( C \) is the set of connecting arcs, \( \text{events} = \{\text{Visitor Appears, Teller Appears, Teller in Safe,} \ldots\} \) is the set of events that are relevant to the activity, and \( \delta : T \rightarrow \text{events} \) is the labeling function mapping transitions to an event label implied by the figure. For example, \( \delta(T_3) = \text{Visitor in Safe} \). \( S = \{P_1\} \) is the place node representing the “start” of the activity, and \( F = \{P_{10}\} \) is the set of place nodes representing the recognition of the activity.

### 8.2 Defining a Bayesian Recursive Filter Using a Petri Net

In order to apply particle filter techniques to our approach, we have to translate our formulation of an activity, specified as a Petri Net, into the language of the Bayesian Recursive Filter. More specifically, we have to define the space of possible states our system can take on at each time step, and the space of possible observations at each time step. Furthermore, we must define the dynamic model, \( P(x_t|x_{t-1}) \), which describes the evolution of states in time, as well as the measurement model, \( P(y_t|x_t) \), which defines the likelihood of a particular observation given the system is in a particular state.

#### 8.2.1 Preliminaries

Given a particular activity Petri Net, coupled with an initial marking, we can derive the set of all reachable markings. We denote this set using the letter \( S \). We also use the letter \( R \) to indicate the set of all markings that have a token in the recognized place of the Petri Net. Clearly, \( R \subset S \).

Additionally the Petri Net defines the set of all relevant events that may be observed, denoted \( O \). These events are those used to label the transitions in the Petri Net. Note that since multiple transitions may have the same event label, the number of events is not necessarily equal to the number of transitions.

Since multiple events may be observed at the same time, we define another set, \( M \), as a subset of the powerset of \( O \). This set serves as the space over all relevant event combinations that may occur during the video analysis. Note that \( M \) trivially contains \( \emptyset \), the empty set. Table 8.2.1 summarizes our notation for convenience.

#### 8.2.2 The State Space

The Bayesian paradigm requires a dynamic model in which the current state is fully determined by the previous state(s). Therefore, in order to model Petri Net
the set of all reachable markings
the set of all markings where the activity is recognized
the set of all relevant events
the set of all relevant event combinations
the state of the system at time $t$
the underlying Petri Net marking at time $t$
the event combination occurring at time $t$
the observation at time $t$
the confidence in the observation of the $i$-th event in $O$ at time $t$

<table>
<thead>
<tr>
<th>$S$</th>
<th>the set of all reachable markings</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>the set of all markings where the activity is recognized</td>
</tr>
<tr>
<td>$O$</td>
<td>the set of all relevant events</td>
</tr>
<tr>
<td>$M$</td>
<td>the set of all relevant event combinations</td>
</tr>
<tr>
<td>$x_t$</td>
<td>the state of the system at time $t$</td>
</tr>
<tr>
<td>$x_t(m)$</td>
<td>the underlying Petri Net marking at time $t$</td>
</tr>
<tr>
<td>$x_t(e)$</td>
<td>the event combination occurring at time $t$</td>
</tr>
<tr>
<td>$y_t$</td>
<td>the observation at time $t$</td>
</tr>
<tr>
<td>$y_{t(i)}$</td>
<td>the confidence in the observation of the $i$-th event in $O$ at time $t$</td>
</tr>
</tbody>
</table>

Table 8.1: Notation Summary

dynamics in this paradigm, we must consider the (possibly empty) set of events that occur at each frame as a component of the state. Thus, $x_t = < x_t(m), x_t(e) >$, the state variable at frame $t$ of the video sequence, is a two component tuple. The first component, $x_t(m) \in S$, denotes the marking of the Petri Net at time $t$. The second component, $x_t(e) \in M$, denotes the event(s) (if any) that occur at time $t$.

### 8.2.3 The Observation Space

At each frame we will observe a certainty value for each relevant event in the set $O$. Thus the observation is a vector, $y_t \in [0,1]^{|O|}$. The $i$th component of $y_t$, denoted $y_{t(i)}$, represents the certainty in the observation of $O(i)$, the $i$th event in $O$.

### 8.2.4 Conditional Independence

In addition to defining our state and observation variables, we also assume some conditional independence among the variables to reflect the nature of the underlying Petri Net. First, our state space is factorized using the (first-order) Markov Assumption, which assumes that the state at any frame $t$ is independent of the state at all previous frames, given the state at frame $t - 1$. Thus, it is only necessary to maintain an approximation of the dependence between the states at time slices $t$ and $t - 1$. Additionally, we assume that the events that occur in a particular frame can only be those corresponding to enabled transitions in the underlying Petri Net. Furthermore, given the events that occur at time $t$, the observation variable is independent of the current marking. More concretely, $x_t(e)$ is independent of $x_{t-1(m)}$ and $x_{t-1(e)}$ given $x_t(m)$. Also $y_t$ is independent of $x_t(m)$ given $x_t(e)$. These conditional independence relations are illustrated by the simple graphical model shown in Figure 8.2.
8.2.5 The Dynamic model

The dynamic model, $P(x_t|x_{t-1})$, is decomposed as follows:

$$P(x_t|x_{t-1}) = P(x_{t(m)}, x_{t(e)}|x_{t-1(m)}, x_{t-1(e)}) = P(x_{t(m)}|x_{t-1(m)}, x_{t-1(e)})P(x_{t(e)}|x_{t(m)})$$  \hspace{1cm} (8.1)

where the second equality is due to the conditional independence of $x_{t(e)}$ from $x_{t-1(m)}$ and $x_{t-1(e)}$ given $x_{t(m)}$ (see Figure 8.2). Thus, it suffices to derive $P(x_{t(e)}|x_{t(m)})$ and $P(x_{t(m)}|x_{t-1(m)}, x_{t-1(e)})$ from the Petri Net structure. We derive an uninformative $P(x_{t(e)}|x_{t(m)})$, giving equal probability to all enabled transitions from the structure of the Petri Net as follows:

$$\hat{P}(x_{t(e)}|x_{t(m)}) = \begin{cases} 1 & \text{if all transitions in } x_{t(e)} \text{ are enabled in marking } x_{t(m)} \\ 0 & \text{otherwise} \end{cases}$$ \hspace{1cm} (8.2)

where $x_{t(e)} = \emptyset$ (the empty set) is considered to be enabled in all markings. We proceed to normalize the above:

$$P(x_{t(e)}|x_{t(m)}) = \frac{\hat{P}(x_{t(e)}|x_{t(m)})}{\sum_{x' \in M} P(x'|x_{t(m)})}$$ \hspace{1cm} (8.3)
Similarly, \( P(x_{t(m)}|x_{t-1(m)}, x_{t-1(e)}) \) is derived from the Petri Net as follows:

\[
\hat{P}(x_{t(m)}|x_{t-1(m)}, x_{t-1(e)}) = \begin{cases} 
1 & \text{if } x_{t(m)} \leftarrow x_{t-1(m)}|x_{t-1(e)} \\
0 & \text{otherwise} 
\end{cases} 
\]  

(8.4)

where \( x_{t(m)} \leftarrow x_{t-1(m)}|x_{t-1(e)} \) indicates there is a path from state \( x_{t-1(m)} \) to state \( x_{t(m)} \) via transitions labeled with the event(s) \( x_{t-1(e)} \). We then proceed to normalize the above:

\[
P(x_{t(m)}|x_{t-1(m)}, x_{t-1(e)}) = \frac{\hat{P}(x_{t(m)}|x_{t-1(m)}, x_{t-1(e)})}{\sum_{x' \in S} \hat{P}(x'|x_{t-1(m)}, x_{t-1(e)})} 
\]  

(8.5)

### 8.2.6 The Measurement Model

In order to derive the measurement model, \( P(y_t|x_t) \), we first assume that all components of the observation vector are independent of one another. This assumption yields the following equation:

\[
P(y_t|x_t) = \prod_{i=1}^{\|O\|} P(y_{t(i)}|x_t) 
\]  

(8.6)

where \( y_{t(i)} \) denotes the \( i \)th component of the observation vector. Furthermore, using the assumption that \( y_t \) is conditionally independent of \( x_{t(m)} \) given \( x_{t(e)} \) (see Figure 8.2) we get:

\[
P(y_t|x_t) = \prod_{i=1}^{\|O\|} P(y_{t(i)}|x_t) = \prod_{i=1}^{\|O\|} P(y_{t(i)}|x_{t(m)}, x_{t(e)}) = \prod_{i=1}^{\|O\|} P(y_{t(i)}|x_{t(e)}) 
\]  

(8.7)

It therefore suffices to derive \( P(y_{t(i)}|x_{t(e)}) \) from the Petri Net structure to obtain the measurement model.

The construction of the observation likelihood, \( P(y_{t(i)}|x_{t(e)}) \), models the notion that if a particular event occurs, the observed certainty in this event should be high. The sigmoid function (centered on 0.5) is a natural fit for modeling this intuition. Conversely if an event does not occur, the observed certainty in this event should be low. The complement to the sigmoid function models this alternate situation. Recall, the \( i \)th component of \( y_t \), denoted \( y_{t(i)} \), represents certainty in the \( i \)th event in set \( O \), denoted \( O(i) \). Since, \( x_{t(e)} \in M \subseteq 2^O \)(the powerset of \( O \)) we can check the membership of \( O(i) \) in the set \( x_{t(e)} \).

\[
P(y_{t(i)}|x_{t(e)}) = \begin{cases} 
\varphi(y_{t(i)}) & \text{if } O(i) \in x_{t(e)} \\
1 - \varphi(y_{t(i)}) & \text{otherwise} 
\end{cases} 
\]  

(8.8)

where \( \varphi(z) \) is the sigmoid function centered on 0.5:
\[ \varphi(z) = \frac{1}{1 + e^{-k(z-0.5)}} \]  

(8.9)

### 8.2.7 The Prior Probability

The prior probability, \( P(x_0) \), is dictated by the initial marking of the Petri Net

\[
P(x_0) = P(x_{0(m)}, x_{0(e)}) = \begin{cases} 
1 & \text{if } x_{0(m)} = \text{initial marking and } x_{0(e)} = \emptyset \\
0 & \text{otherwise} 
\end{cases} 
\]  

(8.10)

### 8.3 Propagating Certainty

Recall that our objective in this work is to translate a list of events with associated certainty values into an activity label (with its own associated certainty value). In this section we describe the mechanics of how this “propagation” of certainty, from the event level to the activity level, is achieved. Our approach to the propagation of certainty is based on the particle filter. Informally, at each discrete time slice we maintain a set of hypotheses, called particles, on the state (progress) of the activity. The sum over the weights of all particles in a particular state is the certainty of this state. Together these particles will approximate the posterior distribution, \( P(x_t | y_{1:t}) \). More formally, we denote the set of particles, \( X_t = \{ x_t^{(1)}, x_t^{(2)}, x_t^{(3)}, \ldots, x_t^{(N)} \} \), where \( N \) is the number of particles, \( t \) is the time, and each \( x_t^{(i)} \in S \times M \), for \( i = 1..N \). The set of weights corresponding to the particles at time \( t \) is denoted as \( w_t = \{ w_t^{(1)}, w_t^{(2)}, w_t^{(3)}, \ldots, w_t^{(N)} \} \), such that, \( \sum_i w_t^{(i)} = 1 \). The posterior distribution is then approximated by the particles as described in equation 5.1.

The dynamic model is denoted \( P(x_t | x_{t-1}) \), and is derived from the activity Petri Net as described in section 8.2.5. The proposal distribution, from which samples are drawn, is set to be equal to the dynamic model. This is a simplifying assumption that is adopted in many works utilizing the particle filter framework.

### 8.3.1 Initialization

The particle set \( x_0 \) is initialized by sampling from the prior probability distribution, \( P(x_0) \), defined above. The corresponding weights are initialized to \( 1/N \), where \( N \) is a parameter indicating the number of particles.

For \( i = 1 \ldots N \)

\[ x_0^{(i)} \sim P(x_0), \quad w_0^{(i)} = 1/N \]
8.3.2 Update

At each frame \( t \) from 1 to \( T \), we denote the observation vector as \( y_t \). Recall that this vector resides in \([0,1]^{|O|}\), where each entry \( y_{t(i)} \), represents the observation certainty in the \( i \)th event in set \( O \). For each frame \( t \) we do the following:

1. Sample a new set of particles from the proposal distribution using the set of particles from the previous frame (the prediction step of the Bayesian Recursive Filter):
   
   For \( i = 1 \ldots N \)
   
   \[
   x_t^{(i)} \sim P(x_t|x_{t-1}^{(i)})
   \]

2. Update the weights of each particle using the observation as follows (the correction step of the Bayesian Recursive Filter):
   
   For \( i = 1 \ldots N \)
   
   \[
   \hat{w_t}^{(i)} = w_{t-1}^{(i)} \ast P(y_t|x_t^{(i)})
   \]

3. Normalize the weights:
   
   For \( i = 1 \ldots N \)
   
   \[
   w_t^{(i)} = \frac{\hat{w_t}^{(i)}}{\sum_i \hat{w_t}^{(i)}}
   \]

As previously mentioned, the particle weights provide an approximation of the posterior probability distribution, \( P(x_t|y_{1:t}) \). Thus, summing the weights of those particles where \( x_{t(m)} \in R \), allows determining the probability that the activity is recognized.

8.4 An Example

Consider the Petri Net definition of an activity depicted in Figure 8.3. For simplicity we only consider the arcs shown in the figure and do not include edges...
modeling non-activity context. Now let us illustrate the application of our approach on this net.

8.4.1 Constructing the Bayes Recursive Filter Components

Table 8.2 shows $S$, the set of states reachable from the initial state, corresponding to the activity PN in Figure 8.3. The set $R$ of recognized states consists of a single state $\sigma_{10}$. $O$, the set of observable events, implied by Figure 8.3 is as follows:

$$O = \{\text{Visitor Appears, Teller Appears, Teller in Safe, Visitor in Safe, Visitor Disappears}\}$$

(8.11)

The set $M$ is the set of relevant event combinations. These are events which may occur simultaneously according to the activity PN in Figure 8.3.

$$M = \{\emptyset, \{\text{Visitor Appears}\}, \{\text{Teller Appears}\}, \{\text{Teller in Safe}\}, \{\text{Visitor in Safe}\},$$

$$\{\text{Visitor Disappears}\}, \{\text{Visitor Appears, Teller Appears}\}\}$$

(8.12)

Supposing that the initial marking of the Petri Net in Figure 8.3 is $\sigma_1$, we can define the following simple prior probability:

$$P(x_0) = P(x_0(m), x_0(e)) = \begin{cases} 1 & \text{if } x_0(m) = \sigma_1 \text{ and } x_0(e) = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

(8.13)

Let us now consider the construction of the dynamic model. The first component of the dynamic model is $P(x_t(m)|x_{t-1}(m), x_{t-1}(e))$, which describes the Petri Net dynamics. Recall that, $x_{t-1}(m)$ denotes the Petri Net marking in the previous frame , $x_{t-1}(e)$ denotes the set of events (if any) that occurred in the previous frame, and $x_t(m)$ denotes the resulting marking.

Consider the case of $x_{t-1}(m) = \sigma_2$, in which the marking in the previous frame is one where places $P_2$ and $P_3$ contain tokens. By examining Figure 8.3, we can see that if event $\text{Visitor Appears}$ occurs (i.e. $x_{t-1}(e) = \{\text{Visitor Appears}\}$) the resulting marking would be one where places $P_3$ and $P_4$ have tokens (marking $\sigma_4$.

<table>
<thead>
<tr>
<th>State</th>
<th>$\sigma_1$</th>
<th>$\sigma_2$</th>
<th>$\sigma_3$</th>
<th>$\sigma_4$</th>
<th>$\sigma_5$</th>
<th>$\sigma_6$</th>
<th>$\sigma_7$</th>
<th>$\sigma_8$</th>
<th>$\sigma_9$</th>
<th>$\sigma_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Places w/ token</td>
<td>$P_1, P_2, P_3$</td>
<td>$P_2, P_3$</td>
<td>$P_3, P_4$</td>
<td>$P_1, P_5$</td>
<td>$P_6$</td>
<td>$P_7$</td>
<td>$P_8$</td>
<td>$P_9$</td>
<td>$P_{10}$</td>
<td></td>
</tr>
</tbody>
</table>

Table 8.2: The reachable states of the activity depicted in Figure 8.3.
in Table 8.2. We model these dynamics as a discrete probability distribution as follows:

\[
P(x_{t(m)}|x_{t-1(m)} = b, x_{t-1(e)} = \{Visitor Appears\}) = \begin{cases} 
1 & \text{if } x_{t(m)} = \sigma_4 \\
0 & \text{otherwise}
\end{cases} \quad (8.14)
\]

We use the same process to define the remainder of \(P(x_{t(m)}|x_{t-1(m)}, x_{t-1(e)})\):

<table>
<thead>
<tr>
<th>(x_{t(m)} / x_{t-1(m)}, x_{t-1(e)})</th>
<th>(\sigma_1 \emptyset)</th>
<th>(\sigma_2 \emptyset)</th>
<th>(\sigma_2, {Visitor Appears})</th>
<th>(\sigma_2, {Teller Appears})</th>
<th>(\sigma_2, {Visitor Appears, Teller Appears})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma_1)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\sigma_2)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\sigma_3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\sigma_4)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The second component of the dynamic model, \(P(x_{t(e)}|x_{t(m)})\), defines a distribution over the events that can occur at frame \(t\). Here \(x_{t(m)}\) denotes the underlying Petri Net marking at time \(t\), and \(x_{t(e)}\) denotes a (possibly empty) set of events.

Again consider the case of \(x_{t(m)} = \sigma_2\) (places \(P_2\) and \(P_3\) contain tokens). Examining Figure 8.3 we see that the two transitions, respectively labeled with events \(Visitor Appears\) and \(Teller Appears\) are both enabled in this marking. Thus the events \(\{Visitor Appears\}\) and \(\{Teller Appears\}\) can occur. Additionally, since the two transitions are not in conflict (firing one would not disable the other), the event combination \(\{Visitor Appears, Teller Appears\}\) can also occur in this marking. Recall also that the empty event set, \(\emptyset\), is enabled in all markings. Thus

\[
\hat{P}(x_{t(e)}|x_{t(m)} = \sigma_2) = \begin{cases} 
1 & x_{t(e)} = \emptyset \text{ or } \{Visitor Appears\} \text{ or } \{Teller Appears\} \text{ or } \{Visitor Appears, Teller Appears\} \\
0 & \text{otherwise}
\end{cases} \quad (8.15)
\]

After the normalization step we get:

\[
P(x_{t(e)}|x_{t(m)} = \sigma_2) = \begin{cases} 
1/2 & \text{if } x_{t(e)} = \emptyset \text{ or } \{Visitor Appears\} \text{ or } \{Teller Appears\} \text{ or } \{Visitor Appears, Teller Appears\} \\
0 & \text{otherwise}
\end{cases} \quad (8.16)
\]

The remainder of \(P(x_{t(e)}|x_{t(m)})\) is calculated similarly:

<table>
<thead>
<tr>
<th>(x_{t(e)} / x_{t(m)})</th>
<th>(\sigma_1)</th>
<th>(\sigma_2)</th>
<th>(\sigma_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>1/2</td>
<td>1/2</td>
<td></td>
</tr>
<tr>
<td>{Visitor Appears}</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>{Teller Appears}</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
</tr>
<tr>
<td>{Visitor Appears, Teller Appears}</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
</tr>
</tbody>
</table>

Now what remains is to construct the measurement model. Recall that for \(i = 1 \ldots |O|\) we construct a separate model, \(P(y_{t(i)}|x_{t(e)})\), for each event \(O(i)\). As an example, let us suppose the first event in an arbitrary ordering of the elements of \(O\) is \(Visitor Appears\), so that \(O(1) = Visitor Appears\) and \(y_{t(1)}\) is the
certainty with which \textit{Visitor Appears} is observed at frame $t$. Since, of all the sets in $M$, $O(1) = \textit{Visitor Appears}$ is only a member of the sets $\{\textit{Visitor Appears}\}$ and $\{\textit{Visitor Appears, Teller Appears}\}$ we set

$$P(y_{t(1)}|x_{t(e)}) = \begin{cases} \varphi(y_{t(1)}) & \text{if } x_{t(e)} = \{\textit{Visitor Appears}\} \text{ or } \{\textit{Visitor Appears, Teller Appears}\} \\ 1 - \varphi(y_{t(1)}) & \text{otherwise} \end{cases}$$

The remainder of the measurement model is similarly constructed:

$$P(y_{t(i)}|x_{t(e)}):$$

\begin{tabular}{|c|c|c|c|}
\hline
\textit{Visitor Appears} & \textit{Teller Appears} & \textit{Visitor Appears, Teller Appears} \\
\hline
$1 - \varphi(y_{t(1)})$ & $\varphi(y_{t(1)})$ & $\varphi(y_{t(1)})$ \\
$1 - \varphi(y_{t(2)})$ & $\varphi(y_{t(2)})$ & $\varphi(y_{t(2)})$ \\
\hline
\end{tabular}

\subsection{Propagating Certainty}

Now let us use the above construction, to make the particle filter mechanism more concrete. We shall use an example with $N = 3$ particles. We initialize these particles by sampling from the prior distribution function, $P(x_0)$, described above. One likely sampling from this distribution is:

$$x_0^{(1)} = \{\sigma_1, \emptyset\}, x_0^{(2)} = \{\sigma_1, \emptyset\}, x_0^{(3)} = \{\sigma_1, \emptyset\}$$

where $x_0^{(1)} = \{\sigma_1, \emptyset\}$ is shorthand for $x_{0(1)} = \sigma_1, x_{0(2)} = \emptyset$. We also initialize the weights to $1/N$

$$w_0^{(1)} = 1/3, w_0^{(2)} = 1/3, w_0^{(3)} = 1/3$$

Now let us consider how the activity recognition module may process the hypothetical input of the event recognition layer. Let us suppose at time $t = 1$, the event \textit{Teller Appears} is recognized with 0.8 certainty and no other events are recognized. The resulting observation vector $y_1$ will have 5 components ($O$ has 5 members), all of which will have a 0 certainty, save for component $j$ which represents the event \textit{Teller Appears} ($y_{1(j)} = 0.8$). Using our algorithm, we initially sample from the proposal distribution, which is equivalent to the dynamic model. This is done by initially sampling from $P(x_{t|m})|x_{t-1|m}, x_{t-1(e)}$ and then using the sampled result to sample from $P(x_{t(e)}|x_{t|m})$.

Continuing our example, sampling $x_{1|m}^{(1)} \sim P(x_{t|m})|x_{t-1|m} = \sigma_1, x_{t-1(e)} = \emptyset$ would result in $x_{1|m}^{(1)} = \sigma_2$ (the only outcome with non-zero probability). Then, $x_{1(e)}^{(1)} \sim P(x_{t(e)}|x_{t|m} = \sigma_2)$ could result in any one of four outcomes, each with equal probability ($\emptyset, \{\textit{Visitor Appears}\}, \{\textit{Teller Appears}\}, \{\textit{Visitor Appears, Teller Appears}\}$). Let us assume one of several possible result of such a sampling:

103
\( x_1^{(1)} = \{\sigma_2, \{\text{Teller Appear}\}\}, x_2^{(2)} = \{\sigma_2, \emptyset\}, x_1^{(3)} = \{\sigma_2, \{\text{Visitor Appear}\}\} \)

Since \( x_1^{(1)} = \{\sigma_2, \text{Teller Appear}\} \)

\[
P(y_{1(i)}|x_1^{(1)}) = \begin{cases} 
\varphi(y_{1(i)}) & \text{if } i = j \\
1 - \varphi(y_{1(i)}) & \text{otherwise}
\end{cases} \quad (8.18)
\]

Thus since \( y_{1(i)} = 0 \) for all \( i \neq j \) and \( y_{1(j)} = 0.8 \) in our example:

\[
P(y_1|x_1^{(1)}) = \prod_{i=1}^{\lvert O \rvert = 5} P(y_{1(i)}|x_1^{(1)}) = \varphi(0.8) \cdot (1 - \varphi(0))^4 \quad (8.19)
\]

Updating the weights is then done according to:

\[
\hat{w}_1^{(1)} = w_0^{(1)} \cdot P(y_1|x_1^{(1)}) = w_0^{(1)} \cdot \varphi(0.8) \cdot (1 - \varphi(0))^4 = .3091 \quad (8.20)
\]

where we are using

\[
\varphi(z) = \frac{1}{1 + e^{-k(z-0.5)}} \quad (8.21)
\]

with \( k = 10 \)

We adjust the other particle weights similarly:

\[
\hat{w}_1^{(2)} = w_0^{(2)} \cdot P(y_1|x_1^{(2)}) = w_0^{(2)} \cdot (1 - \varphi(0.8)) \cdot (1 - \varphi(0))^4 = .0154 \quad (8.22)
\]

\[
\hat{w}_1^{(3)} = w_0^{(3)} \cdot P(y_1|x_1^{(3)}) = w_0^{(3)} \cdot (1 - \varphi(0.8)) \cdot (1 - \varphi(0))^3 \cdot \varphi(0) = 1.03e - 4 \quad (8.23)
\]

Our final step is to normalize our weights:

\[
\sum_i \hat{w}_1^{(i)} = .3246 \quad (8.24)
\]

\[
w_1^{(1)} = \frac{\hat{w}_1^{(1)}}{.3246} = .9522 \quad (8.25)
\]

\[
w_1^{(2)} = \frac{\hat{w}_1^{(2)}}{.3246} = .0474 \quad (8.26)
\]

\[
w_1^{(3)} = \frac{\hat{w}_1^{(3)}}{.3246} = .0001 \quad (8.27)
\]
Chapter 9
Factoring Context

One major failing of the current approaches to activity representation and recognition, including the one proposed in the previous chapter, is that they fail to separate the physical constraints of the scene (which we call context) and the constraints of the activity. For example, a simple activity might consist of a person entering zone 1 before entering zone 2 (activity constraints). However, there does not exist a physical constraint preventing the person from entering zone 2 first (context constraints). The absence of this separation leads to complex activity models, which endeavor to contain both the activity and context constraints. This type of modeling assumes that everything that occurs in the video sequence occurs within the context of the activity. This assumption is not valid for general video sequences in the surveillance video domains. For instance in the bank scenario considered in our experiments, the bank is not always being robbed. It is possible for the customer to enter the bank, the cashier to stand behind the counter, and even the cashier to enter the safe during the course completely legitimate activities, even though these are all components of a “Bank Attack” activity. Note that, as in the previous chapter, we are using the term activity to mean multi-thread complex event, and the term event to mean an atomic sub-event of the activity restricted to a temporal interval (with respect to the terminology defined in Chapter 2).

This chapter describes an extension to the previous chapter: FPFPN (Factored Particle Filter Petri Net), an activity representation and recognition approach which incorporates the intuition described above. That is, the context state and activity state are modeled and estimated separately. This intuition yields simpler activity models, that are more straightforward to construct, and achieve improved recognition results. Furthermore, the factored nature of our approach, results in a reduced state space. Thus, less samples(particles) are required to estimate the state at every time instance. As the complexity of particle filter approaches is a (linear) function of the number of particles used, our construction
yields a more efficient recognition algorithm.

In this chapter, we approach the activity recognition problem from a different direction, based on the intuition that uncertain event observations are dependent on the state of the scene (which we call the context). Thus, we can use our event observations to estimate the context. The progress of the activities that we are interested in is also dependent on the evolution of this context. That is, given the context, the activity state is independent of the event observations themselves. Thus an estimation of the context can be used to estimate the progress of the activity. In most cases the constraints of the scene are few, compared to the constraints of an activity, and can be modeled in a straightforward fashion. This approach also simplifies the modeling of activities since each activity is now not burdened with modeling of events outside the activity context.

In this chapter we will provide the details of our model construction and inference procedures. Our model construction creates a Petri Net which is segmented into several fragments. These fragments are of one of two types. Context fragments model all relevant events that can occur in a particular surveillance domain. Activity fragments model temporal constraints on event ordering within activities. Once these fragments are defined, a Bayes recursive framework (BRF) can be inferred. Within this framework we apply a particle filter based estimation of the state. This estimation is divided into two parts estimation of the context, and estimation of the activity state.

9.1 Constructing the Petri Net Activity Model

9.1.1 Constructing Context Fragments

Context fragments model all relevant states an object can take on as well as the relevant events in the event domain. These fragments capture the physical constraints of the domain, independent of any activities than can occur. Each transition node, in a context fragment is labeled with an associated event that may be observed during the processing of a video sequence. Each place in the event context fragment can also be associated with a semantic meaning. Often there will be several independent context fragments representing different facets of the scene object state (e.g. which area of the scene the object is in, how fast the object is moving, how close an object is to other objects, etc.). These fragments will be unconnected.

9.1.2 Constructing Activity Fragments

Each relevant activity is represented by several Petri Net fragments. These fragments represent the temporal constraints on event ordering that the activity
contains. Hence we refer to these as constraint fragments. Each such constraint fragment will make use of a template fragment which represents the particular type of temporal constraint. The template fragment is then specialized according to the parameters of the constraint (i.e., the specific events participating in the constraint). This specialization is achieved by labeling the appropriate transition nodes in the constraint fragment according to the constraint’s dependence on the context fragment marking.

### 9.1.3 The Template Fragments

Figure 9.1 shows the most common template fragments used in constructing our activity models. These relations correspond to commonly used interval relations: Before, Overlap and During. The General Overlap relation is meant for the case in which two events are constrained to occur at the same time but their starting and ending points are not relevant to the activity definition. Although this relation could be constructed by combining a number of other fragments, having a separate simpler relation fragment makes modeling this type of constraint less demanding on the recognition process.

In the paradigm of activity recognition we utilize an online approach to relation detection. That is, we would like to recognize a particular activity and all its composing constraints as soon as they occur. Consider the interval relation such as $A \text{ During } B$, where $B$ is significantly longer. In an online recognition, evaluated at every frame, waiting until interval $B$ is concluded would result in a delayed recognition of the interval relation. A relaxed approach to the during relation, which permits recognition of the relation after interval $A$ ends (under the assumption that interval $B$ ends eventually), results in a much earlier detection of the relation. Using this relaxed approach temporal constraints can be validated earlier than they would be under the strict approach, and thus earlier recognition of activities is possible.

### 9.1.4 Chaining Relations

When modeling activities in the surveillance domain it is often necessary to chain relations. That is, to define a relation between the occurrence of a relation and the occurrence of an event interval (or another relation). For example, we may require event $A$ to occur before event $B$, and this before relation to occur during event $C$. We facilitate this modeling functionality by connecting the fragment of the inner relation to the fragment of the fragment of the outer relation in the appropriate way. More specifically, for each relation template we define the markings for which the relation is "on" (in progress) and the markings for which the relation is "off". We can now substitute these markings, for the markings...
in the context fragment, which primitive relations rely on. Figure 9.2 illustrates this construction.

### 9.2 Constructing the Probabilistic Model

In order to apply particle filter techniques to our approach we have to translate our formulation of an activity, specified as a Petri Net, into the language of the Bayesian Recursive Filter (BRF). More specifically, we have to define the space of possible states our system can take on at each time step, and the space of possible observations at each time step. Furthermore, we must define the dynamic model, \( P(x_t|x_{t-1}) \), which describes the evolution of states in time. As well as the measurement model, \( P(y_t|x_t) \), which defines the likelihood of a particular observation given the system is in a particular state. We will take advantage of the disjoint fragments of our Petri Net model construction in order to construct a BRF framework, where efficient state estimation is possible.

#### 9.2.1 Notation

To keep our formulations concise we introduce the following notation to denote vector valued variables:

\[ {^{i}x^{j}_{t}} \]

Here \( i \) denotes the factor, \( t \) denotes the time, and \( j \) denotes the particle id. To indicate a particular component of a vector we add the additional index \( c \):

\[ {^{i}x^{j}_{t(c)}} \]

some of these indices may be omitted as appropriate, but their configuration is kept consistent throughout the chapter.

#### 9.2.2 Preliminaries

As we have detailed in previous section, the activity models in our approach are composed of multiple Petri Net fragments. In this section we will refer to these fragments as factors, each with its own state, which in combination form the system state. Like their corresponding fragments, these factors are divided into two groups, context factors and activity factors. We will denote the set of all factors as \( F \), the set of all context factors as \( C \), and the set of all activity relation factors as \( A \). Clearly \( C \cup A = F \) and \( C \cap A = \emptyset \). The set of activity labels in the model is denoted \( \text{activities} \). We also define the function \( \delta : A \rightarrow \text{activities} \), as a map from each activity relation fragment to the activity it is involved in. Given
a particular Petri Net fragment $i$, coupled with an initial marking, we can derive the set of all reachable markings. We denote this set using the letter $S_i$. The fragment $i$ also defines the set of all relevant events that may be observed. These events are those used to label the transition nodes in the Petri Net fragment. Note that since multiple transitions may have the same event label, the number of events is not necessarily equal to the number of transitions. We denote the set of all events in fragment $i$ by $O_i$. Since multiple events may be observed at the same time we define another set, $M_i$, as a subset of the powerset of $O_i$. This set serves as the space over all relevant event sets that may occur during the video analysis. Note that $M_i$, trivially contains $\emptyset$, the empty set.

### 9.2.3 The State Space

In order to model Petri Net dynamics in this paradigm we must consider the (possibly empty) set of events that occur at each frame as part of the state. The state of our model is conveniently divided into independent factors which allows us to estimate the joint state over all factors by estimating the state of each factor separately.

Thus, $x_t$, the state variable at frame $t$ of the video sequence, will be factored into several variables representing the various factors of the activity model. That is $x_t = <^1x_t, ^2x_t, \ldots, ^Fx_t>$. Each of these components $^ix_t = <^ix_{t(m)}, ^ix_{t(e)}>,$ will take on a value with two components. The first component, $^ix_{t(m)}$, will denote the marking of PN factor $i$ at time $t$. The second component, $^ix_{t(e)}$, will denote the event set relevant to factor $i$, that occurs at time $t$.

More formally, $^ix_{t(m)} \in S_i$ and $^ix_{t(e)} \in M_i$. Thus, the state space of $^ix_t$ is $S_i \times M_i$.

### 9.2.4 The Observation Space

At each frame we will observe a certainty value for each relevant event in set $O$. Thus the observation is a vector, $y_t \in [0,1]^{|O|}$. Since the observation is independent of the activity state given the context state (see next section), $y_t$ will be factored according to the various context factors. That is, $y_t = <^1y_t, ^2y_t, \ldots, ^Cy_t>$. Each $^iy_t \in [0,1]^{|O_i|}$ is a vector. The $j$-th entry in vector $^iy_t$, which we will denote $^iy_{t(j)}$, denotes the certainty in the observation of $O_i(j)$ the $j$-th event in set $O_i$, according to an arbitrary pre-defined order.

### 9.2.5 Conditional Independence

In addition to defining the space of our state and observation variables, we also assume some conditional independence relationships among the variables. First,
we make use of the (first-order) Markov Assumption, which asserts that the state at any frame \( t \) is independent of the state at all previous frames, given the state at frame \( t-1 \). Hence, it is only necessary to consider the dependence between the states at frames \( t \) and \( t-1 \).

Recall that our state space for each factor is decomposed into two components: marking and event set. Each marking at time \( t \) is dependent on the marking at time \( t-1 \) as well as those event set that occurred at time \( t-1 \). Furthermore, the event set that occurs at time \( t \) is dependant only on the marking at time \( t \). This assumptions are a natural modeling of the Petri Net dynamics and are illustrated by a graphical model in Figure 9.4.

Another assumption made in our model is that the context state fully determines the observation. In other words, the observation is independent of the activity state given the context state. These assumption is illustrated in the simple graphical model shown in Figure 9.3.

Now let us take a closer look at how the activity state is determined. Like the context state, the activity state is composed of two components: the marking and event set. Similar to the context state, each activity state marking at time \( t \) is determined by the activity state marking at time \( t-1 \) as well as the events that occurred at time \( t-1 \). However, unlike the context state, the activity state events that occur at time \( t \) are not determined only by the activity state marking at time \( t \), but rather by a combination of the activity state marking and the context state marking at time \( t \). These dependencies are shown in the graphical model in Figure 9.5.

### 9.2.6 The Dynamic model

In this section we will derive a Dynamic model from the previously defined Petri Net Model Structure. In doing so we will translate transitions labeled with events into a probability of whether these transitions have fired (upon observation of the corresponding event). As such we will mildly abuse terminology by referring to enabled transitions as enabled events. Similarly we may say “all transitions in an event set are enabled”, instead of the more precise “ all events in the event set have a corresponding transition node which is enabled”.

In constructing the dynamic model, \( P(x_t|x_{t-1}) \), let us consider it in two decomposable pieces, the context dynamic model, denoted \( P_c(x_t|x_{t-1}) \), and the activity dynamic model, denoted \( P_a(x_t|x_{t-1}) \). We will make use of a simplifying assumption that these pieces are independent of one another such that:

\[
P(x_t|x_{t-1}) = P_c(x_t|x_{t-1}) \cdot P_a(x_t|x_{t-1}) \tag{9.1}
\]

Let us first consider the decomposition of the context dynamic model \( P_c(x_t|x_{t-1}) \):
\[ P_c(x_t|x_{t-1}) = \prod_{i \in C} P(i_{t1}|x_{t-1}) = \prod_{i \in C} P(i_{t1(m)}, i_{t(e)}|i_{t-1(m)}, i_{t-1(e)}) \quad (9.2) \]

\[ = \prod_{i \in C} P(i_{t1(m)}|i_{t-1(m)}, i_{t-1(e)}) P(i_{t(e)}|i_{t(m)}) \quad (9.3) \]

where the second equality is due to the independence between context factors. The third equality is due to the conditional independence relationships discussed in the previous section (see Figure 9.4).

Similarly, the activity dynamic model, \( P_a(x_t|x_{t-1}) \), is decomposed as follows:

\[ P_a(x_t|x_{t-1}) = \prod_{i \in A} P(i_{t1}|x_{t-1}) = \prod_{i \in A} P(i_{t1(m)}, i_{t(e)}|i_{t-1(m)}, i_{t-1(e)}) \quad (9.4) \]

\[ = \prod_{i \in A} P(i_{t1(m)}|i_{t-1(m)}, i_{t-1(e)}) P(i_{t(e)}|i_{t(m)}) \]

note that in the latter decomposition we used the term \( P(i_{t(e)}|i_{t(m)}) \) instead of \( P(i_{t(e)}|i_{t(m)}, C_t) \). This is a simplifying assumption, which indicates that although the events in the activity factor \( i \) at time \( t \), depend on the context, our simplified model will not consider this dependence for the time being (we will correct for this during inference). Using the above models it suffices to derive \( P(i_{t1}|i_{t(m)}) \) and \( P(i_{t(e)}|i_{t-1(m)}, i_{t-1(e)}) \) for each factor \( i \) from the Petri Net structure. As no information is available on the first component of the dynamic model, we define an uninformative \( P(i_{t(e)}|i_{t(m)}) \), for some factor \( i \) giving equal probability to all enabled transitions from the structure of the Petri Net factor as follows:

\[ \hat{P}(i_{t1}|i_{t(m)}) = \begin{cases} 
1 & \text{if all transitions in } i_{t(e)} \text{ are enabled in marking } i_{t(m)} \\
0 & \text{otherwise} 
\end{cases} \quad (9.5) \]

where \( i_{t(e)} = \emptyset \) (the empty set) is considered to be enabled in all markings. We then normalize

\[ P(i_{t1}|i_{t(m)}) = \frac{\hat{P}(i_{t1}|i_{t(m)})}{\sum_{x^e \in M_i} \hat{P}(i_{t1}|x^e_{t(m)})} \quad (9.6) \]

Similarly, \( P(i_{t1}|i_{t-1(m)}, i_{t-1(e)}) \) is derived from the Petri Net fragment \( i \) as follows:
\[
\hat{P}(^ix_t|m) | ^ix_{t-1}(m), ^ix_{t-1}(e) = \begin{cases} 
1 & \text{if } ^ix_t(m) \leftarrow ^ix_{t-1}(m) | ^ix_{t-1}(e) \\
0 & \text{otherwise} 
\end{cases} \tag{9.7}
\]

where \(^ix_t(m) \leftarrow ^ix_{t-1}(m) | ^ix_{t-1}(e)\) indicates there is a path from state \(^ix_{t-1}(m)\) to state \(^ix_t(m)\) via transitions labeled with the event(s) \(^ix_{t-1}(e)\). We then normalize:

\[
P(\ ^ix_t | m) | ^ix_{t-1}(m), ^ix_{t-1}(e) = \frac{\hat{P}(^ix_t(m) | ^ix_{t-1}(m), ^ix_{t-1}(e))}{\sum_{x' \in S} \hat{P}(x' | ^ix_{t-1}(m), ^ix_{t-1}(e))} \tag{9.8}
\]

### 9.2.7 The Measurement Model

Like the dynamic model the measurement model, \(P(y_t|x_t)\), can also be decomposed. First we use our assumption that only the context state factors determine the observation, and each context state factor has its own disjoint set of observations.

\[
P(y_t|x_t) = \prod_{i \in C} P(\ ^iy_t | ^ix_t) \tag{9.9}
\]

We also make use of the assumption that each observation is independent of all other observations. Thus, for each factor \(i\):

\[
P(\ ^iy_t | ^ix_t) = \prod_{j=1}^{O_i} P(\ ^iy_{t(j)} | ^ix_t) \tag{9.10}
\]

where \(^iy_{t(j)}\) is the \(j\)-th component of the vector \(^iy_t\). For some factor \(i\) and vector component \(j\) we can further simplify our formula:

\[
P(\ ^iy_{t(j)} | ^ix_t) = P(\ ^iy_{t(j)} | ^ix_{t(m)}, ^ix_{t(e)}) = P(\ ^iy_{t(j)} | ^ix_{t(e)}) \tag{9.11}
\]

where the second equality is due to the conditional independence relationships described in Figure 9.4.

Thus, for each (context) Petri Net factor \(i\) it suffices to derive \(P(\ ^iy_{t(j)} | ^ix_{t(e)})\) from the Petri Net structure, to obtain the measurement model. The construction of the observation likelihood, \(P(\ ^iy_{t(j)} | ^ix_{t(e)})\), models the notion that if a particular event occurs, the observed certainty in this event should be high. That is, if the event set occurring at time \(t\), denoted \(^ix_{t(e)}\), contains the \(j\)-th event \(O_i(j)\), the certainty in the observation \(^iy_{t(j)}\) should be high. The sigmoid function (centered on 0.5) is a natural fit for modeling this intuition. Conversely if an event does
not occur, the observed certainty in this event should be low. The complement to the sigmoid function models this situation. Each vector component in \( {\mathbf{y}}_t \) represents certainty in one of the events in set \( O \). Let us use \( O_i(j) \) to denote the event whose certainty is represented by the \( j \)-th component of \( {\mathbf{y}}_t \). Since, \( ^i x_{t(e)} \in M_i \subseteq 2^{O_i} \) (the powerset of \( O_i \)) we can check the membership of \( O_i(j) \) in the set \( ^i x_{t(e)} \) thus:

\[
P(^i y_{t(j)} | ^i x_{t(e)}) = \begin{cases} \varphi(^i y_{t(j)}) & \text{if } O_i(j) \in ^i x_{t(e)} \\ 1 - \varphi(^i y_{t(j)}) & \text{otherwise} \end{cases} \tag{9.12}
\]

where \( \varphi(z) \) is the sigmoid function centered on 0.5 and \( k \) is a parameter:

\[
\varphi(z) = \frac{1}{1 + e^{-k \cdot (z-0.5)}} \tag{9.13}
\]

### 9.2.8 The Prior Probability

The prior probability over the marking variable of the state, \( P(^i x_{0(m)}) \), is dictated by the initial marking of the Petri Net Fragment \( i \):

\[
P(^i x_{0(m)}) = \begin{cases} 1 & \text{if } ^i x_{0(m)} = \text{initial marking} \\ 0 & \text{otherwise} \end{cases} \tag{9.14}
\]

### 9.3 Performing Inference

Recall that our objective in this work is to translate a list of events with associated certainty values into an activity label (with its own associated certainty value). In this section we describe the mechanics of this “propagation” of certainty from the event level to the activity level. Our approach to the propagation of certainty is based on the particle filter. This approach attempts to estimate the system state based on all available information. Recall that this state is decomposed into the context state, which estimates the properties of all objects in the scene, and the activity state which estimates how many activities (and their components) have been recognized. Throughout our analysis of the video we will maintain a set of particles for each factor of the activity model. Each particle contains a hypothesis of the current state of the factor (see section 9.2.3) and corresponding weight. Together these particles will approximate the posterior distribution over the factor, \( P(^i x_t | ^i y_{1:t}) \). This distribution is the estimation of the factor state given all information we have seen so far.

Since all factors are independent we can compute the total posterior distribution using the formula:
$$P(x_t|y_{1:t}) = \prod_{i \in F} P(i x_t|i y_{1:t}) \quad (9.15)$$

Formally, we denote by the set of particles for factor $i$ at time $t$ as, $^iX_t = \{^i x_t^{(1)}, ^i x_t^{(2)}, ^i x_t^{(3)}, \ldots, ^i x_t^{(N)}\}$, where $N$ is the number of particles, and each $^i x_t^{(j)} \in S_t \times M_t$, for $j = 1..N$. The set of weights corresponding to the particles for factor $i$ at time $t$ is denoted as $^i w_t = \{^i w_t^{(1)}, ^i w_t^{(2)}, ^i w_t^{(3)}, \ldots, ^i w_t^{(N)}\}$, such that, $\sum_{j=1}^N i w_t^{(j)} = 1$.

The posterior distribution is then approximated by the particles as follows:

$$P(i x_t|i y_{1:t}) = \sum_{j=1}^N i w_t^{(j)} \delta(^i x_t, ^i x_t^{(j)}) \quad (9.16)$$

where $\delta$ indicates the Dirac delta function.

The dynamic model is denoted $P(x_t|x_{t-1})$, and is derived from the various Petri Net fragments as described in section 9.2.6. The proposal distribution, from which samples are drawn, is set to be equal to this dynamic model. This is a simplifying assumption that is adopted in many works utilizing the particle filter framework.

### 9.3.1 Initialization

The particle set $^i X_0$, is initialized in two stages. First we sample the marking component of the state variable from the prior probability distribution, $P(^i x_0^{(m)})$ (see section 9.2.8) defined above. Second, we sample the event combination component of the state variable from $P(^i x_0^{(e)}|^i x_0^{(m)})$, using the dynamic model (this is a special case of the prediction step). The corresponding weights are initialized to $1/N$, where $N$ is a parameter indicating the number of particles.

For $j = 1 \ldots N$

$$^i x_0^{(j)} \sim P(^i x_0^{(m)})$$

$$^i x_0^{(j)} \sim P(^i x_0^{(e)}|^i x_0^{(m)})$$

$$^i w_0^{(j)} = \frac{1}{N}$$
9.3.2 Update

At each frame $t$ from 1 to $T$, we denote the observation vector as $y_t$. Recall that this vector resides in $[0, 1]^{\mid O \mid}$, where each entry represents the observation certainty in one of the events in set $O$. This vector is given as input from the event recognition layer. Each factor has its own subset of relevant observation components, denoted $y_t$.

For each frame $t$ we do the following:

1. for each context factor $i \in C$
   
   1.1. Update the weights of each particle using the observation as follows (the correction step of the Bayes Recursive Filter):
   
   For $j = 1 \ldots N$
   $$i \hat{w}_t^{(j)} = i w_{t-1}^{(j)} \cdot P(i y_t | i x_t^{(j)})$$
   (9.17)

   1.2. Normalize the weights:
   
   For $j = 1 \ldots N$
   $$i w_t^{(j)} = \frac{i \hat{w}_t^{(j)}}{\sum_l i \hat{w}_t^{(l)}}$$
   (9.18)

   1.3. Sample the next set of particles from the proposal distribution (the prediction step of the Bayes Recursive Filter):
   
   For $j = 1 \ldots N$
   $$i x_t^{(j)} \sim P_e(i x_t | i x_{t-1}^{(j)})$$
   (9.19)

2. for each activity relation factor $k \in A$

   2.1. Update the weights of each particle using the observation as follows (the correction step of the Bayes Recursive Filter):
   
   For $j = 1 \ldots N$
   $$k \hat{w}_t^{(j)} = k w_{t-1}^{(j)} \cdot \frac{P(C(k x_{t(e)}^{(j)}, k x_{t(m)}^{(j)}))}{P(k x_{t(e)}^{(j)} | k x_{t(m)}^{(j)})}$$
   (9.20)

   Here $P(C(k x_{t(e)}^{(j)}, k x_{t(m)}^{(j)}))$ denotes the marginal probability over the context fragment for the condition required to enable event(s) $k x_{t(e)}^{(j)}$ while in marking $k x_{t(m)}^{(j)}$.

   2.2. Normalize the weights:
   
   For $j = 1 \ldots N$
   $$k w_t^{(j)} = \frac{k \hat{w}_t^{(j)}}{\sum_l k \hat{w}_t^{(l)}}$$
   (9.21)
2.3. Sample the next set of particles from the proposal distribution (the prediction step of the Bayes Recursive Filter):

For \( j = 1 \ldots N \)

\[
k^{(j)} \sim P_a(x_i | k^{(j)} x_{t-1})
\]  

(9.22)

As previously mentioned, the particle weights provide an approximation of the posterior probability distribution, \( P(x_t | y_{1:t}) \). Calculating the marginal over all recognized places in each of the activity fragments and computing the product of these marginals, gives us the certainty in the occurrence of each of the activities.

9.3.3 Calculating the Marginals

During the inference process we are often interested in calculating the probability of the context state configuration being consistent with some logical clause of literals (e.g. \( E_1 \land \neg D_1 \)), representing the conditions under which a transition in one of the activity fragments should be allowed to fire. Each literal in this clause represents a place node in one of the context fragments. For example, in the clause above, literal \( E_1 \) represents place node \( E \) in context fragment \( c_1 \) (depicted in Figure 9.6).

Let us define a positive literal to be consistent with some marking \( m \) of the corresponding context fragment, if the place node referred to by the literal contains one or more tokens in marking \( m \). For example, the positive literal \( E_1 \) is consistent with any marking of context fragment \( c_1 \) in which place node \( E \) contains a token.

Similarly, a negative literal is defined to be consistent with some marking \( m \) of the corresponding context fragment, if the place node referred to by the literal contains no tokens in marking \( m \). For example, the negative literal \( \neg D_1 \) is consistent with any marking of context fragment \( c_1 \) in which place node \( D \) contains no token.

Using our approximation of the posterior it is possible to compute the marginal probability of some positive literal \( A \) with corresponding context fragment/factor \( c \) as follows:

\[
P(A) = \sum_{j=1}^{N} I_A(x_t^{(j)}) \cdot c_{w_t^{(j)} x_t^{(j)}}
\]

(9.23)

where

\[
I_A(x') = \begin{cases} 
1 & \text{if marking } x' \text{ is consistent with literal } A \\
0 & \text{otherwise}
\end{cases}
\]

(9.24)
Similarly the marginal probability of some negative literal \( \neg B \) with corresponding context fragment \( c \) is given by:

\[
P(\neg B) = \sum_{j=1}^{N} (1 - I_B(c^{(j)}_{x^{(l)}_m})) \cdot c^{(j)}_w \tag{9.25}
\]

using this definition for the marginal probability of literals, we can now compute the marginal probability of a conjunction of literals (e.g. \( C = A \land B \)):

\[
P(C) = \prod_{c \in C} P(c) \tag{9.26}
\]

where \( c \) is a literal in conjunction \( C \).

Similarly we compute the marginal probability of a disjunction (e.g. \( D = A \lor B \)):

\[
P(D) = 1 - \prod_{d \in D} (1 - P(d)) \tag{9.27}
\]

where \( d \) is a literal in disjunction \( D \).

Now if we are given a formula in conjunctive normal form (e.g. \( C = D_1 \land D_2 \land D_3 \), where each \( D_i \) is a disjunction) we can compute its marginal probability as follows:

\[
P(C) = \prod_{D_i \in C} \left( 1 - \prod_{d \in D_i} (1 - P(d)) \right) \tag{9.28}
\]

where \( d \) is a literal in disjunction \( D_i \).

### 9.3.4 The Context Update step

In this section we derive the update equations used for inference in section 9.3.2. The update step of the particle filter is analogous to the correction step of the Bayesian recursive filter. That is, we aim to correct our prediction in light of new information. This correction is achieved by dividing the estimation of the posterior, evaluated at the sampled particle values, by the proposal distribution used to approximate the posterior during sampling.

In the context portion of the inference procedure (Equation 9.20), we use the context dynamic model, \( P_c(x_{l} | x_{l-1}) \), to perform the prediction (i.e. sample particles). We should correct using the posterior evaluated at \( y_{l} \), given by

\[
P(y_{l} \mid x_{l})P_c(x_{l} \mid x_{l-1})
\]

Hence for each particle \( j \) the correction step is as follows:
\[ i_{w_{t}^{(j)}} = i_{w_{t-1}^{(j)}} \cdot \frac{P(i_{y_{t}^{(j)}}|i_{x_{t}^{(j)}})P_{c}(i_{x_{t}^{(j)}}|i_{x_{t-1}^{(j)}})}{P_{c}(i_{x_{t}^{(j)}}|i_{x_{t-1}^{(j)}})} = i_{w_{t-1}^{(j)}} \cdot P(i_{y_{t}^{(j)}}|i_{x_{t}^{(j)}}) \] (9.29)

### 9.3.5 Activity Update Step

Let us consider the “true” posterior over the activity state implied by the conditional independence relationships shown in Figure 9.5.

\[
P_{a}(i_{x_{t}}) = P(i_{x_{t}(m)}; i_{x_{t}(e)}) = \sum_{C_{t}} P(i_{x_{t}(m)}|i_{x_{t-1}(m)}; i_{x_{t-1}(e)}) \cdot P(i_{x_{t}(e)}|i_{x_{t}(m)}; C_{t}) \cdot P(C_{t})
\]

(9.30)

where the sum over \( C_{t} \) denotes summing over all possible conditions that can exist in the context fragment.

Although there are many possible such conditions we observe that, keeping \( i_{x_{t}} \) fixed:

\[
P(i_{x_{t}(e)}|i_{x_{t}(m)}; C_{t}) = \begin{cases} 
1 & C_{t} = C(i_{x_{t}(m)}; i_{x_{t}(e)}) \\
0 & \text{otherwise}
\end{cases}
\]

(9.31)

where \( C(i_{x_{t}(m)}; i_{x_{t}(e)}) \) denotes the condition that enables event set \( i_{x_{t}(e)} \) in marking \( i_{x_{t}(m)} \).

Using the above we can simplify:

\[
\sum_{C_{t}} P(i_{x_{t}(e)}|i_{x_{t}(m)}; C_{t}) \cdot P(C_{t}) = P(C_{t} = C(i_{x_{t}(m)}; i_{x_{t}(e)}))
\]

(9.32)

and thus the posterior (Equation 9.30) is reduced to:

\[
P_{a}(x_{t}) = P(i_{x_{t}(m)}|i_{x_{t-1}(m)}; i_{x_{t-1}(e)}) \cdot P(C_{t} = C(i_{x_{t}(m)}; i_{x_{t}(e)}))
\]

(9.33)

In the activity fragment update (Equation 9.22) \( i_{x_{t}(m)} \) and \( i_{x_{t}(e)} \) are known from the prediction(sampling) step, and \( P(C_{t} = C(i_{x_{t}(m)}; i_{x_{t}(e)})) \) is known by calculating the marginal over the context fragment using the particle approximation of the context posterior.

Thus in the correction phase we can use the following formula:
\[ \begin{align*}
  i_{w_t}^{(j)} &= i_{w_t-1}^{(j)} \cdot \frac{P(i \mid x_{t(m)}^{i}x_{t-1(m)}^{i}x_{t-1(e)}) \cdot P(C_t = C(i \mid x_{t(m)}^{i}x_{t(e)}))}{P(i \mid x_{t(m)}^{i}x_{t-1(m)}^{i}x_{t-1(e)})} \\
  &= i_{w_t-1}^{(j)} \cdot \frac{P(C_t = C(i \mid x_{t(m)}^{i}x_{t(e)}))}{P(x_{t(e)}^{i} \mid x_{t(m)}^{i})}
\end{align*} \] (9.34)

### 9.4 Example

In this section we will run through constructing an example activity model in the Bank surveillance domain. We shall start by defining the roles that participate in our relevant activities. In our experiments we are interested in two roles: Visitor and Cashier. Now we can construct the context fragments. In the bank setting, each object can occupy one of a number of disjoint zones. We will represent this knowledge of the context by creating the context fragment shown in Figure 9.6. All transition nodes within the context fragment are labeled with events that we may observe in our video sequence. Referring to the figure, In Entrance Zone, In FrontCounter Zone, etc. are all events which can be recognized up to some certainty by our event recognition module. One such fragment will be created for each of the context roles.

Now let us consider one of the activities we are interested in recognizing in the bank surveillance domain, namely the Bank Attack activity. In natural language we can describe this activity as follows: A visitor enters the bank as the cashier is behind the counter, the visitor goes behind the counter, the cashier and the visitor walk into the safe area at the same time. Translating this description in a set of temporal constraints, we have During(visitor in entrance, cashier in backcounter), Before(cashier in backcounter, cashier in safe), Before(visitor in backcounter, visitor in safe), GeneralOverlap(visitor in safe, cashier in safe). Clearly, these relations correspond to the relations we have previously discussed, so it is straightforward to construct the appropriate PN fragments for them. These fragments are shown in Figure 9.7.

Recall that we consider the right-most place node of each fragment in the figure to be the “source” place node and the left-most place node to be the “sink” place node. The source place node will contain a token when the system is initialized. The sink place will contain a token when the relation has been observed. When all relation fragments participating in the activity contain tokens in their recognized place node, the activity is recognized. The dynamics of the activity Petri Net fragments are straightforward, each transition node is labeled with a “context condition”. These conditions are conjunctions of literals. Each literal corresponds to a place node within the context fragment(s). A literal
becomes true when the corresponding place node contains a token (or does not if it is a negative literal). If a transition is enabled while the appropriate context condition becomes true, that transition is fired, modifying the token distribution throughout the Petri Net fragment.

### 9.4.1 Constructing the Bayes Recursive Filter Components

Let us consider the marking space of each fragment presented in the above example. The context fragment (Figure 9.6) will allow a token to be contained in only one of five places at any given time. Thus, using an initial marking with a single token, there are five reachable markings for this fragment. The activity fragments similarly have a small number of reachable markings, summarized in Table 9.1. Recall that $S_i$ denotes the set of all possible markings in Petri Net fragment $i$. For example, the state space of context fragment 1, $S_{c_1} = \{(c_1)\sigma_1, (c_1)\sigma_2, (c_1)\sigma_3, (c_1)\sigma_4, (c_1)\sigma_5\}$. $(c_1)\sigma_1$ denotes the marking in which place node $A_1$ in the context fragment contains a token.

$O_{c_1}$, denotes the set of relevant events to context fragment $c_1$, and is determined from the unique transition labels in the PN fragment definition. The fragment depicted in Figure 9.6 implies the following:

$$O_{c_1} = \{\text{Cashier In Entrance Zone, Cashier In FrontCounter Zone, Cashier In Safe Zone, Cashier In BackCounter Zone, Cashier Disappear}\}$$

Recall, that we create an instance of the context fragment for each activity role. Thus in our example, the set $O_{c_2}$ refers to events involving the Cashier object. A second set $O_{c_2}$ refers to events involving the Visitor object.

Set $M_i$, represents all relevant combinations of events in set $O_i$ that may occur. Clearly, $M_i \subset 2^O$. This set contains combinations of events that may
occur simultaneously according to the definition of fragment $i$. In examining the
context fragment in Figure 9.6, we note that all transitions are in conflict with
one another. That is no two transitions may fire at the same time (the firing of
one would disable the firing of the other). Thus,

\[ M_{c_1} = \{\emptyset, \{\text{Cashier In Entrance Zone}\}, \{\text{Cashier In FrontCounter Zone}\}, \{\text{Cashier In Safe Zone}\}, \{\text{Cashier In BackCounter Zone}\}, \{\text{Cashier Disappear}\}\} \]

Let us now discuss the construction of the elements of the BRF for the con-
text fragment depicted in Figure 9.6, $c_1$, whose initial state is $(c_1)\sigma_1$. The prior
distribution is straightforward to construct given our initial state. We can define
the following simple prior probability distribution over the marking space $S_{c_1}$:

\[
P((c_1)x_{0(m)}) = \begin{cases} 
1 & \text{if } (c_1)x_{0(m)} = (c_1)\sigma_1 \\
0 & \text{otherwise}
\end{cases}
\]

Let us now illustrate the construction of the context dynamic model corre-
sponding to fragment $c_1$. Recall that, $i^e_{x_{t-1}(m)}$ denotes the marking of PN frag-
ment $i$ in the previous frame , $i^e_{x_{t-1}(e)}$ denotes the event set that occurred in the
previous frame, and $i^e_{x_{t}(m)}$ denotes the resulting marking of fragment $i$.

The first component of the context dynamic model is

\[ P((c_1)x_{t(m)}|(c_1)x_{t-1(m)}, (c_1)x_{t-1(e)}) \]

which describes the Petri Net fragment dynamics.

Let us examine the case of $(c_1)x_{t-1(m)} = (c_1)\sigma_2$, that is, in the previous
time frame place node $B_1$ contains a token. By examining Figure 9.6 we can see that if event
Cashier In FrontCounter Zone occurs (i.e. $(c_1)x_{t-1(e)} = \{\text{Cashier In FrontCounter Zone}\}$) the resulting marking would be one where
place $C_1$ contains a token $(c_1)x_{t(m)} = (c_1)\sigma_3$. These dynamics are captured as a
discrete probability distribution as follows:

\[
P((c_1)x_{t(m)}|(c_1)x_{t-1(m)}, (c_1)x_{t-1(e)}) = \begin{cases} 
1 & \text{if } (c_1)x_{t(m)} = (c_1)\sigma_3 \\
0 & \text{otherwise}
\end{cases} \quad (9.35)
\]

We use the same process to define the remainder of

\[ P((c_1)x_{t(m)}|(c_1)x_{t-1(m)}, (c_1)x_{t-1(e)}): \]

\[
\begin{array}{cccccccc}
(c_1)x_{t(m)} & (c_1)x_{t-1(m)} & (c_1)x_{t-1(e)} \\
(c_1)\sigma_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
(c_1)\sigma_2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
(c_1)\sigma_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
(c_1)\sigma_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
(c_1)\sigma_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\end{array}
\]

121
The second component of the context dynamic model, \( P^{(c_1)x_t(e)}(c_1)x_t(m) \), defines a distribution over the relevant event sets, \( M_{c_1} \), that can occur at frame \( t \). Again consider the case of \( (c_1)x_t(m) = (c_1)\sigma_2 \). Examining Figure 9.6 we see that four transitions, respectively labeled with events Cashier Disappear, Cashier In BackCounter Zone, Cashier In FrontCounter Zone, Cashier In Safe Zone are enabled in this marking. These correspond to the possible event sets when fragment \( c_1 \) takes on marking \( (c_1)\sigma_2 \). Recall also that the empty event set, \( \emptyset \), is enabled in all markings.

\[
\hat{P}^{(c_1)x_t(e)}(c_1)x_t(m) = (c_1)\sigma_2 = \begin{cases} 1 & \text{if } (c_1)x_t(e) = \emptyset \text{ or } \{ \text{Cashier Disappear} \} \text{ or } \{ \text{Cashier In BackCounter Zone} \} \\ 0 & \text{otherwise} \end{cases}
\]

After the normalization step we get:

\[
P^{(1)x_t(e)}x_t(m) = (c_1)\sigma_2 = \begin{cases} \frac{1}{5} & \text{if } (c_1)x_t(e) = \emptyset \text{ or } \{ \text{Cashier Disappear} \} \text{ or } \{ \text{Cashier In BackCounter Zone} \} \\ 0 & \text{otherwise} \end{cases}
\]

The remainder of \( P^{(c_1)x_t(e)}(c_1)x_t(m) \) is calculated similarly:

\[
P^{(c_1)x_t(e)}(c_1)x_t(m) = \begin{array}{|c|c|c|c|c|c|}
\hline
(c_1)x_t(e) / (c_1)x_t(m) & x_1 & x_2 & x_3 & x_4 & x_5 \\
\hline
\emptyset & 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\
\{ \text{Cashier Disappear} \} & 0 & 1/5 & 1/5 & 1/5 & 1/5 \\
\{ \text{Cashier In BackCounter Zone} \} & 1/5 & 1/5 & 1/5 & 1/5 & 0 \\
\{ \text{Cashier In FrontCounter Zone} \} & 1/5 & 1/5 & 0 & 1/5 & 1/5 \\
\{ \text{Cashier In Safe Zone} \} & 1/5 & 1/5 & 1/5 & 0 & 1/5 \\
\{ \text{Cashier In Entrance Zone} \} & 1/5 & 0 & 1/5 & 1/5 & 1/5 \\
\hline
\end{array}
\]

Once the context dynamic model is constructed, it remains to construct the context measurement model. Recall that for each event, \( O_i(j), j = 1 \ldots |O_i| \), we construct a separate model, \( P^{(1)y_t(j)}x_t(e) \).

Again let us consider the context fragment \( c_1 \), depicted in Figure 9.6. Suppose the first event in an arbitrary ordering of the elements of \( O_{c_1} \) is Cashier In Entrance Zone, so that \( O_{c_1}(1) = \text{Cashier In Entrance Zone} \) and \( (c_1)y_t(1) \) is the certainty with which Cashier In Entrance Zone is observed at frame \( t \). Since, of all the sets in \( M_{c_1} \), \( O_{c_1}(1) = \text{Cashier In Entrance Zone} \) is only a member of one of the sets (i.e \( \{ \text{Cashier In Entrance Zone} \} \)) we set

\[
P^{(c_1)y_t(1)}(c_1)x_t(e) = \begin{cases} \varphi^{(c_1)y_t(1)} & \text{if } (c_1)x_t(e) = \{ \text{Cashier In Entrance Zone} \} \\ 1 - \varphi^{(c_1)y_t(1)} & \text{otherwise} \end{cases}
\]

where \( \varphi(\cdot) \) is the sigmoid function defined in Equation 9.13.

Similarly, the remainder of the measurement model is constructed:

\[
P^{(c_1)y_t(j)}(c_1)x_t(e):\]

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
O_{c_1}(j) / (c_1)x_t(e) & \emptyset & \{ \text{Cashier In Entrance Zone} \} & \{ \text{Cashier In BackCounter Zone} \} & \{ \text{Cashier In FrontCounter Zone} \} \\
\hline
\{ \text{Cashier In Entrance Zone} \} & 1 - \varphi^{(c_1)y_t(1)} & \varphi^{(c_1)y_t(1)} & 0 \\
\{ \text{Cashier In BackCounter Zone} \} & 1 - \varphi^{(c_1)y_t(1)} & 1 - \varphi^{(c_1)y_t(1)} & 0 \\
\{ \text{Cashier In FrontCounter Zone} \} & 1 - \varphi^{(c_1)y_t(1)} & 0 & 0 \\
\{ \text{Cashier In Safe Zone} \} & 1 - \varphi^{(c_1)y_t(1)} & 1 - \varphi^{(c_1)y_t(1)} & 0 \\
\{ \text{Cashier In Entrance Zone} \} & 1 - \varphi^{(c_1)y_t(1)} & 1 - \varphi^{(c_1)y_t(1)} & 0 \\
\hline
\end{array}
\]
9.4.2 Inference

Now let us use the above to make the particle filter mechanism more concrete. We shall use an example with \( N = 3 \) particles. These particles are factored over the number of PN fragments in our example.

Before beginning our inference process we have to initialize our particle set for each factor of our model. Recall from section 9.3 that this is achieved by first sampling the marking, \( ^i x_{0(m)} \), from \( P( ^i x_{0(m)} ) \) and then using the sampled value, sample the event combination, \( ^i x_{0(e)} \), from \( P( ^i x_{0(e)} | ^i x_{0(m)} ) \).

Let us again consider the context fragment \( c_1 \) shown in Figure 9.6. The marking component of the three particles in our example is initialized by sampling from \( P( (c_1) x_{0(m)} ) \). Since by the construction of this distribution all probability mass is concentrated on the initial marking \( (c_1) \sigma_1 \), this marking will be the outcome of these samplings. Now we sample the event combination of each particle using \( P( (c_1) x_{0(e)} | (c_1) x_{0(m)} ) \). Examining the construction of this probability distribution above, we can see that each of the event combinations: \( \emptyset \), \{Cashier In Entrance Zone\}, \{Cashier In FrontCounter Zone\}, \{Cashier In Safe Zone\}, and \{Cashier In BackCounter Zone\} is given equal probability (1/5). Thus, one likely result of our initialization would be the following particle assignments:

\[
\begin{align*}
(c_1) x_0^{(1)} &= \{(c_1) \sigma_1, \{\text{Cashier In BackCounter Zone}\}\}, \\
(c_1) x_0^{(2)} &= \{(c_1) \sigma_1, \emptyset\}, \\
(c_1) x_0^{(3)} &= \{(c_1) \sigma_1, \{\text{Cashier In FrontCounter Zone}\}\}
\end{align*}
\]

where \( (c_1) x_0^{(1)} = \{(c_1) \sigma_1, \{\text{Cashier In BackCounter Zone}\}\} \) is shorthand for \( (c_1) x_{0(m)} = (c_1) \sigma_1 \), \( (c_1) x_{0(e)}^{(1)} = \{\text{Cashier In BackCounter Zone}\} \).

We also initialize the weights to \( 1/N \) (recall \( N = 3 \) in our example)

\[
(c_1) w_0^{(1)} = 1/3, \quad (c_1) w_0^{(2)} = 1/3, \quad (c_1) w_0^{(3)} = 1/3
\]

We follow the same procedure to initialize all other context and activity factors. For example one possible initialization for the particles pertaining to activity factor \( a_1 \), which we shall refer to later in this example is given by:

\[
\begin{align*}
(a_1) x_0^{(1)} &= \{(a_1) \sigma_1, \emptyset\}, \quad (a_1) x_0^{(2)} = \{(a_1) \sigma_1, \{\text{trans}_1\}\}, \quad (a_1) x_0^{(3)} = \{(a_1) \sigma_1, \{\text{trans}_1\}\}
\end{align*}
\]

After the initialization is complete we can evaluate new observations in an online fashion using our particle filter framework. For the sake of this example let us focus on how this is achieved in fragments \( c_1 \) and \( a_1 \). Let us suppose our event recognition component for the first frame of the video considers the
event \{Cashier in BackCounter Zone\} to have occurred with 0.8 certainty, and no other events to have occurred. Thus the observation vector \((c_1)\{y_1\}\) will have 5 components \((O_{c_1})\) has 5 members), all of which will have a 0 certainty, save for component \(k\) which represents the event Cashier In BackCounter Zone (i.e. \((c_1) y_{1(k)} = 0.8\)).

Continuing our example, since \((c_1) x_1^{(1)} = \{c_1, \{\text{Cashier In BackCounter Zone}\}\}:

\[
P((c_1) y_{1(j)} | (c_1) x_1^{(1)}) = \begin{cases} 
\varphi((c_1) y_{1(j)}) & \text{if } j = k \\
1 - \varphi((c_1) y_{1(j)}) & \text{otherwise}
\end{cases}
\]

where \(k\) represents the index of event \{Cashier In BackCounter Zone\}. Thus since \((c_1) y_{1(j)} = 0\) for all \(j \neq k\) and \((c_1) y_{1(k)} = 0.8\)

\[
P((c_1) y_{1(j)} | (c_1) x_1^{(1)}) = \prod_{j=1}^{\mid O_{c_1} \mid = 5} P((c_1) y_{1(j)} | (c_1) x_1^{(1)}) = \varphi(0.8) \cdot (1 - \varphi(0))^4
\]

Updating the weights is then done according to:

\[
(c_1) w_1^{(1)} = (c_1) w_0^{(1)} \cdot P((c_1) y_{1} | (c_1) x_1^{(1)}) = (c_1) w_0^{(1)} \cdot \varphi(0.8) \cdot (1 - \varphi(0))^4 = 0.3091
\]

where we are using

\[
\varphi(z) = \frac{1}{1 + e^{-h \cdot (z - 0.5)}}
\]

with \(h = 10\).

We adjust the other particle weights similarly:

\[
(c_1) w_1^{(2)} = (c_1) w_0^{(2)} \cdot P((c_1) y_{1} | (c_1) x_1^{(2)}) = (c_1) w_0^{(2)} \cdot (1 - \varphi(0.8)) \cdot (1 - \varphi(0))^4 = 0.0154
\]

\[
(c_1) w_1^{(3)} = (c_1) w_0^{(3)} \cdot P((c_1) y_{1} | (c_1) x_1^{(3)}) = (c_1) w_0^{(3)} \cdot (1 - \varphi(0.8)) \cdot (1 - \varphi(0))^2 \cdot \varphi(0) = 1.03 e^{-4}
\]

Our final step is to normalize our weights:

\[
\sum_{i=1}^{N=3} (c_1) w_1^{(i)} = 0.3246
\]

\[
(c_1) w_1^{(1)} = \frac{(c_1) w_0^{(1)}}{0.3246} = 0.9522
\]

\[
(c_1) w_1^{(2)} = \frac{(c_1) w_0^{(2)}}{0.3246} = 0.0474
\]

\[
(c_1) w_1^{(3)} = \frac{(c_1) w_0^{(3)}}{0.3246} = 0.0001
\]
After the correction step for the context fragment above is complete, we perform the prediction step. That is we use the current context state estimation to sample from the context proposal distribution, $P_c$, to derive an updated estimation of the context state.

The prediction for fragment $c_1$ is done in two phases: initially we sample from $P_c((c_1)x_{t(m)}| (c_1)x_{t-1(m)}, (c_1)x_{t-1(e)})$, and then, using the sampled result, we sample from $P((c_1)x_{t(e)}| (c_1)x_{t(m)})$.

Continuing our example, the marking component of particle $1$, $(c_1)x_{1(m)}^{(1)} \sim P((c_1)x_{t(m)}| (c_1)x_{t-1(e)}) = (c_1)\sigma_1$, $(c_1)x_{t-1(e)} = \{\text{Cashier In Back Counter Zone}\}$ would likely result in $(c_1)x_{1(m)}^{(1)} = (c_1)\sigma_5$ (the only outcome with non-zero probability). Then, $(c_1)x_{1(e)}^{(1)} \sim P((c_1)x_{t(e)}| (c_1)x_{t(m)} = (c_1)\sigma_1)$ could result in any one of five outcomes, each with equal probability ($\emptyset, \{\text{Cashier In FrontCounter Zone}\}, \{\text{Cashier In Backcounter Zone}\}, \{\text{Cashier In Safe Zone}\}, \{\text{Cashier Disappear}\}$).

Similarly the other particle markings are given by: $(c_1)x_{1(m)}^{(2)} = (c_1)\sigma_1$, and $(c_1)x_{1(m)}^{(3)} = (c_1)\sigma_4$. Thus, let us assume one of several possible result of such a sampling:

$$(c_1)x_1^{(1)} = \{c_1\sigma_5, \emptyset\}, (c_1)x_1^{(2)} = \{c_1\sigma_1, \{\text{In Safe Zone}\}\}, (c_1)x_1^{(3)} = \{c_1\sigma_4, \{\text{In BackCounter Zone}\}\}$$

After the update and prediction of the context fragments, we perform the update and prediction of the activity fragments. The update step of the activity fragments differs slightly from the update step of the context fragments, in that we update and prediction of the activity fragments. The update step of the activity fragments is the only context with label $t_{x_{t(m)}}$, to derive an updated estimation of the activity condition.

For example, consider particle $(a_1)x_0^{(1)}$ (initialized earlier). The marking variable $(a_1)x_{0(m)} = (a_1)\sigma_1$ and the event set variable $(a_1)x_{0(e)} = \{\text{trans}_1\}$, therefore we are interested in $P(C((a_1)\sigma_1, \{\text{trans}_1\}) = P(\text{"E}_1 \land \neg D_1")$. Recall that $C(t_{x_{t(m)}}, t_{x_{t(e)}})$ is our notation for referring to the context condition given to a transition with label $t_{x_{t(m)}}$ enabled in marking $t_{x_{t(m)}}$.

This quantity is given by the following formula (see section 9.3.3):

$$P(\text{"E}_1 \land \neg D_1") = P(E_1) \cdot (1 - P(D_1))$$

where $P(E_1)$ is the probability place node $E_1$ in context fragment $c_1$ contains a token, and $P(D_1)$ is the probability that place node $D_1$ in context fragment $c_1$ contains a token.

Examining Table 9.1 we see that marking $(a_1)\sigma_4$ is the only context marking where place node $D_1$ contains a token. Furthermore, marking $(a_1)\sigma_5$ is the only marking where place node $E_1$ contains a token.
using the formula:

\[ P(E_1) = \sum_{j=1}^{N} (I_{E_1}[c_1 x_0^{(j)}]) \cdot (c_1) w_0^{(j)} = 1 \cdot (c_1) w_0^{(1)} + 0 \cdot (c_1) w_0^{(2)} + 0 \cdot (c_1) w_0^{(3)} = (c_1) w_0^{(1)} = .9522 \]

This is due to the fact that \( I_{E_1}[c_1 x_0^{(j)}] = 0 \) for all \( j \neq 1 \) (i.e. place \( E_1 \) does not contain a token in these markings).

Similarly:

\[ P(D_1) = \sum_{j=1}^{N} (I_{D_1}[c_1 x_0^{(j)}]) \cdot (c_1) w_0^{(j)} = 0 \cdot (c_1) w_0^{(1)} + 0 \cdot (c_1) w_0^{(2)} + 1 \cdot (c_1) w_0^{(3)} = (c_1) w_0^{(3)} = .0001 \]

Thus:

\[ P(\text{"E}_1 \land \neg D_1\text{"}) = P(E_1) \cdot (1 - P(D_1)) = .9522 \cdot (1 - .0001) = .9521 \]

We use this value to update our activity weight:

\[ (a_1) \hat{w}_1^{(1)} = (a_1) w_0^{(1)} \cdot \frac{P(\text{"E}_1 \land \neg D_1\text{"})}{P(a_1 x_0^{(1)} | a_1 x_0^{(1)})} = 1/3 \cdot \frac{.9521}{1/5} = 1.587 \]

The remaining particle weights are updated in the same manner:

\[ (a_1) \hat{w}_1^{(2)} = (a_1) w_0^{(2)} \cdot \frac{1 - P(\text{"E}_1 \land \neg D_1\text{"})}{P(a_1 x_0^{(2)} | a_1 x_0^{(2)})} = 1/3 \cdot \frac{.0479}{1/5} = .08 \]

\[ (a_1) \hat{w}_1^{(3)} = (a_1) w_0^{(3)} \cdot \frac{1 - P(\text{"E}_1 \land \neg D_1\text{"})}{P(a_1 x_0^{(3)} | a_1 x_0^{(3)})} = 1/3 \cdot \frac{.0479}{1/5} = .08 \]

We then normalize in the same manner as above

\[ \sum_i (a_1) w_1^{(i)} = 1.683 \]

\[ (a_1) w_1^{(1)} = \frac{(a_1) \hat{w}_1^{(1)}}{1.683} = .9430 \]

\[ (a_1) w_1^{(2)} = \frac{(a_1) \hat{w}_1^{(2)}}{1.683} = .0284 \]

\[ (a_1) w_1^{(3)} = \frac{(a_1) \hat{w}_1^{(3)}}{1.683} = .0284 \]
After the update step the prediction step of the activity fragment proceeds similar to the prediction step of the context fragment. For completion we will carry it this step in our example.

Continuing our example, particle \((a_1)x_{1(\pi)}^{(1)} \sim P((a_1)x_{1(\pi)}^{(1)}) = (a_1)\sigma_1, (a_1)x_{0(\varepsilon)} = \emptyset)\) would likely result in \((a_1)x_{1(\pi)}^{(1)} = (a_1)\sigma_1\) (the only outcome with non-zero probability). Note that in marking \((a_1)\sigma_1\), \(trans_1\) is the only enabled transition. Thus, \(x_{1(\pi)}^{(1)}(e)(1) \sim P(x_{1(\pi)}^{(1)}(e)| x_{1(\pi)}^{(1)}(\pi) = (a_1)\sigma_1)\) could result in one of two outcomes, each with equal probability \((\emptyset, \{trans_1\})\).

Similarly the other particle markings are given by: \((a_1)x_{1(\pi)}^{(2)} = (a_1)\sigma_2\), and \((a_1)x_{1(\pi)}^{(3)} = (a_1)\sigma_2\)

Thus, a plausible result of the prediction step yields the following particle values:

\[
\begin{align*}
(a_1)x_1^{(1)} &= \{(a_1)\sigma_1, \{trans_1\}\}, \quad (a_1)x_1^{(2)} = \{(a_1)\sigma_2, \{trans_2\}\}, \quad (a_1)x_1^{(3)} = \{(a_1)\sigma_2, \emptyset\}
\end{align*}
\]
Figure 9.1: The template fragments used for the various temporal relations.
Figure 9.2: An example of chaining relations. Here we use the markings of the nested inner relation PN fragment as context for the PN relation fragment of the outer relation.
Figure 9.3: The conditional independence relations between the context, activity and observation variables. The activity state is independent of the observation given the context.

Figure 9.4: The conditional independence relations between the context and observation variables. The observation, $i\mathbf{y}_t$, is independent of the context factor marking, $i\mathbf{x}_{t(m)}$, given the relevant events at time $t$, $i\mathbf{x}_{t(e)}$. 
Figure 9.5: The conditional independence relations between the context and activity variables. The context at time $t$ (denoted $C_t$) is a factor in determining which events occur at time $t$ in each activity factor.

Figure 9.6: Context Fragment for the Bank Surveillance Dataset
Figure 9.7: Temporal Relation Fragment. Transition label letters correspond to place nodes in context fragment (Figure 9.6). Transition Label sub-indexes correspond to the object instantiation (1-cashier, 2-visitor).
Chapter 10

Evaluating Probabilistic Models For Event Recognition

In our experiments we endeavored to provide a rigorous empirical comparison of our activity recognition approach to other leading approaches. In order to make the comparison fair we used the same video processing and event recognition system as input to all approaches. We evaluated the different approaches over three datasets. As in the previous two chapters, we are using the term *activity* to mean multi-thread complex event, and the term *event* to mean an atomic sub-event of the activity restricted to a temporal interval (with respect to the terminology defined in Chapter 2).

Figure 10.1: Flowchart of System
10.1 System Description

In order to arrive at an activity classification of an unlabeled video sequence we constructed an experimental framework with several components (See Figure 10.1). The raw image data (Input #1) is fed to Module #1 (Object Detection and Tracking) to generate a list of objects and their corresponding locations at each frame. In our experiments, we used the ground truth locations on the synthetic bank dataset, a particle filter tracker (Isard and Blake, 1998) on the ETISEO dataset, and a tracker based on boosting (Babenko, Yang, and Belongie, 2009) for the Technion dataset. We used a semi-automatic approach to tracking, where if an object track had been lost by the tracker, the tracker was again reinitialized using the ground truth. The result of this process is a mostly accurate but at times noisy tracker output. The Technion dataset was also evaluated using “ideal” tracking conditions, to allow evaluation of the activity recognition without the influence of observation uncertainty.

The object locations are input into Module #2 (Event Recognition). Specification of scene information (Input #2), such as zone definitions and camera calibration information, is also input into Module #2. We chose to implement a simple rule based event recognition similar to the one adopted in (Vu, Bremond, and Thonnat, 2003). In our experiments we recognize several kinds of events including: Appearance, Disappearance, In Zone, Moving, Stopped, Near (2 objects), Far (2 objects). We make use of the scene specification to recognize these events and associate the appropriate certainty value. Figure 10.2 show a keyframe from our datasets annotated with the output of Modules #1 and #2.

The list of events and their corresponding certainty values that occur at each frame output by Module #2 are input into Module #3 (Activity Recognition). The activity model specifications (which may be Petri Nets, Propagation Nets, Temporal Constraint Logic Models) (input #3) are also input into Module #3. This module outputs a list of recognized activities in the video sequence. In our experiments we implemented three different types of activity recognition approaches. The Store Totally Recognized Scenarios (STRS) approach is implemented based on (Vu, Bremond, and Thonnat, 2003) and represents deterministic approaches to activity recognition. In this approach, event inputs were thresholded and only those whose certainties were above a threshold were passed into the activity recognition module. The Propagation Net (P-Net) approach is implemented based on (Shi et al., 2004) and represents competing probabilistic approaches to activity recognition. Note that the P-Net approach relies on an available duration model for each of the events in the activity. In general such a model is not available and an uninformative duration model was applied. The exception to this was the Synthetic Bank dataset, which allowed us to provide a distribution over each event duration. The Particle Filter Petri Net (PFPN) is
implemented using the methods explained in detail in Chapter 8. The Factored Particle Filter Petri Net (FPFPN) is implemented using the methods discussed in Chapter 9. We used $N = 2000$ particles in our experiments. The value of this parameter was determined empirically to provide the best tradeoff between performance and running time.

### 10.2 Metrics for Performance

In order to give performance results that allow for comparison of the various activity recognition approaches, we must determine when an activity has been recognized. For those approaches that output a certainty associated with each recognition (P-Net, PPFPN and FPFPN) we simply threshold this value. In all of our experiments an activity is considered to be recognized if its associated certainty is above some threshold $\theta$. Since at each frame we may have a different activity recognition certainty, we simply threshold the maximum certainty for this activity over the length of the video sequence.

Comparing the activity recognition output to the available ground truth allows us to compute the number of true positives, true negatives, false positives, and false negatives. Since each clip has only a few activities occurring in it, the number of positive examples is significantly smaller than the number of negative examples. This is also the case in real surveillance systems. For this reason an approach may achieve a fairly high accuracy score by classifying all examples as negative. Thus when evaluating each approach, it is important to take into account the tradeoff between true and false positive recognitions. To this end, we have chosen to present the results as an ROC curve, which plots the true positive rate (also known as recall) against the false positive rate. These metrics are given by the formulas:

$$
\text{true positive rate} = \frac{tp}{(tp + fn)} \quad (10.1)
$$
false positive rate = \frac{fp}{(fp + tn)} \quad (10.2)

Clearly, as we decrease our recognition threshold, \( \theta \), from 1.0 to 0 we will achieve a higher number of both true and false positives, causing both above rates to rise. An ideal threshold selection would achieve a high true positive rate and a low false positive rate. Generally, a tradeoff exists between these rates and we must choose a threshold to favor one or the other. The ROC curve presentation illustrates this tradeoff for a number of values of \( \theta \), for each of the algorithms we have evaluated in this chapter. We have also computed the Area Under the Curve (AUC) which quantifies this tradeoff as a real number.

It should be clarified that the STRS algorithm is not a stochastic algorithm, thus its decision is binary and cannot be thresholded. To compile the curve for this algorithm we instead varied the threshold for event recognition, that is, the certainty threshold above which an event will be considered recognized.

### 10.3 Metrics for Complexity

In particle filter approaches the temporal complexity is a function of the number of particles used. The number of particles must be increased as the state space becomes more complex. Thus a decrease in the complexity of the state space can be translated into an improvement in efficiency. We have chosen an additional metric to quantify this improvement. We plot the recognition rate (expressed as an f-score) of the various approaches against the number of particles used. A recognition rate which remains steady as the number of particles decreases implies that using less particles (i.e. computational resources) would yield similar performance results. Hence such a method can be said to be able to scale better as activities and contexts become more complex. Specifically, the f-score is computed from the precision and recall as follows:

\[
f\text{-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\]

### 10.4 Datasets

Unfortunately, a surveillance video dataset with annotated activities is time consuming to compile. Recent public datasets pertaining to surveillance video, such as VIRAT (Oh et al., 2011), emphasize event recognition and do not include repeated non-trivial activities such as the ones we consider in this work. Existing activity datasets have only a few examples of each activity. Thus, in addition to using a publicly available dataset, we captured and annotated our own dataset.
In order to allow evaluation of our approach on a large number of examples we created synthetic animated video clips. These clips can be created at considerably less cost than real data and serve to further illustrate the effectiveness of our approach. Example activity clips from the datasets used in our experiments can be viewed at (EXA, 2012).

### 10.4.1 Synthetic Bank Dataset

In the first set of experiments we considered 500 short synthetic video clips lasting from 274-526 frames (9-17 seconds) each. Based on (Albanese et al., 2008a) we considered the activities (1) Bank Attack (2) Attempted Bank Attack (3) Normal...
Customer Interaction (4) Cashier accesses safe (5) Outsider enters safe. Most of the clips contain one or more of these activities. An example sequence from this dataset is shown in Figure 10.3.

10.4.2 ETISEO Building Entrance Dataset

In order to evaluate our approaches on an existing, publicly available set of video sequences, we chose the ETISEO Building Entrance dataset (Nghiem et al., 2007). This dataset is a publicly available dataset of real videos which includes multiple camera views of the scene and includes several non-trivial activities. The ground truth of object locations across the different camera locations is available for download. The ground truth activity labeling was manually annotated by the authors. This dataset contains 6 sequences from up to 4 camera angles (though not all sequences contain data for all cameras angles). We defined 6 activities that can take place in this domain: (1) Arrive By Car, (2) Arrive On Foot, (3) Depart by Car, (4) Depart on Foot, (5) Meet and Walk Together, and (6) Meet and Walk Apart. Note that different scene objects may be involved in one or more activities.

The various camera angles provide an additional challenge to the event recognition module: how to integrate event recognitions from different camera angles. In our experiments we chose a simple approach of merging events from all camera angles into a single list before inputting this list into the activity recognition module. In the case where the same event is detected in multiple camera angles, we choose the event with the highest certainty value. The sequences in this dataset ranged from 924 to 1649 frames in (30-54 seconds) in length. Each sequence contained one or more of the activities. An example sequence from this dataset is shown in Figures 10.4.

10.4.3 Technion Parking Lot Dataset

The Technion Parking Lot dataset was captured and manually annotated by the authors. This dataset is intended to expand the breadth of our experiments on real data. We captured over two hours of surveillance footage with multiple objects and several activities. Several staged activities were acted out in order to test our system’s ability to recognize rarely occurring activities. These include: (1) Car Theft- a person approaches several vehicles before finally entering a final vehicle and driving off, (2) Break In- A person enters a car and then exits without driving away, (3) Arrive by car, (4) Depart by Car, and (5) Drop Off events. For the latter four activities our dataset contains both staged and real occurrences of the activities. In this dataset there were a total of 24 activities ranging from 43 to 207 seconds in length, with a median length of 100 seconds. Each activity includes at least two and up to four objects. An example sequence from this
dataset is shown in Figures 10.5.

10.5 Results and Discussion

Figures 10.6, 10.10 shows the quantitative results obtained on the synthetic Bank Dataset with and without the FPFPN approach. Note that for these experiments we applied the P-Net approach twice. In the first run a duration model for each of the events in the activity was used. In the second run an uninformative duration model was used.

Figures 10.7, 10.11 shows the quantitative results obtained on the ETISEO Building Entrance Dataset.

The technion dataset proved to be the most challenging and all approaches yielded lower recognition results when applied to it. Figure 10.8, 10.12 show the ROC curve obtained on the Technion Parking Lot Dataset. Figure 10.9, 10.13 show the results on this same dataset under “ideal” tracking conditions (i.e. the tracking information was taken directly from ground truth annotation of object locations.)

Tables and summarize the performance results in terms of area under the curve to allow a numerical comparison.

Figures 10.14 – 10.16 show a plot of the f-score of each method as the number of particles is varied. The f-score represents the harmonic average of the precision and recall scores for the best threshold value.

In examining Figures 10.6 - 10.9 as well as Table 10.1 we can make several observations regarding our experiments. The first observation is that the lowest false positive rates are achieved by the deterministic STRS approach. In the bank dataset, this approach achieves the highest AUC, and achieves a comparable true positive rate with all other approaches. However, in the other experiments,
**Figure 10.7:** ROC Curve for the ETISEO Building Entrance Dataset

**Figure 10.8:** ROC Curve for the Technion Parking Lot Dataset (real tracking)

**Figure 10.9:** ROC Curve for the Technion Parking Lot Dataset ("ideal" tracking)
Table 10.1: Area Under The Curve

<table>
<thead>
<tr>
<th>Method</th>
<th>Bank</th>
<th>ETISEO</th>
<th>Technion(Real)</th>
<th>Technion(Perfect)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle Filter PN</td>
<td>0.92285</td>
<td>0.82672</td>
<td>0.80766</td>
<td>0.91036</td>
</tr>
<tr>
<td>Propagation Net w/ Duration Model</td>
<td>0.93274</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Propagation Net w/o Duration Model</td>
<td>0.71401</td>
<td>0.7672</td>
<td>0.76389</td>
<td>0.90007</td>
</tr>
<tr>
<td>Constraint Propagation</td>
<td>0.93858</td>
<td>0.75995</td>
<td>0.72278</td>
<td>0.84328</td>
</tr>
</tbody>
</table>

Figure 10.10: ROC Curve Comparing Recognition Performance for the Bank Dataset including FPFPN

Figure 10.11: ROC Curve Comparing Recognition Performance for the ETISEO Dataset including FPFPN

Table 10.2: Area Under The Curve including FPFPN

<table>
<thead>
<tr>
<th>Method</th>
<th>Bank</th>
<th>ETISEO</th>
<th>Technion(Real)</th>
<th>Technion(Perfect)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factored Particle Filter PN</td>
<td>0.9991</td>
<td>0.9166</td>
<td>0.8701</td>
<td>0.9355</td>
</tr>
<tr>
<td>Particle Filter PN</td>
<td>0.92285</td>
<td>0.82672</td>
<td>0.80766</td>
<td>0.91036</td>
</tr>
<tr>
<td>Propagation Net</td>
<td>0.71401</td>
<td>0.7672</td>
<td>0.76389</td>
<td>0.90007</td>
</tr>
<tr>
<td>Constraint Propagation</td>
<td>0.93858</td>
<td>0.75995</td>
<td>0.72278</td>
<td>0.84328</td>
</tr>
</tbody>
</table>
where the observation and semantic uncertainty grows, the relative performance of this approach degrades in comparison to the stochastic approaches. The reason behind this phenomenon is that deterministic approaches such as STRS must observe every event component of an activity in order for the activity to be recognized. If one of these event components is observed with low certainty (below the threshold), or not observed at all, the deterministic approach will not be able to recognize the activity and will record a false negative. Stochastic approaches that are able to reason on uncertain input, such as PFPN, have a mechanism to “fill in the gaps” which allows the recognition of an activity, in spite of this missing information. Informally, this is done by predicting all the events
that may happen, and then validating these predictions with future observations. The downside of this approach is that occasionally event ”hallucinations” will occur inappropriately, causing false recognitions of activities. In other words, the stochastic approaches tends to err on the side of false positives. Clearly, this result implies a tradeoff. If a particular application requires a near zero false positive rate, a deterministic approach to activity recognition is a better fit for this application. In the inverse case where a false negative is more costly than a false positive (e.g. sending a policeman to the bank as opposed to letting the bank be robbed), a stochastic approach such as PFPN is preferable. Among the stochastic methods examined in Figures 10.6 - 10.9, PFPN achieves a lower
false positive rate while achieving higher or roughly equivalent true positive rate. The PFPN approach also achieves a higher AUC score than the P-Net model in all but the bank dataset experiment. It should be noted that in the bank dataset experiment the P-Net model has more information at its disposal (i.e. the duration model). We explain this improvement by noting that the PFPN framework allows construction of a holistic activity model that models events in both the activity and non-activity context. In contrast, the P-Net only models the activity, not taking into account events that might occur outside the activity context. Recall that many activities in the surveillance domain have similar event composition, and sometimes differ only in the temporal configuration of the component events. In other words, events often occur outside the context of the activity being modeled (as part of a competing activity or as a result of image processing errors). If this type of occurrence is not appropriately modeled, a stochastic activity recognition approach may inappropriately estimate the activity state. Often these types of event observations result in false positive activity recognitions.

The PFPN approach uses the information encoded in the Petri Net definition of the activity to reduce the state space of the problem. That is, rather than evaluating every possible marking and event pairing at each frame, the PFPN evaluates only those markings that are reachable from the current hypothesis of the state. The possible event occurrences are also limited to only those events that may occur in a given marking. Note that no assumption of a known duration

Figure 10.16: F-Score vs. Number of Particles for the Technion Data Set (perfect tracking) for FPFPN vs PFPN
model of the events’ time interval, or the inter-event time interval is made. Indeed such a model is not available in any real dataset, rather only in synthetic data such as our Bank dataset. However, as can be seen in Figure 10.6, this advantage did not enable the P-Net approach to greatly outperform the PFPN approach.

The FPFPN approach outperforms the baseline methods in our experiments (see AUC results in Table 10.2). We attribute this performance advantage to the core intuition that distinguishes FPFPN from the other methods. Namely, the modeling of what is possible in the scene (i.e. context) is separate from the modeling of the activity structure. Thus, the observation is used to estimate the context, which in turn is used to estimate the activity state. This is in contrast to the competing approaches which generally seek to estimate the activity state directly from the observation, resulting in a more complex model which makes successful state estimation more difficult.

Like other stochastic approaches, PFPN and FPFPN model the activity recognition problem as a joint probability estimation over many random variables. The size of this state space is directly related to the time complexity of these algorithms. In particular, Particle Filter algorithms, whose complexity depends on the number of particles, require a number of particles that increases with the size of the state space. Thus, in order to make stochastic activity recognition tractable, assumptions are made to reduce the size of this space. In the P-Net model, each event participating in the activity is represented as a variable which contains a start time and a duration. During recognition, the P-Net approach applies an assumption to allow simplification of the joint distribution over the activity state: once an event node becomes active (the corresponding event is in progress), its continued activation depends only on the duration of the activation. Thus, it is assumed that the event duration (or its distribution) is known at the time of recognition. While this assumption works well to reduce the space of feasible solutions for activity recognition, it is only valid in cases where event durations are known to be distributed according to a parametric distribution and a satisfactory amount of labeled training data is available to allow tuning of the duration model parameters. However, in many surveillance applications these conditions do not hold and this assumption reduces the robustness of this approach to the recognition of activities whose composing event durations have large variance and whose distribution cannot be described by common parametric distributions.

The number of particles is the critical factor in determining the running time of the algorithm (this can be understood by inspecting the inference procedure in section 9.3.2). For this reason, it is interesting to examine the performance of the various approaches as the number of particles is reduced. Figures 10.14 − 10.16 show a plot of the f-score of each method as the number of particles is varied. The f-score represents the harmonic average of the precision and recall scores for the best threshold value. It is apparent from the figure that the results achieved by
FPFPN can be also be achieved with significantly less particles (5% of the amount used in our experiments). This observation implies that a FPFPN activity recognition approach can be tuned to achieve significantly smaller running times than the competing formalisms without sacrificing the recognition performance.

The results in the figure are due to the factorization of the state space into simple fragments which have a small number of markings and events (recall that the state of the Bayesian Recursive Filter is composed of all combinations of the underlying Petri Net’s markings and transitions). Because each factor has a relatively small state space (e.g. Figures 9.6, 9.7) a small number of particles is needed to cover this space. Since we estimate the state of each factor independently, we require a small number of (factored) particles to estimate the joint state (context and activity) of the entire system. Thus a small number of particles will provide as good of a recognition performance as a larger number. This is in contrast to other approaches, such as PFPN, which model an activity as a single fragment, which takes into account all possible events including those which are not part of the activity. In addition to being complex and difficult to construct, these models also imply a significantly larger state space than that of the small fragments created by FPFPN. Such a space requires more particles to span, and is the reason that recognition performance deteriorates as the number of particles used decreases.
Chapter 11

Conclusion

The ability of humans to recognize events in video sequences still greatly exceeds that of the most successful automatic video event recognition approaches. One reason for this may be that humans are equipped with a great deal of semantic knowledge about what constitutes a particular event within an event domain. While it is difficult problem to encode all human knowledge, with a powerful representational formalism we contend that it is possible to distill the knowledge needed to correctly recognize and classify events in a particular event domain, particularly simple domains such as those found in surveillance applications.

The Petri Net (PN) formalism is such a representational mechanism which allows us to express semantic knowledge about an event domain. PN event models can robustly model the complex temporal relationships that are inherent to the types of events we are interested in recognizing. The PN representation of the event domain can be used to devise efficient event recognition algorithms with good accuracy.

In this thesis we have explored the application of the Petri net formalism to represent events. In order to efficiently recognize these events we designed algorithms that are able to cope with uncertainty inherent to video data. This uncertainty is caused by noisy signal from the low-level video processing, as well as the fuzziness associated with semantic concepts such as close to or moving fast, that are used to define many events.

Initially, we set out to clarify the terminology in use throughout the field of video event understanding. The hierarchical and temporal extents of video events, the variable nature of the building blocks which are used to construct event models, and the semantic ambiguity of human language have resulted in a confusing terminology fraught with ambiguity and contradiction between various related works in the field. In this work we propose a unifying terminology, divorced from the inexactness of human language.

Throughout this work we have proposed several approaches for representation
and recognition of video events, and experimentally evaluated them in the domain of intelligent surveillance. Our initial work focused on formulating the semantics of the scene using Petri nets in a way that enables capturing natural variance in the duration of sub-events. Subsequently, we proposed a solution to allow propagation of uncertainty from the sub-event to the event level. This required the definition of a model which can capture the semantics of the event definition while still affording probabilistic online reasoning, which can update its estimation of the event recognition as new information becomes available. In constructing this model we made the observation that many of the events we are interested in share many of the same constraints, those constraints that define the physical limitations of the scene. Separating the modeling of these constraints from the model of temporal constraints that define each event allows simplified model construction, component reuse, improved probabilistic inference, and ultimately more accurate and efficient event recognition.

In order to evaluate our algorithms we implemented several competing state of the art approaches for representation and recognition of the types of events we are interested in, composite multi-threaded events (sometimes referred to as activities or scenarios), and compared these to our approaches using the same low-level video processing as input. This comparison was done across several datasets, including synthetic datasets, publicly available datasets, and datasets captured and annotated by the authors. This was done in order to increase the breadth of the experimental basis for our conclusions, as a limited number of datasets exist which contain the complex types of events that we consider in this work.

Our proposed representation and recognition approach was able to outperform the baseline approaches in our evaluations. We also show that our most recent work provides an increase in efficiency that can be ascribed to the separation of context constraints from event constraints.

In summary, the goal of understanding events as they occur in video data remains a worthy area of academic research. When the semantic definition of an event can be formalized as a Petri Net, this thesis offers solutions for how the recognition of these events as they occur in the video data. However, this approach relies on an initial non-trivial modeling of both the scene constraints and the event constraints. Doing away with this step, and inferring event definitions from a large corpus of data still remains the holy grail of this research area. However, even if this were possible, many events of interest (e.g. bank robbery) are so rare that data for them simply does not exist, certainly not in the quantities that would enable inferring complex event models. Thus, it would seem, some way to model semantic knowledge and integrate that knowledge into the recognition process, such as the approaches proposed in this thesis, will be a necessary component of any solution to the problem of video event understanding.
for the foreseeable future.
References


Chelq, N. and M. Thonnat. 1996. Realtime image sequence interpretation for video-surveillance applications. In International Conference on Image Processing (ICIP ’96), Lausanne, Switzerland.


161


167


168


תקציר

マトリクスのビストムとそのフォーマショニングを用いた新しい技術は、最適化問題の解を持つビストムを生成します。

この新しい技術は、ビストムのフォーマショニングを用いて、ゲートウェイと非ゲートウェイを区別することを目指しています。

ゲートウェイとは、ネットワーク内の通信を制御する役割を担うノードを指し、非ゲートウェイとはそれ以外のノードを指す概念です。

ゲーム理論の枠組みを用いて、この新しい技術は、ゲートウェイノードの最適な配置を決定します。

この技術は、ネットワークの安定性と信頼性を向上させるために開発されました。

この研究は、技術学会の公式発表として承認されています。
"Transition Nodes"
ה fv�מיים הירשליים של התואר והיא האלגוריתמים שלהי, הנכון על בניי
ייפון של הסטיזיה מוארת בשתי פעולות. האלגוריתמים של האת הסיכולים על ממון
מסכולים להמחדumu חסר התיאוריה התוכנית בידיעו. לאטרט היסק הסתחובות
אלגנטים, המאפרת עדכון מבוקש מחשבות ועליה זמני. Düיםונות אבדה ויהי
היאושנה שמסיימת פתרון מוכללביעות מוסווז.

iii
ה뭔ך נשעה בחכמה פורפ' אחדר יבלוק
בפコレטה למודיע המหาש

הכרת תודה

אני מודה להכינוו על התמיכה הכספית הגדולה בנסוליתי
הבת מארעות ברידא

חיבר על מחקר

לשם مليי חלך של הדרישות לkeligת חנאות
ודקטור לפילוסופיה

גל ליביא

הwęס סנט הลบינו — מון סנטולני לירשל

ابلד הצע"ב

خوف

ספטמבר 2012
הבחנה מאורעות ביידאצ

גל לבייא