SEMANTIC VIDEO ANALYSIS FOR SURVEILLANCE SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KARANİ KARDAŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JANUARY 2018

Approval of the thesis:

SEMANTIC VIDEO ANALYSIS FOR SURVEILLANCE SYSTEMS

submitted by **KARANİ KARDAŞ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver ______________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün ______________
Head of Department, **Computer Engineering**

Prof. Dr. Ahmet Coşar ______________
Supervisor, **Department of Computer Engineering**, **METU**

Prof. Dr. Nihan Kesim Çiçekli ______________
Co-supervisor, **Department of Computer Engineering, METU**

Examining Committee Members:

Prof. Dr. Ferda Nur Alpaslan ______________
Department of Computer Engineering, METU

Prof. Dr. Ahmet Coşar ______________
Department of Computer Engineering, METU

Assoc. Prof. Dr. Pınar Karagöz ______________
Department of Computer Engineering, METU

Assist. Prof. Dr. Nazlı İkizler Cinbiş ______________
Department of Computer Engineering, Hacettepe University

Assist. Prof. Dr. Aykut Erdem ______________
Department of Computer Engineering, Hacettepe University

Date: ______________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name:	KARANİ KARDAŞ
Signature :	_____________

ABSTRACT

SEMANTIC VIDEO ANALYSIS FOR SURVEILLANCE SYSTEMS

Kardaş, Karani

Ph.D., Department of Computer Engineering

Supervisor: Prof. Dr. Ahmet Coşar

Co-Supervisor: Prof. Dr. Nihan Kesim Çiçekli

January 2018, 118 pages

This thesis presents novel studies about semantic inference of video events. In this respect, a surveillance video analysis system, called SVAS is introduced for surveillance domain, in which semantic rules and the definition of event models can be learned or defined by the user for automatic detection and inference of complex video events. In the scope of SVAS, an event model method named Interval-Based Spatio-Temporal Model (IBSTM) is proposed. SVAS can learn action models and event models without any predefined threshold values and generates human readable and manageable IBSTM event models. The thesis proposes hybrid machine learning methods. A set of feature models named Threshold Model, which reflects the spatio-temporal motion analysis of an event, is kept as the first model. As the second model, Bag of Actions (BoA) model is used in order to reduce the search space in the detection phase. Markov Logic Network (MLN) model, which provides understandable and manageable logic predicates for users, is kept as the third model. SVAS has high performance event detection capability due to its interval-based hierarchical approach. It determines related candidate intervals for each main model of IBSTM and uses the related main model when needed rather than using all models as a whole. The main contribution of this study is to fill the semantic gap between humans and video computer systems such that, on one hand it decreases human intervention through its learning capabilities, but on the other hand it also enables human intervention when necessary through its manageable event model method. The study achieves all of them in the most efficient way through its machine learning methods. The proposed system is applied to different event datasets from CAVIAR, BEHAVE, CANTATA and our synthetic datasets. The experimental results show that our approach improves the event recognition performance and precision as compared to the current state-of-the-art approaches.

Keywords: Event Detection, Markov Logic Networks, Video Surveillance, Event Model Learning, Event Inference

ÖZ

GÖZETİM SİSTEMLERİ İÇİN ANLAMSAL VİDEO ANALİZİ

Kardaş, Karani

Doktora, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Ahmet Coşar

Ortak Tez Yöneticisi: Prof. Dr. Nihan Kesim Çiçekli

Ocak 2018, 118 Sayfa

Bu tez video olaylarının anlamsal çıkarımı konusunda geliştirilmiş çalışmaları sunar. Bu bağlamda, gözetim alanında karmaşık video olaylarının otomatik algılanması ve çıkarılması için anlamsal kuralların ve olay modellerinin öğrenilebileceği veya kullanıcı tarafından tanımlanabileceği bir gözetim video analizi sistemini (SVAS) tanıtmaktadır. SVAS kapsamında, Interval-Based Spatio-Temporal Model (IBSTM) (Aralık Tabanlı Uzamsal ve Zamansal Model) adlı bir olay modeli yöntemi önerilmiştir. SVAS, önceden tanımlanmış eşik değerleri olmadan eylem modellerini ve olay modellerini öğrenebilir ve anlaşılabilir ve yönetilebilir IBSTM olay modelleri üretir. Melez makine öğrenme yöntemleri önerilir ve kullanılır. Bir olayın uzamsal ve zamansal hareket analizini yansıtan Threshold Model (Eşik Modeli) isimli bir küme özellik modeli, ilk model olarak tutulur. İkinci model olarak, tanıma aşamasındaki arama kümesini azaltmak için Bag of Actions (BoA) (Eylem Çantası) modeli kullanılmıştır. Kullanıcılar için anlaşılabilir ve yönetilebilir mantık yüklemleri sağlayan Markov Logic Network (MLN) (Markov

Mantıksal Ağ) modeli, üçüncü model olarak tutulmaktadır. SVAS, sahip olduğu aralık tabanlı hiyerarşik yapısı nedeniyle yüksek performanslı olay tanıma kabiliyetine sahiptir. IBSTM' in her ana modeli için ilgili aday aralıklarını belirler ve tüm modelleri bir bütün olarak kullanmak yerine ihtiyaç duyulduğunda ilgili ana modeli kullanır. Bu çalışmanın ana katkısı, bir yandan öğrenme kabiliyeti ile insan müdahalesini azaltmak, diğer yandan da yönetilebilir olay modeli yöntemi yoluyla gerektiğinde insan müdahalesini mümkün kılacak şekilde, insanlar ve video bilgisayar sistemleri arasındaki anlamsal boşluğu doldurmaktır. Çalışma, sahip olduğu makine öğrenme yöntemleri aracılığıyla tüm bunları en verimli şekilde başarmaktadır. Önerilen sistem CAVIAR, BEHAVE, CANTATA ve sentetik veri kümelerinden oluşan farklı olay veri kümelerine uygulanmıştır. Deneysel sonuçlar yaklaşımımızın, günümüz yaklaşımlara kıyasla olay tanıma performansını ve hassaslığını geliştirdiğini göstermektedir.

Anahtar Kelimeler: Olay Tanıma, Markov Mantık Ağları, Video Gözetimi, Olay Modeli Öğrenme, Olay Çıkarımı

Dedicated to My Family

ACKNOWLEDGMENTS

I would like to express my gratitude and appreciation to my supervisor (my co-supervisor now since she is on sabbatical) Prof. Dr. Nihan Kesim Çiçekli for her endless encouragement, guidance and support throughout this study. I would like to thank Prof. Dr. Ahmet Coşar for becoming my supervisor when I needed. I also would like to thank Assoc. Prof. Dr. Pınar Karagöz and Assoc. Prof. Dr. İlkay Ulusoy who gave me invaluable support during my studies. In addition, I would like to thank Prof. Dr. Ferda Nur Alpaslan, Assist. Prof. Dr. Nazlı İkizler Cinbiş and Assist. Prof. Dr. Aykut Erdem for being members in my thesis defense committee. I also would like to express my special thanks to my best friend Göktan Güzel who helped me a lot for increasing the writing quality of the thesis.

This work was partially supported by the Ministry of Science, Industry and Technology of Turkey and by Havelsan Inc. under Grant SANTEZ 00896.STZ.2011-1. For this reason, I would like to thank the Ministry of Science, Industry and Technology of Turkey and Havelsan Inc.

Last but not least, I would like to thank my family who always encouraged me to complete my dissertation. Particularly, I would like to express my inmost gratitude to my wife, Ayşegül. Without her help, support and encouragement, I could not complete this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BoA	Bag of Actions
BN	Bayesian Networks
CRF	Conditional Random Fields
DBN	Dynamic Bayesian Networks
DMLN	Dynamic Markov Logic Network
EC	Event Calculus
FSM	Finite State Machines
FOL	First Order Logic
HMM	Hidden Markov Models
IBSTM	Interval-Based Spatio-Temporal Model
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
MLN	Markov Logic Network
MPE	Most Probable Explanation
PEL	Probabilistic Event Logic
PN	Petri Nets
SVAS	Surveillance Video Analysis System
SVM	Support Vector Machines
TH Model	Threshold Model

CHAPTER 1

INTRODUCTION

1.1 Motivation

Nowadays, surveillance camera systems play an important role in public security. However, most of these systems are "Simple Image Recording Systems" with the mere capability of recording the images and "Visual Analysis Systems" with limited detection and tracking capabilities. Today, when an incident related with the public security occurs, pre-recorded videos are reviewed by humans. This situation delays the response time of security operations to the incident. Surveillance systems are not used efficiently since so many human interventions are needed and hence automation is limited.

Considering the rapidly increasing number of cameras, it is necessary to detect events automatically. However, that feature is beyond the capabilities of currently available systems. Complex event detection and recognition has become a hot topic in recent years. There are various studies (e.g. [96], [7], [97], [73], [76], [16], [25], [120], [122], [4] and [67]) about this subject. However, many of the applications are scene-dependent and consider certain scenarios. Existing solutions are highly domain-specific and even event-specific. In addition, some systems (e.g. [5], [68], [83], [62], [91] and [97]) are so human-oriented that many human interventions are required.

In general, event detection is a complex process which requires two main levels of processing. These levels can be considered as low level and high level processing. At low levels, objects and people are detected and tracked throughout the video frames and their spatio-temporal relations are calculated with respect to their positions through time. At high levels, using the information obtained from low levels, events are detected by using models, which have been defined or learned apriori. At both levels, studies continue to increase detection and recognition performance. There are various successful studies taking low levels into consideration (e.g. [38], [127] and [27]) in which especially actors for surveillance domain (such as car, bag and human) can be detected and tracked successfully. In addition, even skeleton motions can be captured successfully in studies about game industry such as KINECT / XBOX. Although low level processing is not the main concern of our study, studies on low level processing will be briefly discussed in the related work section of this thesis as well.

High level processing include video event detection, inference of events and prediction of possible events. Finding semantic relations between actors and detecting events are at least as important as the detection and tracking of actors. At high levels, semantic relations are determined by using actor trajectories and spatio-temporal relations. High levels can be considered as a set of levels according to the inference goal. At the first level, features such as speed, distance, and direction are extracted and then interval-based actions (or sub-events, simple events, primitive events) such as run, walk and stand can be inferred. At a higher level of inference, complex events such as meet, left object, fight can be extracted by using the inferred actions. At one step higher level of inference, prediction of possible events can be extracted, which is an important issue in public security for quick interference or preventing undesired situations.

At high levels, the method of event modeling is very important to fulfill the requirements of video event inference. There has been a considerable amount of work on the detection and recognition of video events and video event modeling. Various event types are analyzed, compared and categorized in many different application domains. The literature survey on video event recognition (which is discussed in Section 2) reveals that methods for event modeling should possess some important features.

As the first feature, methods for event modeling should deal with uncertainty in order to be fault-tolerant [57]. Dealing with uncertainty is important since the information coming from the low levels is not always perfect due to noise, occlusions etc. Hence, the semantics should be extracted in such a way that the erroneous or missing information, caused by the aforementioned reasons, is compensated at high level without leading to false event detection.

Performance is another important feature. In surveillance systems, the size of videos is continuously increasing. Learning process does not require high performance algorithms, but the performance is critical for the inference process. Inference processes should be almost real time for quick interference or to prevent undesired situations.

Nowadays, in most of the event recognition applications, event models are defined by a domain expert. However, defining a model for an event is a difficult process. The main reason is that an event may have many scenarios. Domain expert has to prepare all possible scenarios of the event, which is not feasible in many situations. All scenarios must be considered to describe all possible happenings of an event.

The structure of the scene that is recorded and processed is another issue that must be considered in event modeling. It is apparent that automatic inference of event models from data is essential for adaptability and scalability of event understanding systems. If the event model could be learned and defined automatically, it would be easier to deal with such situations. As a result, learning ability from training data is another important feature of event model methods. The advantages of automatic model learning can be summarized as follows:

• There is no need to define a strict model for an event for all scenarios, scenes or low level algorithms.

• All scenarios of the event can be considered automatically with the help of the training data.

• Event model can be updated according to the new event scenarios easily and quickly.

On the other hand, some current solutions are fully machine-oriented, in which machine learning techniques are used in order to prevent user intervention. Most of these systems generate unreadable and unmanageable event models. Event models are learned from training data to provide event detection and recognition. However, these systems need large amount of training data. Automated inference should be increased for intelligent video surveillance, but limited user intervention increases the quality of video inference capability. The ideal event model method should have robust representational capability including semantic relations [57]. It should be semantically meaningful for the user and enable user intervention when needed. The model quality naturally has a high influence on the detection and recognition performance.

1.2 Contributions of the Thesis

This thesis introduces a surveillance video analysis system, shortly called SVAS, which aims at solving the mentioned problems encountered at high levels. Outputs of low-level operations are considered as inputs of the proposed system. In SVAS, semantic rules and the definition of the event models can be learned or defined by the user for automatic detection and inference of complex video events. The resulting framework makes event detection and recognition flexible, while enabling domain and scene independent. The system decreases human intervention but enables human intervention when needed.

We propose a new interval-based hybrid event model method called Interval-Based Spatio-Temporal Model (IBSTM). IBSTM is both machine and human understandable high-level event model, in which different suitable machine learning techniques are used at different phases of the event inference.

SVAS generates the collection of human understandable IBSTM rules as the event model in order to help the user intervene in the learned model when needed. This kind of flexible framework also provides users to define new event models and train them if there is no training data. The necessity of large training data reduces. IBSTM uses Markov Logic Networks (MLN) [82] to generate user understandable models. MLN combines the flexibility of First Order Logic (FOL) and the power of Markov Network on handling uncertainty. First Order Logic is easy to understand for end users, so it provides good semantic information about complex events. However, MLN has some performance deficiencies in video domain since it does not consider the nature of videos. MLN considers time variables in the same way as other variables. MLN tries to find the relation between all variables. This behavior slows down MLN particularly when dealing with huge amount of data flow. To solve MLN's performance problems, IBSTM extends MLN for video domain in a hierarchical manner with the Bag of Actions (BoA) and the proposed Threshold Model methods.

The major contributions of this thesis can be summarized as follows:

- SVAS is developed for surveillance domain. It is scene-independent and can consider different scenarios for various events. It is possible to define basic spatial, temporal and logical relations in the surveillance domain. In addition, SVAS can be used in both calibrated and uncalibrated scenes.
- A new event model method named Interval-Based Spatio-Temporal Model (IBSTM) is proposed. SVAS can learn action models and event models and generate manageable IBSTM event models.
- Threshold Model is proposed to reflect the spatio-temporal motion analysis of an event.

- Hybrid machine learning methods are used and extended. Different suitable machine learning techniques are used at different phases of the event inference such as Bayesian Networks, Bag of Actions and Markov Logic Networks.
- There are not any predefined thresholds in SVAS. Threshold Model can also be used for learning thresholds.
- SVAS can handle uncertainty in order to be fault-tolerant in noisy conditions. Proposed algorithms generate probabilistic results to prevent discretization problems.
- SVAS decreases human intervention through its event model learning ability from training data to ease user operation and prevent user errors.
- IBSTM fills the semantic gap between humans and video systems. Generated event models are readable for the user. SVAS enables the user to control and manage the event model. In addition, the user can define new event models.
- SVAS has high performance event detection capability due to its interval-based hierarchical manner and its high performance algorithms. Threshold Models and BoA Models provide great efficiency in both action and complex event detection by eliminating irrelevant intervals.
- Time variables in SVAS can be defined as point based or interval based.
- The proposed algorithms are tested in different event datasets from CAVIAR, BEHAVE and synthetic datasets. Results show that SVAS improves the event recognition performance and precision as compared to the current state-of-the-art approaches.

The assumptions and limitations of this thesis are stated as follows:

(i) In this thesis, we focus on single camera videos.

(ii) We assume that videos are captured by stationary cameras.

(iii) We focus only on high-level video processing. The outputs of low-level video processing are taken as inputs of the proposed system.

1.3 Organization of the Thesis

The thesis is organized as follows:

Chapter 2 presents the background information and related work on video event detection and recognition. First, basic concepts used in this dissertation are defined. The video event detection concept is introduced and event model methods are discussed. Then, the relevant literature is reviewed.

In Chapter 3, the proposed system (SVAS) is presented in detail. First, the overall architecture of SVAS is introduced and Trajectory Generation Module is presented. Then, Event Model Learning is explained in detail, which includes Action Model Learning, Action Detection and Complex Event Model Learning processes. In addition, Complex Event Detection and basic prediction approaches of SVAS are discussed. Finally, implementation details and sample application are presented.

Experiments and their results are presented in Chapter 4. There are six types of evaluations in this chapter. Evaluations on publicly available datasets named CAVIAR, BEHAVE and CANTATA are discussed first. In addition, synthetic dataset is created and the system is evaluated to measure the effect of missing values. Performance and quality evaluations conclude the chapter.

Finally, in Chapter 5, a short summary of the study is given and the dissertation is concluded with possible future directions for research.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, general concepts on video event and video event detection are given, the background information for the main topics is covered and related work is reviewed. The chapter is organized as follows: Section 2.1 gives brief information about Video Event Detection. Event model concept is discussed and brief information about event model methods is presented. In this thesis, we propose methods based on Markov Logic Networks. Therefore, in this chapter we also present the background information on Markov Logic Networks and its applications Alchemy [22] and Tuffy [106]. Finally, in Section 2.2, related work on video event detection and recognition is presented in two video processing levels. This section reviews recent studies that are most relevant to this dissertation. In addition it presents a comparison of the related work with this study.

2.1 Video Event Detection

An event is something happening in a location at a given time. Video event detection is the process of searching videos for events and identifying occurred events in videos. Video event detection is a way to understand the semantic content of the video. The main goal of video event detection is to identify spatio-temporal events in video and estimate their start and end times.

Events can be grouped into actions and complex events. An action is a simple event performed by a single actor. Actions can be short body movements such as “walking” and “running”. Complex events are events which include more than one actor who reside in a determined closeness. Actors can follow a path named trajectory through the scene as a function of time. Typically, complex events can be considered as an interaction among humans, or between humans and objects. “Meet”, “Fight” and “Left Object” can be given as examples of complex events.

An emerging trend in video event detection is to detect an event automatically. Detection of such kinds of events is a process of finding events in video using pre-learned or pre-defined event models. For this reason automatic event detection requires event models. The process often relies on the comparison of input video parts with event models.

Pre-defined event models are usually defined by the user using static thresholds or assumptions. This kind of models is generally successful in some conditions. However, this manual process is error-prone. The performance of automated event detection increases if event models are learned. Pre-learned event models require learning ability. This process can be named as Event Model Learning and provides an automatic generation of event models. All scenarios of events can be considered automatically with the help of the training data. This automation facilitates user intervention and minimizes errors created by the user in the event model definition. Events are learned by using features such as actor trajectories and spatial relations between them.

Learning ability requires machine learning techniques. In learning process, there are some basic operations. Feature selection and feature extraction operations are done first. In this phase, it is important to find the most distinguishing features. Then according to the selected machine learning technique, the system learns the event model by using training data. After learning operation is completed, the event model is ready for inference.

Learning can be grouped as supervised learning and unsupervised learning according to the training data. Supervised learning uses labeled training data. The desired output is labeled in the training data. Unsupervised learning uses unlabeled training data. In this case, desired output is unknown. Semi-supervised learning can be considered as another group in which both labeled and unlabelled training data are combined in the learning process.

There are various machine learning techniques in literature. Some brief information for most important ones is given in this section, since they are mentioned in the rest of the thesis.

2.1.1 Support Vector Machines

Support Vector Machines (SVM) [40] are Kernel Methods [121]. In these models no explicit event description exists. In SVM, group of supervised learning algorithms is used for classification and regression. A SVM model predicts whether a new example falls into one category or the other in training phase. The input data is mapped into a high dimensional feature space (kernel trick). In this high dimensional feature space, a linear classifier is created. The main goal is to find a hyper-plane which separates classes. SVM is generally used for non-separable cases. In SVM, it is possible to distinguish two groups by drawing a boundary between two groups in a plane for classification. The place where this border can be created is found by calculating the farthest place to their members of each group. In order to accomplish this, two near and parallel border lines are drawn on the two groups and these boundary lines are drawn closer together to produce a common boundary line.

2.1.2 Graphical Models

Graphical models are models such that relations between variables are established using graphs. In Graphical Models semantic information is given explicitly. Finite State Machines (FSM) [57] and Decision Trees [96] are typical examples of this category [57].

FSM is a deterministic model and shows flow of states. Sequential events are suitable for FSM and they can be defined in FSM as a sequence of states. In Decision Trees, leaves, nodes and edges determine classes, queries and results, respectively. Decision Trees become unsuccessful as the number of classes increases and the number of training data decreases. Overestimating is another problem of Decision Trees.

The formalism of FSM and Decision Trees is well understood. However, they are not probabilistic models. For this reason, they are not suitable for domains where there is uncertainty.

2.1.3 Probabilistic Graphical Models

Probabilistic Graphical Models can handle uncertainty by using joint distribution of random variables. There are two main inference query types in Probabilistic Graphical Models which are conditional probability query and most probable assignment. Markov Network, Hidden Markov Model (HMM), Bayesian Inference, Bayesian Network (BN), Dynamic Bayesian Network (DBN), Conditional Random Fields, Neural Network and Deep Learning are in this category.

Markov Network [82] is an undirected probabilistic graphical model. Nodes represent random variables. Edges represent relations between random variables. Interactions are represented as potential functions. There is one potential function for each clique by default. To increase the performance a log-linear model is used and exponentiated weighted sum of features is used instead of potential functions. Log-linear model is used for making linear in order to decrease the dimension of data. For each clique, a weight and a feature are assigned. Cyclic relations can be defined in Markov Network. Independence checking in Markov Networks is very easy, only neighbor values are considered.

HMM [70] is a kind of directed probabilistic graphical model where the aim is to find hidden state variables. Since they can model the temporal evolution of the state, they are suitable for domains in which continuous knowledge of past and present states exists. Current observations are dependent only on the current state and the current state is only dependent upon the previous state (the Markov property). The parameters of the HMM model may be learned from the training data or specified manually. The number of HMM states are generally determined empirically. In the training phase of HMM, the number of states is specified. High order relations cannot be modeled in a HMM. HMM has an efficient inference and parameter learning algorithm such as the Baum-Welch algorithm which is based on maximum likelihood. Since the complexity of exact algorithms is NP complete, approximation algorithms are used for complex models. There are various extensions of HMMs.

BNs [29] are another kind of probabilistic graphical models which represent a set of random variables and their conditional independencies. They are directed acyclic graphs which are based on Bayesian theorem. Nodes represent random variables, and arcs represent conditional independencies between the variables. For each random variable a Conditional Probability Distribution table is kept. This structure of the BN shows the joint probability over all variables. BNs have efficient inference and learning algorithms. BNs do not model temporal relations. DBNs [70] are temporal extensions of BNs. Cyclic relations can be defined in DBNs. However, DBNs cause high computational complexity and require large amounts of training data.

Conditional Random Fields [107] are undirected probabilistic graphical models. They can be considered as the generalization of HMM. Feature selection is not limited to the current observations in Conditional Random Fields. Unlike HMM, relations can be established between current state and past or future states so combinations of past and future observations can be considered. However, it demands a high parameter learning time.

Neural Network [57] consists of related layers and transmission between these layers. Layers consist of interconnected nodes. These nodes contain an activation function. Training phase of Neural Network is too slow. Deep Learning [59] provides a powerful set of techniques for learning in Neural Networks and the results of Deep Learning methods are very successful. However, the generated model is not semantically readable.

2.1.4 Semantic Models

Semantic models enable explicit specification of complex relations between variables. Semantic models are easy to understand for users and reflect the semantic content well. Petri Nets (PNs), Constraint Satisfaction, Grammars, Logic-based Approaches can be considered as Semantic Models [57]. Normally, semantic models cannot handle uncertainty. To cope with the uncertainty problems, some extensions to these models have been developed.

PN [34] is a kind of graphical model that represents information flow explicitly using states (nodes), transitions (event) and tokens (event instance) as a bipartite graph. PNs can model semantic relations including temporal relations, hierarchy and ordering. Semantic nature of PNs makes learning PN models infeasible. For this reason, PN models are usually deterministic and specified manually by the user.

Constraint Satisfaction [79] represents the model as a set of semantic constraints. The main advantage of Constraint Satisfaction models is that the semantic constraints can be formulated.

Grammar models [75] consist of three component sets, which are terminals, non-terminals and production rules. Stochastic Grammars (Probabilistic Grammars) are an extension of Grammar models in which probabilities can be associated with production rules. For this reason Stochastic Grammars can handle uncertainties.

In logic-based approaches, models are specified as a set of logical rules. Inference is done using logical inference techniques, such as resolution or abduction. These approaches cannot handle uncertainty. Event calculus can also be considered as logic-based model and is based on first-order predicate logic, including temporal formalism, for representing and reasoning about events and their effects [7 and 8]. If the number of predicates increases, performance problems occur in logic-based approaches.

2.1.5 Bag of Words

Bag of Words is another model type in which data (particularly text data) are represented as a bag. Bag of Words ([13] and [109]) method is first developed for document organization. Word frequencies in documents are considered in this method. Topic models ([55] and [109]) are similar to bag of words approach. Main disadvantage of these methods is that they remove all spatial information. For this reason, these methods are considered as non-temporal methods in the literature [109].

2.1.6 Markov Logic Networks

Markov Logic Networks (MLNs) [82] are probabilistic relational graphical models. MLNs can be considered as both Probabilistic Models and Semantic Models. For this reason, MLNs have advantages of those models. MLNs are combination of First Order Logic (FOL) [99] and Markov Network. FOL is a powerful language and it can express complex, relational information well. Constants, variables, functions and predicates can be defined in FOL. FOL is very flexible and provides compact representation for a wide range of domain knowledge. However FOL cannot handle uncertainty. For this reason, it is not suitable alone for real world which includes uncertainty and probability. Rules can be defined as a set of certain facts by using FOL. Markov Network provides uncertainty handling so strict rules becomes softer. Combination of FOL and Markov Network provides ability to model complex information that can include probability. MLNs can be used to model complex relations in a more meaningful way and handle uncertainty.

MLNs are based on first-order logic. MLNs formulas or clauses are attached with weights. MLN formulas define the topology of a Markov network. A MLN is a template for Markov Networks, based on logical descriptions. Predicates in the template are generated as nodes in the network. In this network, edges represent logical connectives in formulas and vertices represent possible groundings of formulas. A ground formula is a formula which is constructed by only ground terms and a ground term is a term which contains no variables. All variables are replaced by constants in ground terms. As a result, a set of weighted first order formulas are generated. Knowledgebase is kept as FOL predicates. Weights attached to predicates determine probability.

Theoretically, a MLN L can be considered as a set of pairs (f, w) where: f is a first-order logic formula and w is a real number which is the weight of the formula. Probability in MLN is formulated as follows:

$$P(x) = \frac{1}{Z} \exp\left(\sum_{k \in ground\ formulas} w_k f_k\ (x) \right)$$

(Z is partition function) (2.1)

$$f_k(x) = \begin{cases} 1, & if\ kth\ formula\ is\ satisfied\ given\ x \\ 0, & otherwise \end{cases}$$

There are three main operations in MLN: parameter learning, inference and structure learning. There are various efficient probabilistic algorithms developed for these operations. Inference operation is the process of calculating the probability or most likely state of query atoms. For inference operation, Most Probable Explanation (MPE) and Maximum A Posteriori (MAP) based algorithms are used ([22] and [82]). MaxWalkSAT algorithm is used in order to maximize the sum of weights of the satisfied clauses. MaxWalkSAT is a kind of weighted satisfiability solver algorithm [82]. However, this algorithm uses too many resources and its performance is low. For this reason, lazy versions of MaxWalkSAT is used [22 and 82]. In addition, approximate inference algorithms such as Markov Chain Monte Carlo (MCMC) [35] is used in order to find marginal and conditional probabilities. MCMC uses randomized sampling method. In order to increase MCMC efficiency, MC-SAT algorithm was developed. MC-SAT can be considered as a combination of MCMC and the SampleSAT satisfiability solver ([22] and [82]).

MLN learns from training samples. In parameter learning, parameters or weights are learned generatively or discriminatively. Weights are learned generatively by maximizing the pseudo-likelihood of the training data. To overcome overestimation problem, discriminative learning is used. The formula of discriminative weight learning is follows:

Maximize conditional likelihood of query (y) given evidence (x):

$$\frac{\partial}{\partial w_i} log P_w(y|x) = n_i(x,y) - E_w[n_i(x,y)]$$

$$n_i(x,y) : \; Number\ of\ true\ groundings\ of\ clause\ i\ in\ data \tag{2.2}$$

$$E_w[n_i(x,y)] : \; Expected\ no.true\ groundings\ according\ to\ model$$

In structure learning, features can be learned from an empty or existing knowledge base using integer linear programming with arbitrary clauses and MAP score. Weighted version of pseudo-likelihood algorithm is used in this process ([22] and [82]).

2.1.6.1 Alchemy

Alchemy is a software package developed for implementation of MLN [22]. Alchemy provides a set of algorithms for structure learning, weight learning, and inference operations of MLN. Alchemy can perform probabilistic and MAP/MPE inferences. MaxWalkSAT and LazySAT are MAP/MPE inference algorithms in Alchemy. Lifted version of Belief Propagation algorithm decreases running time and memory usage. In addition, Alchemy has MC-SAT, Gibbs Sampling, and Simulated Tempering algorithms. Default inference algorithm of Alchemy is MC-SAT algorithm as shown in Algorithm 1. For parameter learning, Alchemy's discriminative weight learning algorithms are

Voted Perceptron, Conjugate Gradient, and Newton's Method [22]. For structure learning, the default algorithm in Alchemy is beam search.

Algorithm 1: MC-SAT Inference Algorithm in Alchemy

```
INPUT: clauses, weights, num samples
1: $x^{(0)} \leftarrow$ Satisfy(hard clauses)
2: for i $\leftarrow$ 1 to num samples do
3:     M $\leftarrow$ Ø
4:     for all $c_k \in$ clauses satisfied by $x^{(i-1)}$ do
5:         With probability $1 - e^{-wk}$ add $c_k$ to M
6:     end for
7:     Sample $x^{(i)} \sim U_{SAT(M)}$
8: end for
```

Typical usage of Alchemy is as follows: First, the model is prepared. Model preparation includes definition of the model as MLN rules by using First Order Logic predicates. Then the weights of the rules are determined by Alchemy using training data. Finally, inference is done with weighted model by using Alchemy.

“learnstruct”, “learnwts” and “infer” are basic commands of Alchemy. “learnstruct”, and “learnwts” are learning commands and take input “.mln” files, output “.mln” file and training “.db” files. “infer” command takes input weighted “.mln” files (either learned or manually weighted), output file for result, evidence “.db “files and query predicates. “.mln” files contain MLN rules with declarations and formulas. “.db” files contain a set of ground atoms.

Sample usage of MLN weight leaning is as follows. First, unweighted MLN file is created in which predicate definitions and rule definitions exist. Figure 1 shows a sample unweighted MLN file.

```
// Predicate definitions
*Friends(person, person)
Smokes(person)
Cancer(person)
// Rule definitions
!Smokes(a1) v Cancer(a1)
!Friends(a1,a2) v !Smokes(a1) v Smokes(a2)
!Friends(a1,a2) v !Smokes(a2) v Smokes(a1)
```

Figure 1 Sample Unweighted MLN file (Unweighted.mln)

Then training file is prepared as shown in Figure 2.

```
Friends(Ali, Ahmet)
Friends(Ali, Nur)
Friends(Ali, Elif)
Friends(Nur, Elif)
Friends(Zeynep, Mehmet)
!Friends(Zeynep, Elif)
Smokes(Ali)
Smokes(Nur)
Cancer(Nur)
```

Figure 2 Sample training file (Training.db)

Sample Alchemy command for weight learning operation is as follows:

learnwts –d -i Unweighted.mln –o Weighted.mln –t Training.db –ne Cancer

Result of this operation is prepared by discriminative weight learning algorithm and written into file named “Weighted.mln”. Figure 3 shows content of the file.

```
// Predicate definitions
*Friends(person, person)
Smokes(person)
Cancer(person)
// Rule definitions
0.5 !Smokes(a1) v Cancer(a1)
0.4 !Friends(a1,a2) v !Smokes(a1) v Smokes(a2)
0.4 !Friends(a1,a2) v !Smokes(a2) v Smokes(a1)
```

Figure 3 Content of Weighted.mln file

Inference is done using weighted .mln file and evidence .db files. The structure of evidence file is similar to training files. Again it contains ground atoms that show evidences. Sample evidence file is shown in Figure 4.

```
Friends(Serhat, Tuncay)
Friends(Serhat, Burak)
Friends(Serhat, Oktay)
Friends(Burak, Oktay)
Friends(Bora, Selma)
!Friends(Bora, Oktay)
Smokes(Serhat)
Smokes(Burak)
```

Figure 4 Sample evidence file (Evidence.db)

Sample Alchemy command for infer operation is as follows:

```
infer -ms -i Weighted.mln -r inferResult.result -e Evidence.db -q Cancer
```

Result of this operation is prepared by MC-SAT algorithm and written into file named “inferResult.result”. Figure 5 shows content of the file.

```
0.75 Cancer(Burak)
0.65 Cancer(Serhat)
0.50 Cancer(Tuncay)
0.45 Cancer(Oktay)
```

Figure 5 Content of inferResult.result file

2.1.6.2 Tuffy

Tuffy [106] is another software tool developed for implementation of MLN. It is an open-source Markov Logic Network inference engine. Tuffy is developed using Java programming language and uses PostgreSQL [78]. PostgreSQL is a powerful, open source object-relational database system. Designers and developers of Tuffy used Alchemy as a reference system. For this reason, Tuffy is very similar to Alchemy and command options are mostly compatible with Alchemy. Tuffy is capable of Markov Random Field partitioning, MAP inference, Marginal inference and Weight learning operations. Number of implemented algorithms of Tuffy is less than Alchemy's. However, since Tuffy is Java-based, it is platform independent. Default inference algorithm of Tuffy is MAP inference algorithm as shown in Algorithm 2. Commands of Tuffy are similar to Alchemy. The main difference is that query parameters are given in query files. Discriminative Weight learning and MAP inference parameters of Tuffy operations for the example given section 2.1.6.1, are as follows:

-learnwt -i Unweighted.mln -e Training.db -queryFile Query.db -r Weighted.txt -mcsatSamples 50 -dMaxIter 100

-i Weighted.mln -e Evidence.db -queryFile Query.db -r inferResult.result

Algorithm 2: MAP Inference Algorithm in Tuffy

```
INPUT: A: initial active ground atoms, C: initial active ground clauses, MaxFlips, MaxTries
1: lowCost ← +∞, s* ← 0
2: for try = 1 to MaxTries do
3:     s ← a random truth assignment to A
4:     for flip = 1 to MaxFlips do
5:         pick a random c ∈ C that's violated
6:         rand ← random real ∈ [0, 1]
7:         if rand ≤ 0.5 then
8:             atom ← random atom ∈ c
9:         else
10:            atom ← atom in c with lowest d-cost
11:        if atom is inactive then
12:            activate atom; expand A, C
13:        flip atom in s; recompute the cost
14:        if cost < lowCost then
15:            lowCost ← cost, s* ← s
16: return s*
OUTPUT: s*: a truth assignment to A
```

2.2 Related Work

Research on complex event detection and recognition has been an active topic in both artificial intelligence and computer vision areas in recent years. There are various studies at all levels of this topic that can be grouped in many different categories such as: methods used, modeling techniques, considered features, studied levels, targeted event types, and domain or input types. In this section, studies are grouped in their prominent characteristics.

2.2.1 Event detection using low-level video processing

Pixel-based operations can be considered as the lowest level processing in complex event recognition process. In these methods, pixel level primitives such as color, texture and gradient are considered. Some of the studies in the literature try to solve event detection, action detection or anomaly detection issues directly using pixel-based operations (e.g. [103], [14] and [48]).

In [103] abandoned and removed objects can be found using background subtraction and foreground analysis. In [14] abandonment of an object studied. If unattended object is detected, then owner of the object is searched.

[92] uses gradient and histogram algorithms in order to detect left objects.

[21] applies some low level algorithms such as Latent SVM on still images to analyze actions such as “take a photo”, “play music”, “riding bike”, “riding horse”, “running” and “walking”.

The recognition of group activities is one of the hot research topics. In [53] discriminative group context feature and gated recurrent unit methods are proposed and used in order to recognize group activities. In [10] a group activity descriptor and recognition method based on trajectory analysis are proposed and used for group activity recognition.

There are some violence detection studies in the literature. [123] proposes semi-supervised dictionary learning approach for violence detection. [111] focuses on fighting event detection using interaction energy force and low level features without any object extraction or tracking method. In [125], Gaussian Model of Optical Flow and Orientation Histogram of Optical Flow based approach is developed for violence detection.

Anomaly Detection is another topic in low-level video event detection and there are lots of studies about this subject (e.g. [113], [128], [90], [65], [48], [118]). An anomaly can be considered as an observation which does not conform to expected normal behavior. Anomaly detection is about detecting those irregular behaviors. In anomaly detection, a model of expected behavior is learned and anomalies are detected by finding patterns that deviate from the model. Traffic events (illegal U-turns) and events of crowded people are considered in anomaly detection. In addition, there are some trajectory based anomaly detection studies in literature (e.g. [77], [60], [89] and [30]). In [61] anomaly detection techniques are discussed in three main groups which are classification-based anomaly detection techniques, statistical-based detection techniques and clustering-based detection techniques.

In all of these studies, there is no high-level reasoning or inference. Moreover, user understandable event representation is not considered.

Some other studies try to generate reliable input data for higher levels. Subjects of these studies contain background subtraction, object detection, object tracking and object recognition (e.g. [38], [127], [101], [116], [117] and [27]). In addition, some studies about human recognition (e.g. [42] and [102]) can also be considered in these groups. In gaming industry, applications such as KINECT has successful tracking capabilities. In [26], action detection is done via KINECT using 3D Histograms of Scene Flow and Global Histograms of Oriented Gradient methods.

There are some trajectory based video analysis studies in literature such as [49], [56], [63], [45], [47], [124], [110]. In these studies, trajectory based analysis is done with low level operations.

2.2.2 Event detection using high-level video processing

The studies at high levels can be grouped according to the methods they use. Rule-based methods, such as [5], [68] and [83] cannot handle uncertainty since they are not probabilistic. [5] uses Jess-Rule Engine in order to resolve conflicts and find optimal solution. However, [68] proposes event morpheme. In this study, there are three levels in the event detection, which are object detection, simple event detection and semantic scene description detection. [119] extends rule-based system in fuzzy-based manner. Rules, which describe events, are given by the domain expert directly. In these studies, there are predefined thresholds.

In event recognition, probabilistic models are often used in many applications. Methods such as Neural Network, Bayesian Inference (e.g. [62]), Bayesian Network (BN), Dynamic Bayesian Networks (DBN) (e.g. [36], [91], [93] and [104]), Hidden Markov Model (HMM) (e.g. [74], [69], [71], [70], [43] and

[20]), Conditional Random Fields (CRF) (e.g. [107] and [112]) can be considered in this group. These methods need training data and events are represented with probabilistic models. Approximation techniques are usually used to perform learning and inference. Event recognition is usually performed by using maximum likelihood estimation given observation sequences. Although these probabilistic approaches can handle uncertainty, they have the disadvantage that the number of actors and states cannot be changed dynamically in the model. They are not flexible, and hence not suitable for the video surveillance applications, where the number of actors always varies in time. They are not suitable for complex models neither. In addition, these models are generative. When the number of the features increases, their performance degrades comparing with the classifier-based approaches [96]. They also cannot model temporal constraints well, since they are based on time points instead of time intervals. In addition, these models have limited representation capabilities and so they are not semantically meaningful because of their complexity. They require large training sets to learn structure that a human cannot easily describe.

There are also video event recognition studies which use other graphical models such as Finite State Machine (FSM) (e.g. [9] and [64]) and Decision Trees (e.g. [96]). FSM is a deterministic model and provide computationally efficient solutions. On the other hand, FSM cannot have hidden states and cannot handle uncertainty because of the sequence of states are fully observable.

Some studies in the literature use semantic event models. Petri nets (PNs) (e.g [34] and [58]), grammar models, constraint satisfaction (e.g. [79], [31] and [1]), and logic-based approaches can be considered in this group. [34] proposes Parking-Lot application using PNs. Nice graphical representation is used.

Semantic event models capture the structure of the event successfully. These models are usually fully specified using domain knowledge and are not usually learned from the training data. Because of their high-level nature, they are often manually specified by a domain expert. These models are deterministic and the reasoning under uncertainty is not feasible generally. Since they are not probabilistic by default, they are sensitive to low-level failures.

Stochastic grammars (e.g. [75], [87] and [88]) constitute a kind of grammar model which can be considered as probabilistic models. They can give a probability score to a number of legal parses. This extension provides a mechanism to deal with the uncertainty. In [75], Stochastic Context Sensitive Grammar is used as And-Or Graph to represent events and relations between events for office events. In addition, event interpolation concept is mentioned in order to solve occlusion problems. [87] defines group activities as a formal representation using context-free grammar. However, [88] defines probabilistic representation of group activities using probability distribution.

Constraint satisfaction models represent events as a set of semantic constraints and recognition problems as constraint satisfaction. The main advantage of this approach is that the constraints can be formulated semantically. So, domain expert can model composite events with complex temporal constraints. There are also some studies (e.g. [84], [85] and [86]) that try to compose probabilistic constraint satisfaction to add an uncertainty handling mechanism. [85] computes the Gaussian probability density function for each feature in order to handle uncertainty. Rules are weighted manually. In [84] and [86] a complex event recognition approach with probabilistic reasoning is proposed and event description language is improved. For each sub-event, utility is assigned by human expert manually.

Logic-based models have well-defined understandable structure. In this type of approaches (e.g. [23] and [95]), knowledge about an event domain is specified easily by the domain expert as a set of logic rules (predicates). Event recognition is done using logical inference techniques such as resolution. However, these techniques are not tractable in general when the number of predicates is too many. In addition, logic-based approaches cannot handle uncertainty. Logic provides methods to be semantically meaningful for user.

However, any false detection or miss may lead to a wrong event detection situation since those methods cannot handle uncertainty. There are also some studies which can handle uncertainty [44]. In these studies, probability is integrated to handle noisy conditions. Some authors also proposed new combined models in the literature. Markov Logic Network (MLN) [82] can be considered as the most important one. MLN combines the advantage of logic with Markov Network and used in event detection in many applications such as ([72], [105], [15], [37], [50], [52] and [41] since it joins uncertainty handling and logical expressiveness properties. This method handles uncertainties in a flexible manner where the number of states and actors are allowed to change in time. Furthermore, relations can be represented in a robust way. However, in MLN models, performance decreases if the number of logic predicates increases. Particularly in surveillance video domain, in which there is continuous data flow, there are so many predicates. In addition, MLN has poor temporal reasoning capabilities. For time variables, relations are queried between one another which are meaningless for unrelated time variables. In Dynamic Markov Logic Network (DMLN); time point based extension is added but there are no rules for computing the intervals. Since MLN rules are semantically understandable, there are also studies in which ontologies are tried to be combined for different domains ([33], [81] and [6]).

Like the other logic-based approaches, Event Calculus does not consider the problems of noise or missing observations that always exist in real world applications. [7], [98], [8] and [97] can be considered as important studies using event calculus in event recognition.

Topic models or bag of words approaches are non-temporal methods in the literature (e.g. [13, 55, 12]). These approaches are mainly proposed for document categorization. In the visual domain, an image or a frame of a video can be represented by a bag of features. For example, in [55] each primitive event is kept as a topic and each activity is kept as bag of words to understand the scene in traffic events. The main reason for the success of these approaches is that they can cope with many object types simultaneously. In these approaches, temporal ordering of observed actions is not necessary, and they do not need explicit tracking or event detection. In addition, they do not require many training data. However, their main disadvantage is that temporal relations, which are very important in many complex event types, are not considered in these models.

For performance reasons, interval-based approaches are also studied in the literature. For instance, [126], [39] and [17] are the studies in which event recognition processes are extended in an interval-based manner, along with MLN as in [100] and [66].

Nowadays, methods that use Deep Learning become increasingly popular (e.g. [19], [46], [114], [32] and [94]). Methods based on Deep Learning have achieved promising performance in image classification and action recognition tasks and are generally used for anomaly detection. [19] uses deep learning methods to extract discriminative features from video data in anomaly detection. [46] presents video event detection application based on a regularized multi-modality deep learning method. The proposed application can encode the relationships between the visual and audio modalities. [114] proposes unsupervised deep learning framework for anomaly detection in complex scenes. The proposed method utilizes deep neural networks in which feature representations can be learned. [32] presents a deep Convolutional Neural Networks infrastructure which can detect pre-defined video events. [94] uses Discriminative Deep Belief Neural Network in order to detect activities.

As stated before, semantically understandable event model is given directly by a user in most of the cases. There are limited studies in which a user understandable model is tried to be generated from training data (e.g. [51], [24] and [73]).

Some studies are worth discussing in detail because of the similarities with the proposed method in this study. These similarities can be grouped into the categories such as the methods and modeling techniques used, the features considered and test data.

In [80], Bayesian classifier method is used. [76] compares many methods such as Hidden Markov Model, J.48 tree, Bayesian classifier and Neuro-Fuzzy. In both of the studies, CAVIAR Dataset is used and there are not any interests for user manageable model. Their feature sets are similar to ours.

In [62], events are recognized in three levels using Bayesian Inference after trajectory smoothing is done using median filter. Events are represented as hypotheses, related cues are represented as evidences. They use Pets2006 Dataset. In [91], Bayesian Inference is used to detect “leave object”, “get object”, “use object”, “walking” and “handup” by exploiting different cues like skin detection, trajectory analysis, people likelihood and group likelihood. In both of these studies, there is no event model learning. Instead, user given predefined models and thresholds are used.

In [96], classifier-based approach is used for recognizing high-level events in CAVIAR Dataset. Space Time Volumes are proposed for describing motions and shapes of objects. These features are clustered. Clustered features can be considered as primitive events. After clustering operation, classification is done using decision trees in order to create event models. “meeting”, “pocket picking”, “fighting”, “leaving bag”, “forbidden zone” are considered events. Created event models are not understandable as models defined using MLN.

In [85] and [86], probabilistic extensions are proposed to handle the uncertainty for Constraint Satisfaction Models in health care system, airport activity monitoring and simple activities such as “person sitting” or “in living room”. There is no event learning process. Event model is predefined and similar as logical rules. There is no weight learning operation for rules, as event model weights are given manually.

In [7], event calculus is used. Event calculus can represent interval-based relations well but cannot handle uncertainty. CAVIAR Dataset is used for detecting “fighting” and “leaving an object” events. There is no simple event detection so events such as “walking” and “inactive” must be given to the system. In [98], they extend the previous study by integrating MLNs to Event Calculus (EC). EC predicates are converted to MLN rules. In this conversion process, time intervals are lost because they define time point based predicates in MLN. In both of these studies, there is no event model learning capability. Even weights of MLN rules are given by manually. Only “meet” event is experimented. In [97], Prob-EC is proposed which is a combination of EC and ProbLog. ProbLog is a probabilistic extension of the logic programming language Prolog. Prob-EC can deal with uncertainty. However, there is no learning mechanism. It has predefined thresholds for some attributes such as closeness. In addition, there is no clear interval-based inference mechanism. Evaluation is done using CAVIAR Dataset.

In [73], meeting detection is studied in which people trajectories are converted into semantic terms. The model is learned by employing a soft-computing clustering algorithm that combines trajectory information and motion semantic terms. However, learned model is not weighted clearly which is important for handling uncertainty. In addition, no time interval-based approach is used. Evaluation is done using CAVIAR Dataset.

[105] is a MLN study to probabilistically infer activities in a parking lot. Domain knowledge is defined as MLN rules without learning. In [15], events that may occur in an office environment are recognized by using DMLN. Only close up view events such as writing, reading, eating are considered. In [37], complex events are inferred from multimodal data using MLNs for surveillance domain. In these three studies, rules are defined as time points instead of time intervals. In addition, there is no event model learning capability and user manageable event model definition.

In [100], MLNs are used in an interval-based manner for cooking plan event such as “make tea” and “make coffee”. Low-Level Events are detected using KINECT. MLN is used for representing complex events without any learning.

In [66], Allen’s Interval Logic [3] is combined with MLNs for basketball domain. In both of these studies, time interval can be defined but, these studies do not have an event model learning capability and a user manageable event model definition. In addition, there is not any interval based inference. Rules and weights are given manually by domain expert.

In [126], a new model is proposed which is a combination of Bayesian Network and Interval Algebra. They propose parameter learning and structure learning algorithms to model events. The proposed model is a kind of directed acyclic graph. Thus, the model is converted into Bayesian Network so that Bayesian Network algorithms can be used. Basketball and American football domains are used for experiments. However, event models that are learned by the system are not user manageable since models are not user readable. In [17], Probabilistic Event Logic (PEL) is presented, which uses weighted event-logic formulas to represent probabilistic constraints among events. However, the low-level uncertainty is not handled. In addition, they consider only the recognition of primitive events of basketball game such as shooting and dribbling.

In [24], Inductive Logic Programming based event model learning is used. They use MLNs for event models and can define interval-based predicates. Experiments are done for only events in airport domain such as “aircraft arrival”, “positioning” and “departure”. This method is not suitable for domains where there is no tree like object type hierarchy because of great increase in search space. [13] uses the bag of activities approach in PETS 2006 Dataset. Simple events are found using DBN and complex events are found using bag of activities. However, it is not suitable for domains in which time relations between simple events are important.

Literature survey about methods for detection, recognition of video events, and video event modeling (e.g. [57], [11], [54], [108] and [2]) reveal that an ideal event model should consider spatial, temporal and logical relationships and should capture high-level semantics such as long-term temporal dependence.

The ideal event model should have a robust representational capability including semantic relations. It should be semantically meaningful for the user, it should also have learning ability from the training data to ease user operation and prevent user errors. In addition, an ideal event model should handle uncertainties to be fault-tolerant.

Another important property the ideal event model should have is that its recognition algorithms should have high performance. According to the literature, there is no event model method which provides all of these features as a whole. Moreover, there is no uncertainty handler event model method, in which models can be learned or can be defined as user manageable rules. In this study, IBSTM is developed for SVAS in order to provide all these important model properties and SVAS is designed as an efficient video analysis system for surveillance domain by considering the aforementioned video domain needs.

CHAPTER 3

SVAS: SURVEILLANCE VIDEO ANALYSIS SYSTEM

In this chapter, SVAS is presented in detail. An overview of the proposed system and the main processes are discussed. The organization of the chapter is as follows: First, the overall architecture of SVAS is introduced in Section 3.1. In Section 3.2, Trajectory Generation Module is presented. In Section 3.3, Event Model Learning is explained in detail. In this section, Action Model Learning, Action Detection and Complex Event Model Learning are presented in sections 3.3.1, 3.3.2 and 3.3.3, respectively. In addition, Section 3.3.3 includes definition and properties of Interval-Based Spatio-Temporal Model (IBSTM). Complex Event Detection is discussed in Section 3.4. In Section 3.5, simple approaches of SVAS for prediction are discussed. Finally, implementation details of SVAS and sample application using SVAS are presented in Section 3.6.

3.1 The Overall Architecture Of The System

In SVAS, there are five main modules, which are Trajectory Generation, Action Model Learning, Action Detection, Complex Event Model Learning and Complex Event Detection. These main modules are used in two main SVAS processes, which are event model learning (Figure 6) and event detection (Figure 7).

In event model learning (Figure 6), actor trajectory generation is the first operation using Trajectory Generation Module. Trajectory Generation Module parses and prepares video data for processing. Event model learning includes two scenarios, which are action model learning and complex event model learning. Action model learning is done using labeled training data for specific actions in Action Model Learning Module. Action Model Learning Module generates action models using the training data. A set of feature models named Threshold Models (TH Models) are kept with a Bayesian Network as action models.

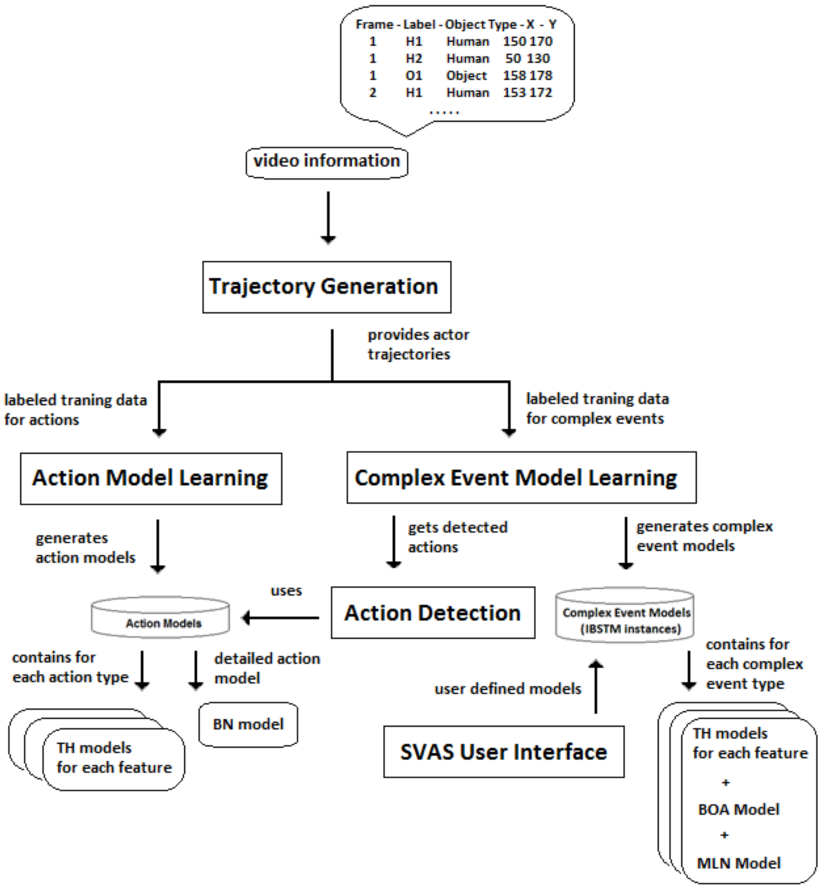

Frame - Label - Object Type - X - Y
1 H1 Human 150 170
1 H2 Human 50 130
1 O1 Object 158 178
2 H1 Human 153 172
.....
video information
Trajectory Generation
provides actor trajectories
labeled traning data for actions
labeled traning data for complex events
Action Model Learning
Complex Event Model Learning
generates action models
gets detected actions
generates complex event models
uses
Action Models
Action Detection
Complex Event Models (IBSTM instances)
contains for each action type
detailed action model
user defined models
contains for each complex event type
TH models for each feature
BN model
SVAS User Interface
TH models for each feature + BOA Model + MLN Model

Figure 6 Event Model Learning Process

In Figure 6, Complex event model learning is done using labeled training data for specific complex events in Complex Event Model Learning Module. Complex Event Model Learning Module generates complex event models as proposed IBSTM models using the training data and Action Detection Module. Generated complex event models are human understandable so that a user can interfere generated models via SVAS User Interface Module if desired. In addition, the user can create his/her own event models by using the same module.

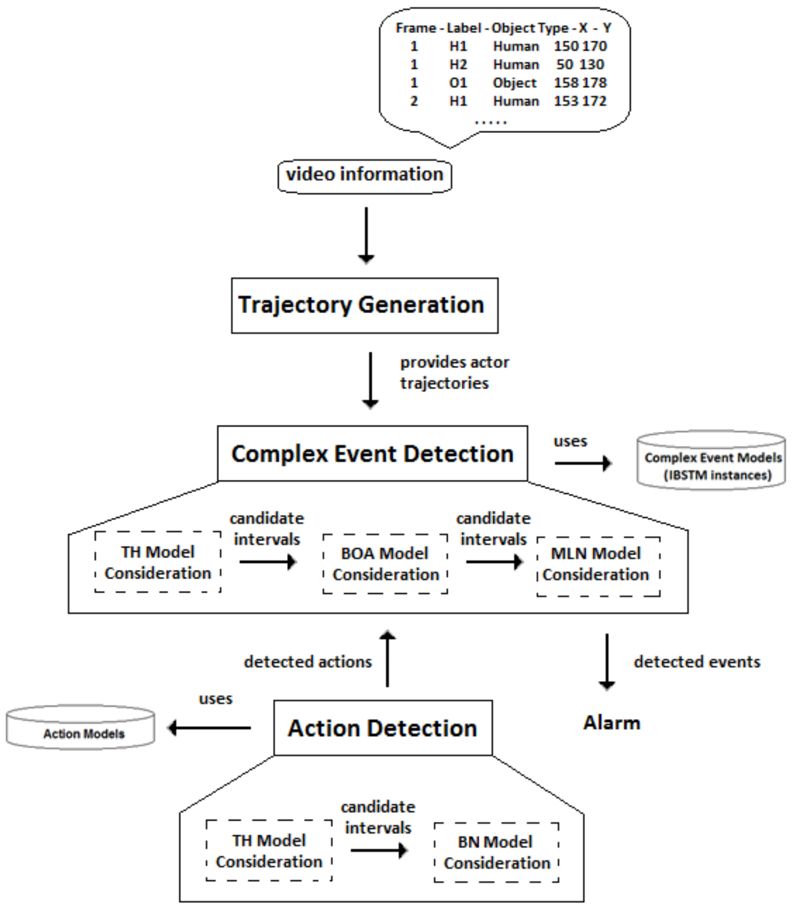


Figure 7 Event Detection Process

In Event Detection Process (Figure 7), actions and complex events are searched in test videos by using learned models. In the first step, the video information is parsed and actor trajectories are generated using Trajectory Generation Module. Complex event detection is performed for each complex event type that is defined in the system. Alarms are generated for the detected events.

SVAS has a high performance event detection capability due to its interval-based hierarchical manner. In Figure 7, both Action Detection and Complex Event Detection consider more than one model. SVAS determines related candidate intervals for each main model and uses the related main model when needed rather than using all models. Particularly, the proposed Threshold Model method is first used in both detection types because of its high-performance capability. In the following sections, these processes are discussed in detail.

3.2 Trajectory Generation

Trajectory Generation generates actor trajectories from input videos for further processing. In this study, CAVIAR [115] and BEHAVE [16] datasets are used. Since these datasets are in XML format, it is not necessary to use low-level operations such as background subtraction, object detection, object recognition or object tracking. However, low-level processing is necessary for raw videos. SVAS Trajectory Generation Module can be integrated with any low-level study available. The overall trajectory generation process is shown in Figure 8:

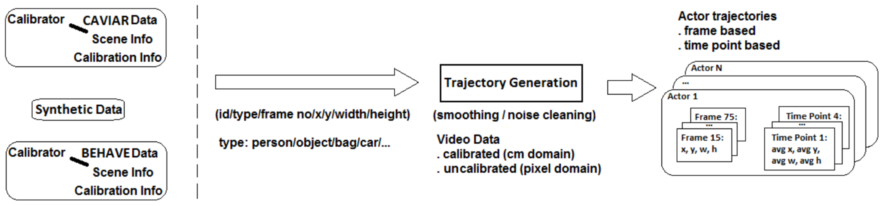


Figure 8 Trajectory Generation

In the first operation, text data input is parsed and a list of tuples which are in the form of <frame no, id no, coordinates (x, y), width, height, type> is obtained. These attributes represent the following:

• frame no: current frame number

• id no: label of the bounding box

• x: x coordinate of the center of the bounding box

• y: y coordinate of the center of the bounding box

• width: width of the bounding box

• height: height of the bounding box

• type: type of the object in the bounding box (car, bag, object or person)

Since the parsing operation depends on video data, a different parser is needed for each video input type. When the input enters the system in this format, the video data is prepared as actor trajectories for processing. Frame-based trajectories of actors are calculated according to the items in the input data. SVAS has cleaning and smoothing capabilities in video data. Noise elimination and smoothing are carried out in this process. The input is analyzed to determine occlusion areas such as columns. Missing trajectories resulting from occlusion areas are assembled. Coordinates of each object and person in 15 consecutive frames are grouped and their arithmetic average is calculated to eliminate noises and obtain smooth trajectories. This grouping operation can be considered as the creation of time-based trajectories. Time-based trajectories per actor also provide greater efficiency for the consideration of some features. The number of items that is considered in calculations is decreased by one fifteenth. 15 consecutive frames are equivalent to nearly 0.5 sec. The number 15 was selected for grouping frames since actions within 0.5 seconds are generally visible to the human eye. This indicates that some features such as speed and direction can be considered at 0.5 second intervals.

SVAS can generate the scene structure using trajectories. Movable and occlusion areas can be determined. However, if scene structure and context information are given, a more accurate scene model is obtained, which increases the success of SVAS. If scene information can be given as input by the user, this information can be used by the system to increase noise elimination. Occlusion and exit areas can be determined and can be used to update trajectories. Contextual information such as static features may improve the event recognition performance. SVAS enables the user to define rules and specific areas such as forbidden zones in the scene. A detailed scene structure (e.g., roads, paths, and entry and exit points) can help to solve many problems and to minimize errors that come up from the low-level operations. For example if exit points are known; when an actor lost in one of those points, it is inferred that he exits. Otherwise this situation causes tracking problems. In addition, detailed scene structure allows high level inferences. For example if the bus stop area is known; for an actor in that area it is inferred that he waits for a bus.

Trajectory Generation Module also has a basic calibration capability. The camera, which captures the video input data, is not always necessarily located at the center top of the scene. The location of the camera causes perspective problems in which the same size actors and the same movement changes are not measured as the same in various parts of the scene. Calibration process is dependent on video data like a parsing operation. Most of the video input providers give calibration data additionally. This data contains information about some positions in which pixel values and distance values are given. The number of calibration points increases accuracy level. Trajectory Generation Module can calculate the distance value of any pixel position. As a result, trajectories are calculated as if they are calculated from a camera which is located at the center top of the scene. The unit of position values is converted to distance units such as centimeters. Higher levels of the proposed system are independent of units. The system can work with both pixel and centimeter units.

In CAVIAR Dataset, information of four points is given for calibration. Pixel and distance values of these calibration points are given as shown in Figure 9. Trajectories of actors are changed from pixel domain to distance domain as follows. For each pair of calibration points the distance of a pixel is calculated by proportion method. The distances are found in centimeters for two calibration points. The average of these values are calculated and determined as the distance of the pixel position.

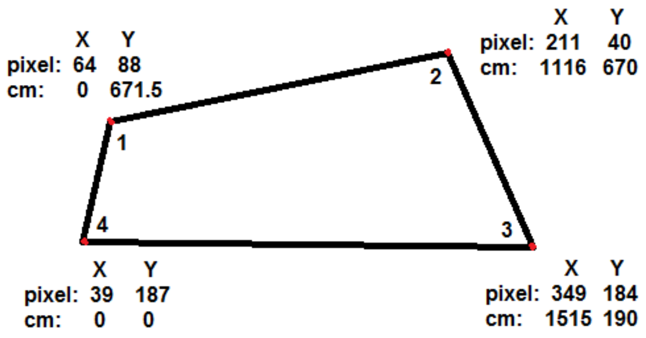


Figure 9 Calibration points of CAVIAR Dataset

Trajectory visualization is another application that provides visualization of trajectories of actors if debugging is needed. Trajectory visualization can be used for the evaluation of trajectories and determining inconsistencies. In Figure 10 illustrates a sample trajectory visualization:

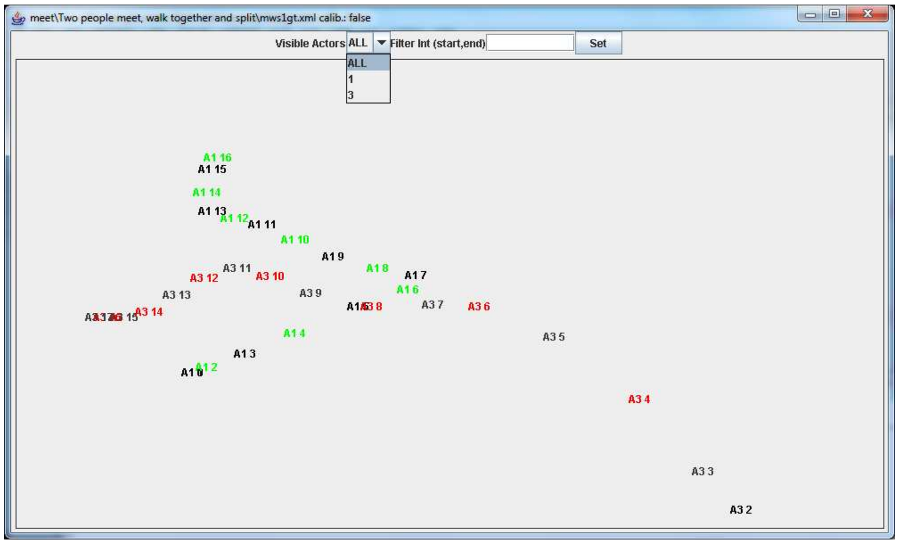


Figure 10 Sample Trajectory Visualization

3.3 Event Model Learning In SVAS

Event Model Learning is one of the most important processes in SVAS, in which models of Actions and Complex Events are learned using the training data. SVAS does not need any predefined thresholds for scene or event type contrary to many studies in the literature (e.g. [97], [5], [68], [83], [62] and [91]). It can learn required thresholds as Threshold Models.

In this section, Event Model Learning is presented in detail (see Figure 6). The learning capability of SVAS is a kind of supervised learning in which labeled training data is used for specific events and actions. In Action Model Learning, the input is the labeled training data. In Complex Event Model Learning, the inputs are pre-learned action models and the labeled training data. Since closed world assumption is used for both of the learning operations in SVAS, learning only requires positive examples.

3.3.1 Action Model Learning

“Stand”, “Walk”, “Run” and “Instant Move” are examples of actions that can be learned and detected in the proposed system. These four are basic actions which must be detected for targeted complex events in the surveillance domain. However, note that proposed algorithms are mostly independent of action types and users can train and define new action types in the system.

It is not necessary to define semantically understandable models for actions because of their simple structure. For this reason, in SVAS, actions are modeled as a combination of Threshold Models and a Bayesian Networks Model, instead of using high-level predicates.

The Action Model Learning is done by the Action Model Learning Module. The training data for each action type is a set of tuples in the form of video file, an actor performing the action and action interval. The Action Model Learning Module takes this input and for each tuple it generates trajectories of the given actor in each interval using the Trajectory Generation Module. A movement analysis is carried out for each trajectory. It is important to note that features are prepared for both time-based trajectories and frame-based trajectories. Time-based trajectories improve efficiency in the raw testing phase, which helps eliminate irrelevant intervals.

In Action Model Learner four key features, which reflect the action, are calculated for actor trajectories. These features are as follows:

1. *Move Change*: It defines the possible position change values for two position data. It also contains information about average speed information. The sum of all move change values (total traveled distance) is divided by interval length which gives the average speed.
2. *Size Change*: It defines the possible size change values for two position data.
3. *Average Distance*: It defines possible position change values between the first and the last positions. The distance change in the interval is divided by the interval length which gives the average distance.
4. *Direction Change*: It defines the possible direction change degree values throughout the interval.

The direction of the actor is not actually known since it is necessary to recognize the front of an actor to detect the actual direction. If this information comes from the low level, then the system can consider the relevant information. Otherwise, the direction of the movement is determined and the direction change degree is calculated as follows: First, the angle between two consecutive position data is calculated. Then, the direction is determined according to the angle which is illustrated in Figure 11. The direction information is prepared for slices of 45 degrees. For example, the angle between 337.5 ^{o} and 22.5 ^{o} is considered as North. Then for each pair of consecutive direction information, direction similarity degree, which is a value between 0 and 1, is calculated. The direction similarity degree is calculated for two directions by considering their closeness. For instance, the similarity degree for a North direction value is shown in Figure 12.

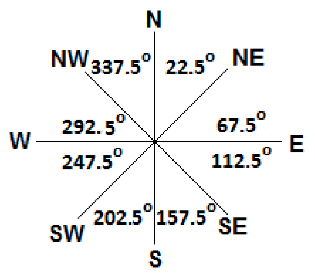


Figure 11 Direction Information

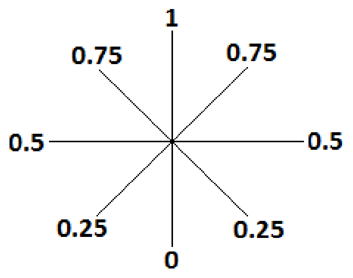


Figure 12 Direction Similarity Degree

For each consecutive position in the interval, a direction change value is calculated by subtracting direction similarity degree from 1. Finally, averages of these direction change values are calculated as the direction change degree. If low levels can provide new features, they can be considered too.

Threshold Models are proposed to reflect the motion analysis of an event by modeling basic trajectory features. A Threshold Model is created for each of these four feature types and for each of trajectory types (frame-based and time-based), during the action learning process. As a result of the action learning process, eight Threshold Models are created for each action type according to feature values in the training data. These steps are summarized in a flow diagram in Figure 13.

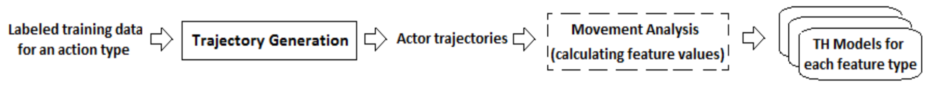

Labeled training data for an action type
Trajectory Generation
Actor trajectories
Movement Analysis (calculating feature values)
TH Models for each feature type

Figure 13 Threshold Model (TH Model) Learning

A Threshold Model consists of a proportion model and a frequency table. After all training data is analyzed, the proportion model is created by considering min3/4, min, average, max and max5/4 values of training data as in Figure 14. The 1/4 buffers for min and max values are used for giving a chance for border values in the detection phase. The proportion model generates weight values between 0 and 1 for the given test data according to similarity. The proportion model is not Gaussian since this distribution cannot be symmetric around the average value. This model is simple and efficient. However, frequencies should also be considered to prevent erroneous results. For example, many movement values are nearly 0 for Stand Action. The average is also close to 0, let’s say 0.2. Assume that the maximum value is 9 in the training data. In this case, the proportion method produces the same result for the intervals (0 - 0.2) and (0.2 - 9). 9 is less frequent and 0 is more frequent. Test data with 0 or 9 gets the same weight value which is 0.5. This behavior is not acceptable for surveillance domain. To solve this problem, frequencies of values in the training data are also kept in a table. Since data type is double and has continuous values, discretization is realized. Values are rounded to integer values to prevent infinite rows in the frequency table. To avoid eliminating small values, values are multiplied by 10 before the round operation. Values (multiplied by 10 and then rounded) and their frequencies are kept in a frequency table as shown in Figure 15. It is also erroneous to take only the frequencies into consideration for values which are less frequent but close to the average. As a result, both proportion model and frequency table are needed for a correct and efficient computation.

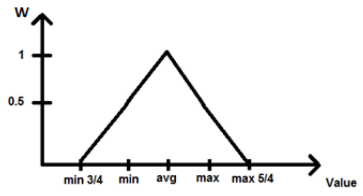
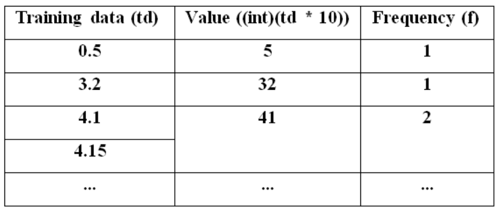


Figure 14 Proportion Model



Training data (td)	Value ((int)(td * 10))	Frequency (f)
0.5	5	1
3.2	32	1
4.1	41	2
4.15		
...	...	...

Figure 15 Frequency Table Method

In this way, all training data can be considered during evaluation. For each feature, the average, minimum and maximum values in the training data and their frequencies are considered as the Threshold Model. In detection phase, the Threshold Model can generate a weight value between 0 and 1 for a test data according to the similarity between the test data and the learned feature model by using two weight calculators. One of them calculates the weight using the proportion model while the other one calculates the weight using the frequency table. Weight calculation using the proportion model is done for a test value given in a test interval by considering regions in the model. As a result, a value between 0 and 1 is generated according to the proportion in regions. For example, if the test value is between avg. and max. values, the weight value is calculated as follows:

$$W_{proportion} = 1 - \left[(TestValue - avg)/(\max - avg)\right] * 0.5 \qquad (3.1)$$

where $W_{proportion}$ is weight value from the proportion method.

Weight calculation using the frequency table is done for a given test value which is in min.- max. interval. The weight value is calculated as follows:

$$W_{frequency} = 0.5 + \left(Value_{frequency}/2 * TotalFrequencyCount\right) \qquad (3.2)$$

where $Value_{frequency}$ is FrequencyValueOfTestValue which is obtained from the frequency table by casting 10 * test value to an integer and $W_{frequency}$ is Weight value from frequency method.

A value between 0.5 and 1 is obtained with this formula. If the test data is not in min.- max. interval, the result becomes 0 and a value between 0.5 and 1 is created by considering the frequency. If the test value is not one of the training data but it is in the min. - max. interval, its frequency becomes 0. To handle this erroneous state, 2-unit close neighbor value frequency is considered.

When the weight of the test data for a feature is required, both calculators are used to calculate a weight value between 0 and 1 and their maximum is chosen as a result.

$$WeightOfTestValue = Max(W_{proportion}, W_{frequency}) \tag{3.3}$$

After learning each Threshold Model for each feature type (currently 8), this set of Threshold Models is kept as a part of the action model which shows raw motion of an action. This set of feature models provides quick elimination of irrelevant intervals. Threshold Models provide great efficiency in the action detection phase. In addition, they are used to eliminate predefined threshold values in the system.

An action model is not only a composition of Threshold Models but also Bayesian Networks. After Threshold Models are learned, a Bayesian Network is learned with the same features, for actions defined in the system. The detailed action model is kept as a Bayesian Network.

As a result; Action Model Learning Module generates action models as Threshold Models for each action type and a Bayesian Networks Model as previously described in Figure 6.

3.3.2 Action Detection

Action Detection is used in both complex event learning and complex event detection operations. In action detection, actor trajectories and features are generated for test data. Similarities between test data and each pre-learned action model are calculated. Similarity calculation is done in two steps to increase efficiency, as shown in Figure 16.

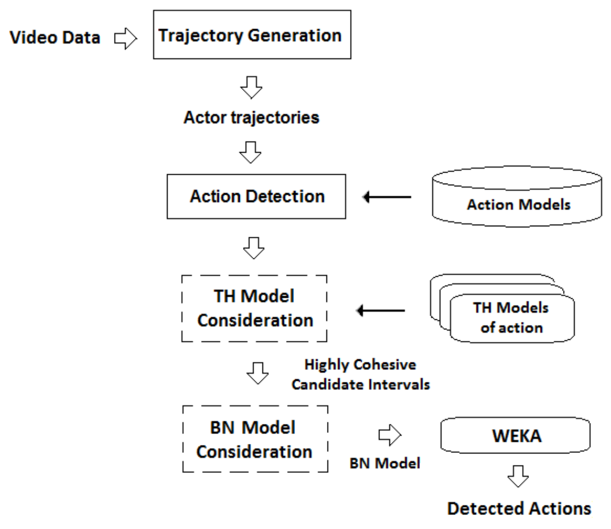


Figure 16 Action Detection

In the first step, candidate event intervals are determined using the pre-learned Threshold Model set and irrelevant intervals are eliminated. Video intervals which are suitable for Threshold Models are prepared as candidate intervals. In the second step, a detailed Bayesian Network analysis is done for candidate intervals using pre-learned Bayesian Networks Model. Pseudocode of this operation is shown below:

Algorithm 3: Action Detection

```
INPUT: V: part of the video
    1: timepoints ← TrajetoryGenerator creates using V
    2: candidateInts ← {}
    3: for all a ∈ action types do
    4:     candidateInts ← findCandidateIntervals using THModels and timepoints
    5: end for
    6: testInputForBN ← generateBNTestData using candidateInts
    7: BNConsideration using testInputForBN and BNModel
    8: d ← ParseBNResults
    9: return d
OUTPUT: d: DetectedActions
```

In detection phase, three seconds long video parts with 0.5 second overlaps are taken into consideration. First, time points of video parts are considered. For these time points, Trajectory Generation Module prepares trajectories and calculates feature values for feature types defined in the system [Pseudocode: 1]. For each action type defined in the system, candidate intervals are found and collected in a list [Pseudocode: 2-5]. The second main step of action detection process is a detailed analysis of candidate intervals using pre-learned Bayesian Network Model [Pseudocode: 6-7]. Bayesian Network evaluation algorithm is used in this step. Bayesian Networks inference, which has the highest value for an interval, is chosen as the detected action for that interval. The result of Bayesian Network evaluation algorithm is parsed and detected actions are determined [Pseudocode: 8-9].

Determining candidate intervals using “findCandidateIntervals” method provides great efficiency. It is not necessary to run Bayesian Network analysis for each time point. Pseudocode of this method is shown below:

Algorithm 4: findCandidateIntervals

```
INPUT: thresholdModelSet, timepoints
1:  candidateTimePoints ← {}
2:  for all tp ∈ timepoints do
3:      weightvalue  ← Ø
4:      featureCount ← Ø
5:      for all f ∈ feature types do
6:          thresholdModel ← get related THModel from thresholdModelSet using f
7:          if thresholdModel is time-point feature then
8:              weightvalue += calculate similarity value using thresholdModel and tp
9:              featureCount++
10:         end if
11:     end for
12:     weightvalue ← weightvalue / featureCount;
13:     if weightvalue > 0.4 then
14:         candidateTimePoints ← tp
15:     end if
16: end for
17: intervals ← generate all intervals using candidateTimePoints
18: candidateIntervals ← {}
19: for all inter ∈ intervals do
20:     weightvalue  ← Ø
21:     featureCount ← Ø
22:     for all f ∈ feature types do
23:         thresholdModel ← get related THModel from thresholdModelSet using f
24:         weightvalue += calculate similarity value using thresholdModel and inter
25:         featureCount++
26:     end for
27:     weightvalue ← weightvalue / featureCount;
28:     if weightvalue > 0.4 then
29:         candidateIntervals ← inter
30:     end if
31: end for
32: candidateIntervals ← prepare unions and intersections using intervals
33: return candidateIntervals
OUTPUT: candidateIntervals
```

The inputs of the method "findCandidateIntervals" are time points and an action Threshold Model; and its output is the set of candidate intervals. This operation is done in two steps. In the first step [Pseudocode: 1-16]; weight values are calculated for each time point, using pre-learned action models of each action type. Only time point related features are considered since the calculation is done for the time point. For example, “Average Distance” feature is not considered because this feature is calculated for the interval. The calculated values which are less than 0.4 are eliminated. The remaining ones are considered as candidate time points for detailed examination. Consecutive candidate time points are merged and intervals are created [Pseudocode: 17]. This operation generates all sub-interval combinations to obtain highly cohesive intervals. Highly cohesive interval method is used to find the best matching intervals. The best intervals are intervals in which the weight values of the detected event are maximum. For example, there may be an interval t4-t14 with a probability of 0.5 for event A. However, in this interval, there may be a highly cohesive sub-interval t7-t10 with a probability of 0.7. To find these highly cohesive intervals, candidate consecutive time points are extracted to generate all sub-interval combinations. For example, for the interval t1-t3; intervals t1-t2, t1-t3 and t2-t3 are generated and added to the interval list. In the second step of the method “findCandidateIntervals” [Pseudocode: 18-31], weight values for the generated intervals are calculated using pre-learned action models of each action type. In this case, all feature types including interval-based features are considered. A value between 0 and 1 is generated for each feature type. The averages of the generated weight values are calculated. Intervals for which weight values are greater than 0.4 are selected as candidate intervals. Then, these candidate intervals are processed further to reduce intervals containing the same action type. [Pseudocode: 32]. When one interval contains the other, they can be updated in two possible ways: if the interval containing the other has higher detection value, then sub-interval is eliminated from the result set. Otherwise, the contained subinterval is kept and the interval which contains the sub-interval, is replaced with two separate subintervals which do not intersect with the contained sub-interval. These sub-intervals are added to the result set. If no interval contains the other but there is an intersection between them then the intersection is determined and new interval is added to the result set for this intersection. The start and the end time of other two intervals are updated. The weight value of the new interval is determined by the maximum weight value of the two intersecting intervals. In the last step of the algorithm, two consecutive intervals with the same action type are replaced with a new joined interval. The weight value of the new interval is calculated according to the weights of intervals by considering their lengths such that:

```
newWeight = (occuredEvent_i.weight * occuredEvent_i.intervallength + occuredEvent_j.weight *
occuredEvent_j.intervallength) / (occuredEvent_i.intervallength + occuredEvent_j.intervallength)
```

As the result, highly cohesive intervals for detected actions are determined as candidate intervals and returned for detailed Bayesian Network Model analysis [Pseudocode: 33].

3.3.3 Complex Event Model Learning

In Complex Event Model Learning, complex event models are generated automatically. In this process two techniques are proposed. In the first method, non-interval-based complex event model learning using Markov Logic Networks is proposed. Second method is more robust method in which complex event models are generated as IBSTM models which reflect spatio-temporal relations and Threshold Models of events according to the training data. In SVAS, Complex Event Models can be learned without any predefined values or thresholds. Complex events such as “Meet”, “Fight”, “Walk Together”, “Run Together”, “Follow”, “Chase”, “Left Object”, “Taken Object” are examples of events that the proposed system can learn and detect. The proposed algorithms are independent of complex event types and the user can train and define new complex event types in the system.

In the following sections, first, non-interval-based complex event model learning method is described, then the properties of our Interval-based Spatio-Temporal Model are described; finally the model learning process is presented.

3.3.3.1 Learning Non-Interval-Based Complex Event Models Using Markov Logic Networks

This method is non-interval-based event model learning method in which Markov Logic Network is directly used. Detected actions are considered as a set of predicates. In this method, a set of predicate types is introduced which define basic spatio-temporal relations and interactions between objects and people in the videos. A set of policies to choose the appropriate predicates is proposed for event learning. First, the video data is converted to a set of Markov Logic Network (MLN) predicates. Then, these policies, together with the discriminative weight learning algorithm, are used to infer the relevance of the predicates to the events being queried. The relevant spatio-temporal rules are learned by using the discriminative weight learning algorithm and proposed methodology which contains a set of policies. Finally, the event model is generated.

First, 1) attributes of objects, 2) attributes of people, 3) spatio-temporal relations between objects, 4) spatio-temporal relations between people, 5) spatio-temporal relations between objects and people are considered and logical predicates are generated. In order to reflect these basic attributes and relations of objects and people, predicates that are generated in this level are grouped as follows:

1. Time based predicates, which are related to only one object or one person: *existPerson*, *existObject*, *stopPerson*, *stopObject*.

2. Time based predicates, which show relations between two objects, two people and one object-one person: *closeDistanceOO*, *closeDistancePP*, *closeDistancePO*.

3. Predicate that shows attributes of only one object or only one person: *smallObject*.

4. Predicate that shows relations between two objects, two people and one object-one person: *ownedBy*.

5. Predicates that show temporal relations: *during*, *before*, *after*.

Meanings of predicates are listed as follows:

– *existPerson(person,time)* (or *existObject(object,time)*): The person (or object) (parameter 1) is visible at that time (parameter 2).

– *stopPerson(person, time)* (or *stopObject(object,time)*): *x, y* coordinates of the person (or object) does not change according to the predetermined threshold for that time (parameter 2).

- *closedistancePO(person,object,time)*: A person and an object are assumed to be close if the distance between them is less than a predetermined pixel value. (*closedistancePP* and *closedistanceOO* are similar to *closedistancePO*).

- *smallObject(object)*: Object is small (height and width values of the object is considered).

- *ownedBy(person, object)*: When an object appears, the nearest person to that object is considered to be the owner of that object. Nearest value is again a predetermined threshold.

- *during(time1,time2)*: The interval between *time1* and *time2* is less than a predetermined threshold value.

- *before(time1,time2)*: *time1* is less than *time2*.

- after *(time1,time2)*: *time2* is less than *time1*.

All predicates that are found in this level are written in a file. This file is the predicate form of the video.

According to a set of policies, the information from the action detection process is queried by using discriminative weight learning algorithm in order to find the predicates relevant to a target event. The relevant predicates compose the event model.

In order to learn the model automatically, a methodology that contains a set of policies is proposed. After the user defines the target event for which the model will be found, the predicates are considered whether they are relevant to the event or not.

Policy 1: The relation between the event and the predicates, which have common parameters with the event, is considered.

There is no need to consider irrelevant predicates, which contain any parameter that is not in the event parameters. This will also improve the performance. For example, if the target event has an object parameter type such as *Event(obj1,…)* then predicates that have object parameter type such as $P_1(obj1)$ is considered. $P_2(person1)$ is not considered because person1 is not in the event parameters. We call P_1 as a candidate predicate for this example.

Policy 2: The event predicates contain two time parameters. One of them shows that event does not occur at that time, the other one shows that event occurs at that time. By this way, effects of predicates are considered according to these time values. For this reason, $E(\ldots,t_n, t_{n+1})$ is an event which occurs while time changes from t_n to t_{n+1}.

We can define all possible relations between predicates and events as follows. Let us take *P* as a predicate which we try to determine whether it is relevant to the event or not (i.e. it affects the event or not). Let us take *E* as the event predicate. There are four possible logical relations between them.

The weights of the relations are as follows (‘$\rightarrow$’ states ‘implies’ operator):

A: Weight of ($E \rightarrow P$) [or weight of ($!P \rightarrow !E$)]

B: Weight of ($!E \rightarrow P$) [or weight of ($!P \rightarrow E$)]

C: Weight of ($E \rightarrow !P$) [or weight of ($P \rightarrow !E$)]

D: Weight of ($!E \rightarrow !P$) [or weight of ($P \rightarrow E$)]

The values of A, B, C and D are calculated by using discriminative weight learning method. The model is generated according to these values. In the following, we introduce some more policies about the values of A, B, C and D.

Policy 3: If the values of A and D are high then the predicate *P* affects *E* alone (*P* is very important for *E* to occur). $P \rightarrow E$ is added as a new model of the event.

Policy 4: If the value of A is high but the value of D is low then *P* affects *E* with some other predicates (*P* is important for *E* to occur). *P* is added by using "AND" operator to the model of event in which important predicates are contained.

Policy 5: If the value of B and the value of C are high then, *!P* affects *E* alone (*!P* is very important for *E* to occur). $!P \rightarrow E$ is added as a new model of the event.

Policy 6: If the value of C is high, then *!P* affects *E* with some other predicates (*!P* is important for *E* to occur). *!P* is added by using AND operator to the model of the event in which important predicates are contained.

The value of A and the value of C cannot be high together because they have opposite meanings. If both values are low, it is inferred that the predicate (*P*) does not have any affect for the event (i.e. it is irrelevant). B and D are not considered alone to make a decision. For example, if only the value of D is high, then it is inferred that the predicate is irrelevant because the value of A is low (when event occurs, the predicate may not be true). This means that there are states such that *!P* but *E*. In addition, low value of D does not show that the predicate is irrelevant. Perhaps the predicate is relevant but it can affect the event only with other predicates. As a result it is inferred that without high value of A or high value of C, there is no need to consider values of D or B. So new policies can be added to improve performance:

Policy 7: If the value of A is low, there is no need to consider D.

Policy 8: If the value of C is low, there is no need to consider B.

As a result; for each of the very important predicates (both values of A and D are high or both values of C and B are high), new model such as $P \rightarrow E$ or $!P \rightarrow E$ is added. For all important predicates only one model such as P_1 *AND* P_2 *AND* P_3 … *AND* $P_N \rightarrow E$ is created.

Predicates can be time dependent or time independent. So, new policies can be added:

Policy 9: Time independent predicates are valid throughout the video.

Policy 10: Time dependent predicates cannot affect the events occurring at a time earlier than the time of the predicate time.

This policy can improve performance. The effect of the candidate predicates is queried by using discriminative weight learning algorithm and by considering all these 10 policies in order to find the complex event model.

The weight learning in MLNs is performed by optimizing a likelihood function, which is a statistical measure of how well the probabilistic model fits the training data. The weights are learned by discriminative estimation. Discriminative learning attempts to optimize the conditional distribution of a set of outputs, given a set of inputs. Training videos are also given to as a database of facts. The weights of A, B, C and D are considered as their values and according to the policies relevant predicates are determined and an event model is generated. After the creation of the complex event model, it is proposed to the user. Since MLN is FOL based, the user can understand the proposed event model and edit it easily.

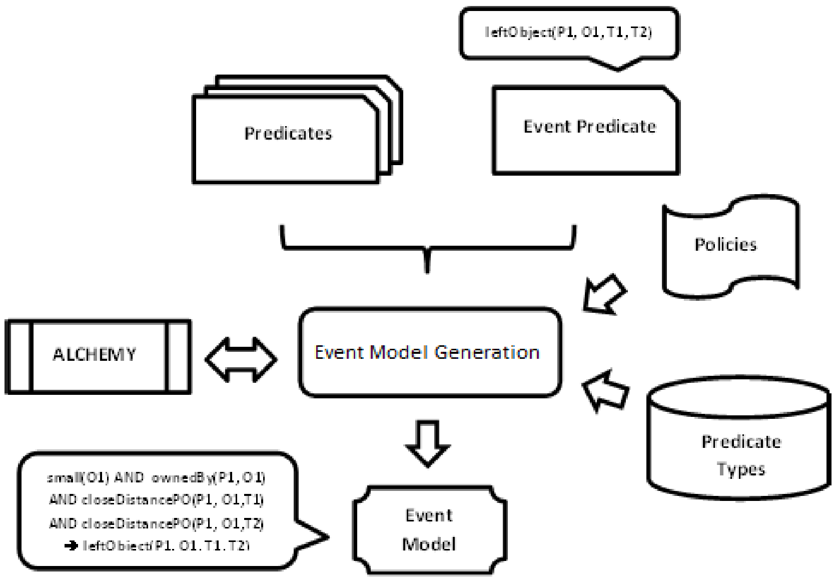

leftObject(P1, O1, T1, T2)
Predicates
Event Predicate
Policies
ALCHEMY
Event Model Generation
Predicate Types
small(O1) AND ownedBy(P1, O1) AND closeDistancePO(P1, O1,T1) AND closeDistancePO(P1, O1,T2) ➔ leftObject(P1, O1, T1, T2)
Event Model

Figure 17 Non-Interval-Based Complex Event Model Generation

All operations that are fulfilled in this process are summarized in Figure 17.

The predicate form of the videos is obtained from the action detection process. By using the defined policies and predicate types, Alchemy is queried and the event model is generated.

3.3.3.2 Interval-Based Spatio-Temporal Model (IBSTM)

IBSTM is a hybrid event model which meets the requirements of an event model that are described in Sections 1 and 2. IBSTM fills the semantic gap between humans and video systems by providing the following basic properties.

1. *Domain convenience:*

IBSTM is suitable for surveillance domain. Various events with different scenarios can be learned and defined in IBSTM. IBSTM can model and recognize similar events such as “fight” and “meet” which have similar logical predicates at high level, but their behavior is quite different at low level. In addition, complex events can be learned in a role-based manner. Properties of each actor can be learned independently, e.g. for fight event, an actor who hits a person or an actor who is hit by a person can be differentiated. IBSTM is scale invariant; there is no limitation on event durations. Spatio-temporal relations can be defined in IBSTM. The basic temporal relations of Allen's Interval Logic [3] are defined in IBSTM. Temporal relations defined in Allen's Interval Logic relations are shown in Table 1.

Time variables can be point or interval based. Basic spatial relations are defined in IBSTM. Distance relations, namely "close", "far", "disconnected" and "touch", topological relations such as "inside" and "outside", and position relations "near", "in front of" and "left of" can be defined. In addition, IBSTM can be used in both calibrated and uncalibrated scenes since the proposed learning and detection algorithms are independent of units such as pixel or centimeter.

Table 1 Temporal relations defined in AIL relations

Relation Name	**Representation using X and Y**
Before	$\underline{X}$ $\underline{Y}$
Overlaps	$\underline{xxXxx}$ $\underline{yyYyy}$
Meets	$\underline{X.Y}$
Equal	$\underline{X}$ $\underline{Y}$
Starts	$\underline{X}$ $\underline{yyYyy}$
During	$\underline{X}$ $\underline{yyYyy}$
Finishes	$\underline{X}$ $\underline{yyYyy}$

2. *Uncertainty handling:*

The input of the IBSTM which comes from low levels can be noisy due to many problems such as the quality of low-level algorithms, structure and complexity of the video scene, camera problems, illumination changes, segmentation issues, occlusions, and tracking and detection problems. These problems may affect the recognition accuracy. Uncertainty can be modeled in IBSTM to minimize errors in noisy conditions. Event models are learned with weights in the training phase. Inference and detection algorithms generate probabilistic results to prevent discretization problems. The proposed "Threshold Model" is also probabilistic which covers all training data for a feature in an efficient way. IBSTM can manage probabilistic input data that comes from low levels. If low levels generate probabilistic data, these data can be used in learning and detection phases.

3. *Understandability*:

IBSTM is based on MLN. MLN model provides user understandable and manageable logic predicates. Generated event models are presented to the user as FOL rules. For this reason, the generated complex event model is semantically readable. This property enables the end user to control and manage the event model. The user can interfere the model if desired. Also if training data is not available, the user can create his/her own event models by using SVAS User Interface Module.

4. *Performance*:

IBSTM model consists of three main models: Threshold Models, BoA model and MLN model. MLN has performance deficiencies in video domain. However, Threshold Models and BoA Models provide great efficiency in both action and complex event detection by eliminating irrelevant intervals. These two models are integrated with MLN to increase the performance of MLN in video domain. As a result, only candidate intervals are queried by MLN.

3.3.3.3 Complex Event Model Learning Process

In Complex Event Model Learning, the inputs are pre-learned action models and the training data for complex events (see Figure 6). Complex Event Model Learning Module takes video data, actors, and intervals as training data for each complex event type. First, the video data is prepared by using the Trajectory Generation Module. Movement analysis is done for each actor so that role-based event model can be learned. For instance, in a "chasing" event, one actor can be learned as the chaser while the other can be learned as the one who is chased.

Threshold Models are prepared for each actor for each feature type used in action models. Spatial relations like distance (closeness) and direction similarity degree between actors are also learned as Threshold Models. Distance Threshold Model defines the spatial model which includes the possible distance values between two actors. Direction Similarity Degree Threshold Model defines the possible direction change degree values between two actors throughout the interval. Direction similarity degree feature is calculated in a way similar to the calculation of "directionChangeDegree" in action learning. For each time point, direction similarity between actors is determined by considering angles as in Figure 12.

In the second step, Complex Event Model Learning Module determines actions in training data using Action Detection Module (Figure 6). Detected actions are used in BoA models and MLN models. Pre-learned and predefined actions are detected by using highly cohesive intervals method as explained in previous sections. For detected actions, BoA Model is created. BoA Model is a kind of "Bag of Words" approach in which actions in the training data are found without considering temporal information, in order to increase detection performance.

BoA model reduces the search space in detection phase by eliminating intervals which are not suitable. Only suitable intervals are queried by MLN.

BoA model is also kept based on the roles of actors of the event. In the last step, MLN models are learned.

In MLN models, actor types, temporal relations, actions, spatial relations and properties can be defined. The predicates that are used in MLN model are described in Table 2.

Table 2 MLN Predicates

Description	**MLN Predicate**
Type	actor, timeint
Actor Type	Person, Object, Car
Actions	Stand(actor,timeint), Run(actor,timeint), InstantMove(actor,timeint), Walk(actor,timeint)
Temporal Relations	Before(timeint, timeint), Meets(timeint, timeint), Overlaps(timeint, timeint), Starts(timeint, timeint), Equal(timeint, timeint), During(timeint, timeint), Finishes(timeint, timeint)
Spatial Relations	Near(actor,actor,timeint), Far(actor,actor,timeint), Inside(actor,actor,timeint), Front(actor,actor, timeint), Rear(actor,actor, timeint), Left(actor,actor,timeint), Right(actor,actor,timeint), Top(actor,actor,timeint), Bottom(actor,actor,timeint), Outside(actor,actor,timeint), Touch(actor,actor,timeint), Disconnected(actor,actor,timeint), DirectionSimilar(actor,actor,timeint)
Properties	Small(actor), OwnedBy(actor,object)
Certain rules	!ComplexEvent(a1,a1,t), Meet(a1,a2,t1) => Meet(a2,a1,t1), Equal(t1, t2) => Equal(t2, t1)

MLN model is not created for all combinations of predicates. To increase performance MLN predicates and rules are created for only related (highly cohesive) intervals. By using this summarization method, the number of predicates decreases considerably. All predicates and rules are weighted and the MLN model is generated. These steps are summarized in a flow diagram in Figure 18.

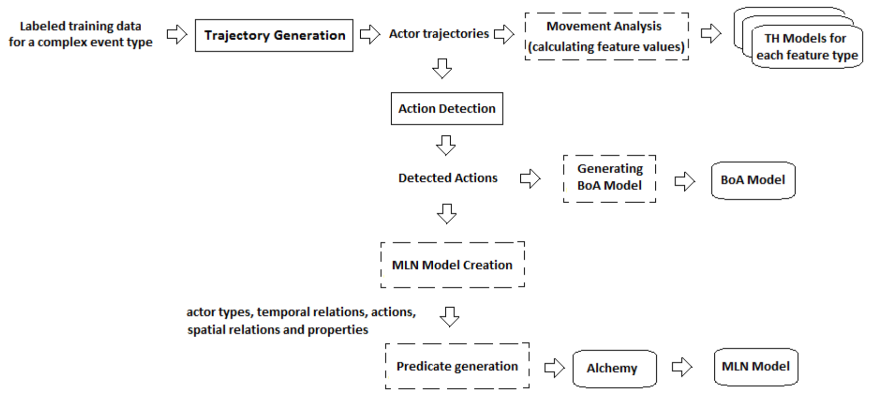

Labeled training data for a complex event type
Trajectory Generation
Actor trajectories
Movement Analysis (calculating feature values)
TH Models for each feature type
Action Detection
Detected Actions
Generating BoA Model
BoA Model
MLN Model Creation
actor types, temporal relations, actions, spatial relations and properties
Predicate generation
Alchemy
MLN Model

Figure 18 Complex Event Model Learning

At the end of the complex event model learning process, a complex event model is generated for each complex event type. A complex event model consists of 3 types of models: Threshold Models, a BoA Model and a MLN model. Such a combined model has efficient inference capabilities as demonstrated in Section 4.

3.4 Complex Event Detection In SVAS

Complex Event Detection is a process of finding complex events in a video using pre-learned IBSTM event models. Figure 19 shows the flow diagram of the operation.

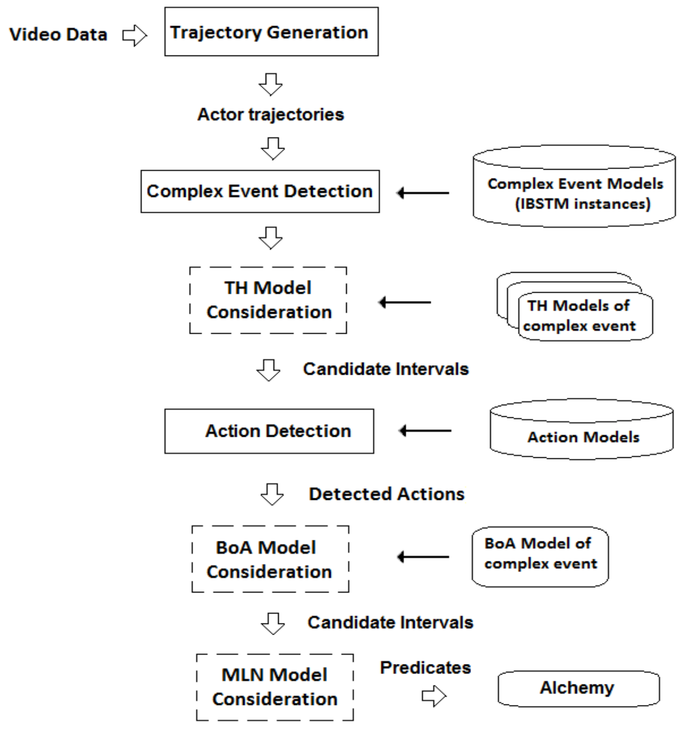


Figure 19 Complex Event Detection

Pseudocode of this operation is shown below:

Algorithm 5: Complex Event Detection

```
INPUT: V: part of the video
  1:  timepoints ⟵ TrajetoryGenerator creates using V
  2:  candidateInts ⟵ {}
  3:  predicates ⟵ {}
  4:  for all ce ∈ complex event types do
  5:      THModels ⟵ from ce
  6:      candidateInts ⟵ getCandidateIntervals using THModels and timepoints
  7:      actions ⟵ detectActions for candidateInts
  8:      candidateInts ⟵ interval elimination using actions and BoAModels
  9:      predicates ⟵ createMLNPredicates using actions for candidateInts
  10: end for
  11: f ⟵ createMLNFactFile using predicates
  12: MC-SAT using f and MLN Model
  13: d ⟵ parseMLNResults
  14: return d
OUTPUT: d: Detected Events
```

When the video is queried for event detection, the video data is parsed and actor trajectories are generated using Trajectory Generation Module [Pseudocode: 1]. First, candidate intervals are determined using Threshold Models of IBSTM [Pseudocode: 5-6]. Trajectories are analyzed for actors and detailed motion analysis is performed using pre-learned Threshold Models of an event. For each complex event type defined in the system, similarity of movement analysis is searched using Threshold Models of events. Inference process is executed in a hierarchical manner to increase performance. In this analysis, SVAS starts with the most distinguishing features. Features are considered according to their effect. For example, in video event which occurs between two actors, spatial features are considered first due to their high-level importance because spatio-temporal features between actors are the most discriminative features in complex events. SVAS does not try to find an event for those actors who are not in an acceptable closeness. Acceptable closeness of an event is learned during learning phase as the Threshold Model. Candidate intervals for actors are determined according to Threshold Model for spatial feature by considering spatial relations between actors. Following this process, the movement features of candidate actors are considered and detailed motion analysis is performed by considering other features. If Threshold Models give the similarity value greater than 0.4, then the candidate intervals are determined for those actors.

After Threshold Model analysis, action detection is done for candidate intervals using Action Detection Module [Pseudocode: 7]. Action Detection Module queries candidate intervals and tries to detect actions using pre-learned action models. Actions residing in the candidate intervals are determined by highly cohesive intervals method in which sub-intervals with higher probabilistic values are searched in an interval by analyzing all sub-intervals. This method contains a set of interval operations such as division, intersection and union to find the best intervals that have higher weights as discussed in Section 3.3.2.

In the second phase of the detection, the BoA model is applied to the candidate intervals [Pseudocode: 8]. It provides a quick elimination of unrelated partitions of the input data. Actions found in candidate intervals are compared with the actions of pre-learned BoA Model. While actions in candidate intervals are detected, they are queried in BoA Model and suitable intervals are prepared for detailed complex event detection. BoA Model generates a value between 0 and 1. The intervals, which are not suitable (i.e. generated value is less than 0.5) for BoA Model, are eliminated. This checking also increases the performance for intervals in which no complex event occurred.

In the last step, candidate intervals which contain suitable actions for BoA Model are queried using MLN model. Since SVAS is interval-based rather than time point based, the number of MLN predicates extremely decreases and minimum MLN graph is created. In addition, unrelated predicate sets are not given to MLN to prevent unnecessary operations. MLN predicates are created for only highly cohesive candidate intervals which reduces the number of variables [Pseudocode: 9]. As a result, the performance of MLN algorithm increases. For the remaining candidate intervals, MLN fact file is created [Pseudocode: 11]. A sample fact file includes predicates as follows:

```
Stand(A1,T1)

Stand(A2,T2)

Equal(T1,T3)

Equal(T2,T3)
```

The remaining candidate intervals are queried with MC-SAT algorithm by using MLN event models that are learned in the training phase and created MLN fact file. A sample result is as follows:

Meet(A1,A2,T2) 0.0250475

Meet(A1,A2,T3) 0.280022

Meet(A2,A1,T2) 0.040046

Meet(A2,A1,T3) 0.311019

Meet(A2,A2,T1) 0.0060494

Intervals in which MC-SAT algorithm gives higher results are considered as alerts for complex events [Pseudocode: 12-14].

3.5 Prediction

Event Prediction is the highest level in surveillance domain. The main goal is to predict events before they occur. SVAS proposes three simple methods for event prediction. As the first method, SVAS uses IBSTM models. In learning phase, SVAS can learn event model with its previous state and post state if training data has suitable features. The previous state is the state of event actors before the event occurs. The post state is the state of event actors after the event. SVAS can generate IBSTM models for not only the duration of the event but also durations before and after the event. SVAS generates pre-event IBSTM model and post-event IBSTM model for these two durations in the learning phase. Pre-event IBSTM model can be used for event prediction in detection phase.

The second method is using BoA models. In detection phase, SVAS uses BoA models of events and determines the number of occurred actions in BoA models. This calculation gives the number of sub actions of a complex event. The number of sub actions increases the prediction of complex events.

The third method considers the event detection values at runtime. After event detection generates the detection value, it is used to decide whether the event has occurred or not. If this value is not adequate for any event detection but it has a continuous increasing tendency for any event then this value is considered as the prediction value.

3.6 Implementation Details and a Sample Application

In this section, implementation details of SVAS and a sample application are presented. SVAS Application is implemented using Java Programming Language. IntelliJ IDEA Community Edition is used for IDE. The most important tools that are used in this thesis are WEKA [28], Alchemy and Tuffy. WEKA Library is used for running the required machine learning algorithms such as Bayesian Networks. Alchemy and Tuffy are used for running MLN algorithms.

The sample application is prepared in order to show some important features of the main processes of SVAS. Learning and inference capability of SVAS is presented using CAVIAR Dataset. Scene boundaries are calculated when training dataset is parsed. The output of this process is shown in Table 3.

Table 3 CAVIAR Dataset scene boundaries

Scene Values	Pixel Values
MINX	3
MINY	1
MAXX	321
MAXY	286

After scene boundaries calculated, action models are learned using the training dataset. In CAVIAR Dataset, four main actions ("running", "inactive", "walking" and "active") can be learned. The action “inactive” can be considered as a stand action and “active” can be considered as an instant move action. Action models are composed of threshold models and a Bayesian network model. Table 4 shows some Threshold Models of actions as a result of Action Learning.

Table 4 Threshold Models of CAVIAR Actions

Action	Threshold Model Name	Max Value	Average Value	Min Value	Frequency Values (Value - Frequency)
Running	*Move Change*	*8.0*	*1.68*	*0.0*	*0 - 45 , 10 - 105 , 14 - 42 , 20 - 55 , 22 - 39 , 28 - 19 , 30 - 13 , 32 - 13 , 36 - 2 , 40 - 4 , 41 - 4 , 42 - 1 , 54 - 1 , 58 - 1 , 63 - 2 , 73 - 1, 76 - 1 , 80 - 1*
	Size Change	*12.65*	*1.89*	*0.0*	*0 - 43 , 10 - 89 , 14 - 47 , 20 - 48 , 22 - 42 , 28 - 6 , 30 - 17 , 32 - 13 , 36 - 10 , 40 - 6 , 41 - 6 , 42 - 2 ,45- 4 , 50 - 7 , 51 - 4 , 60 - 1, 63 - 1 , 67 - 1, 90 - 1 , 126 - 1*
	Direction Change Degree	*1.0*	*0.28*	*0.0*	*0 - 125 , 3 - 101 , 5 - 69 , 8 - 26 , 10 - 13*
	Average Distance	*3.65*	*1.50*	*0.76*	*8 - 1 , 9 - 3 , 10 - 2 , 11 - 1, 12 - 2 , 14 - 1 , 18 - 1 , 22 - 1 , 23 - 1 , 24 - 1 , 36 - 1*
Inactive	*Move Change*	*0.0*	*0.0*	*0.0*	*0 - 375*
	Size Change	*0.0*	*0.0*	*0.0*	*0 - 375*
	Direction Change Degree	*0.0*	*0.0*	*0.0*	*0 - 360*
	Average Distance	*0.0*	*0.0*	*0.0*	*0 - 15*
Walking	*Move Change*	*4.24*	*0.83*	*0.0*	*0 - 143 , 10 - 133 , 14 - 40, 20 - 22 , 22 - 17 , 28 - 1 , 30 - 1 , 32 - 4 , 40 - 2 , 42 - 1*
	Size Change	*8.0*	*1.08*	*0.0*	*0 - 114 , 10 - 131 ,14 - 56, 20 - 20 , 22 - 17 , 28 - 3 , 30 - 5 , 32 - 3 , 36 - 2 , 40 - 3 , 41 - 2 , 42 - 1 , 50 - 1 , 51 - 1 , 54 - 1 , 57 - 1 , 58 - 1 , 70 - 1 , 80 - 1*
	Direction Change Degree	*1.0*	*0.25*	*0.0*	*0 - 163 , 3 - 76 , 5 - 74 , 8 - 20 , 10 - 16*
	Average Distance	*1.33*	*0.67*	*0.06*	*1 - 1 , 4 - 1 , 5 - 1 , 6 - 6 , 7 - 2 , 8 - 1 , 9 - 1 , 10 - 1 , 13 - 1*
Active	*Move Change*	*3.16*	*0.32*	*0.0*	*0 - 262 , 10 - 57 , 14 - 7 , 20 - 11 , 22 - 5 , 30 - 1 , 32 - 2*
	Size Change	*5.66*	*0.45*	*0.0*	*0 - 234 , 10 - 67 , 14 - 19 , 20 - 16 , 22 - 3 , 28 - 2 , 30 - 2 , 40 - 1 , 57 - 1*
	Direction Change Degree	*1.0*	*0.23*	*0.0*	*0 - 206 , 3 - 13 , 5 - 69 , 8 - 8 , 10 - 34*
	Average Distance	*0.38*	*0.19*	*0.0*	*0 - 1 , 1 - 5 , 2 - 4 , 3 - 4 , 4 - 1*

In addition to threshold models, Bayesian Model of actions is learned using WEKA tool. Training data file is prepared for WEKA using the same features. Attributes and classes are defined at the beginning of the file. After that training data values are given. Figure 20 shows part of the training file.

```
@relation action
@attribute moveChange numeric
@attribute sizeChange numeric
@attribute distance numeric
@attribute directionChange numeric
@attribute class {running,inactive,walking,active}
@data
1.28, 1.54, 1.20, 0.24, running
1.11, 1.17, 0.92, 0.29, running
1.12, 1.44, 0.94, 0.39, running
1.08, 1.25, 1.00, 0.24, running
                 …
```

Figure 20 Sample Training File for WEKA

Some of the parameters of the learned Bayesian Network Model are as follows:
class: 4
LogScore Bayes: -202.96185083886107
LogScore BDeu: -300.82134868085024
LogScore MDL: -305.04098870828847
LogScore ENTROPY: -217.01258062051335
LogScore AIC: -260.01258062051335

After action learning, SVAS is ready for action detection or complex event learning. For instance, complex event model of “walkingTogether” can be learned using the training data. The first step is Threshold Model learning. Same features in Threshold Models of actions are learned again per actor. In addition, Threshold Models for features related two actors are learned.

"walkingTogether" complex event is not role-based. Behavior of actors is similar. For this reason, only one Threshold Model set is kept for both actors. In role-based complex events, two Threshold Model sets are kept. Models for "walkingTogether" event are shown in Table 5.

Table 5 Threshold Models for Two Actors

Complex Event	Threshold Model Name	Max Value	Average Value	Min Value	Frequency Values (Value - Frequency)
walking together	*distance*	*79.05*	*30.98*	*5.38*	*54 - 2 , 76 - 1 , 82 - 1 , 85 - 1, 114 - 1 , 117 - 1 , 130 - 1 , 139 - 1 , 163 - 2 , 201 - 1 , 209 - 2 , 210 - 1 , 212 - 1 , 215 - 2 , 219 - 1 , 227 - 1 , 234 - 1 , 236 - 1 , 244 - 1 , 262 - 4 , 269 - 1 , 275 - 1 , 280 - 1 , 288 - 1 , 323 - 1 , 327 - 1 , 336 - 1 , 340 - 1 , 345 - 1 , 354 - 1 , 365 - 1 , 367 - 1 , 413 - 1 , 416 - 1 , 428 - 1 , 432 - 1 , 444 - 1 , 458 - 2 , 655 - 1 , 665 - 1 , 726 - 1 , 743 - 1 , 761 - 1 , 791 - 1*
	Direction Similarity Degree	*1.0*	*0.73*	*0.5*	*5 - 6 , 6 - 1 , 7 - 1 , 8 - 3 , 9 - 1 , 10 - 5*

In complex event learning process, the second main step is learning BoA Models. Detected actions are found for BoA Model learning and MLN Model learning using pre-learned action models. While BoA and MLN models are being constructed, actions are determined by action detection. Candidate intervals are found using Threshold Model analysis. Action detection also includes WEKA inference operation using pre-learned Bayesian Network model. WEKA test file is created for candidate intervals. The structure of this file is similar to the training file as shown in Figure 20. In the BoA model, detected actions are kept with weights. BoA Model of "walkingTogether" event is shown in Table 6.

Table 6 BoA Model of “walkingTogether” event

Is Role Based:	no
Actions	**Values**
running	*0.09*
inactive	*0.11*
walking	*0.67*
active	*0.13*

After BoA Model is constructed, MLN model for the complex event is constructed using Tuffy. First, the unweighted MLN file is prepared. Predicates are determined using detected actions. The generated unweighted MLN file named “walkingTogetherUnweighted.mln” for “walking Together” event is shown in Figure 21.

```
Person(actor)
running(actor,timeint)
inactive(actor,timeint)
walking(actor,timeint)
active(actor,timeint)
directionSimilar(actor,actor,timeint)
near (actor,actor,timeint)
walkingTogether(actor,actor,timeint)
!running(a1,t1) V walkingTogether(a1,a2,t1)
!inactive(a1,t1) V walkingTogether(a1,a2,t1)
!walking(a1,t1) V walkingTogether(a1,a2,t1)
!active(a1,t1) V walkingTogether(a1,a2,t1)
!running(a2,t1) V walkingTogether(a1,a2,t1)
!inactive(a2,t1) V  walkingTogether(a1,a2,t1)
!walking(a2,t1) V walkingTogether(a1,a2,t1)
!active(a2,t1) V walkingTogether(a1,a2,t1)
! directionSimilar (a1,a2,t1) V walkingTogether(a1,a2,t1)
! near(a1,a2,t1) V walkingTogether(a1,a2,t1)
! Person(a1) V walkingTogether(a1,a2,t1)
! Person(a2) V walkingTogether(a1,a2,t1)
```

Figure 21 walkingTogetherUnweighted.mln

Then the training fact file is created using detected actions. “walkingTogetherlearnDB.db” file is prepared for “Walking Together” event. Figure 22 shows part of the training file.

```
Person(A1) Person(A2) walking(A1,T1) walking(A2,T1)
directionSimilar (A1, A2, T1) near (A1, A2, T1)
walkingTogether (A1, A2, T1)
Person(A3) Person(A4) walking(A3,T2) walking(A4,T2)
directionSimilar (A3, A4, T2) near (A3, A4, T2)
 walkingTogether (A3, A4, T2)
Person(A5) Person(A6) inactive(A5,T3) active(A6,T3)
directionSimilar (A5, A6, T3) near (A5, A6, T3 )
 walkingTogether (A5, A6, T3)
Person(A7) Person(A8) inactive(A7,T4) active(A7,T4)
directionSimilar (A7, A8, T4) near (A7, A8, T4 )
walkingTogether (A7, A8, T4)
                    ...
```

Figure 22 walkingTogetherlearnDB.db

The query file is the last input file for Tuffy weight learning. For instance, “walkingTogetherquery.db” file is prepared for “Walking Together” event which contains *“walkingTogether(a1,a2, t1)”* as query. Then Tuffy weight learning algorithm is called. The Tuffy parameters for weight learning of “Walking Together” event is as follows:

```
-learnwt -i walkingTogetherunweighted.mln -e walkingTogetherlearnDB.db -
queryFile walkingTogetherquery.db -r weightedWalkingTogether.mln -mcsatSamples
10 -dMaxIter 100
```

The weighted MLN file for “walkingTogether” event is generated by discriminative weight learning algorithm as shown in Figure 23.

```
walking(actor,timeint)
running(actor,timeint)
walkingTogether(actor,actor,timeint)
inactive(actor,timeint)
active(actor,timeint)
Person(actor)
directionSimilar(actor,actor,timeint)
near (actor,actor,timeint)

0,2005    !running(v0, v1)  V  walkingTogether(v0, v2, v1)
0,1067    !inactive(v0, v1)  V  walkingTogether(v0, v2, v1)
0,8067    !walking(v0, v1)  V  walkingTogether(v0, v2, v1)
0,2135    !active(v0, v1)  V  walkingTogether(v0, v2, v1)
0,1935    !running(v0, v1)  V  walkingTogether(v2, v0, v1)
0,1303    !inactive(v0, v1)  V  walkingTogether(v2, v0, v1)
0,8567    !walking(v0, v1)  V  walkingTogether(v2, v0, v1)
0,1532    !active(v0, v1)  V  walkingTogether(v2, v0, v1)
0,7572    ! directionSimilar (v2, v0, v1)  V  walkingTogether(v2, v0, v1)
0,8566    ! near (v2, v0, v1)  V  walkingTogether(v2, v0, v1)
0,9574    ! Person (v0, v1)  V  walkingTogether(v2, v0, v1)
0,9574    ! Person (v2, v1)  V  walkingTogether(v2, v0, v1)
```

Figure 23 weightedWalkingTogether.mln

After learning is finished, event detection can be performed. A sample interval from CAVIAR Dataset is used for detection. This test data interval is not used in learning phase.

In event detection phase, the first operation is checking the similarities between Threshold Models and eliminating irrelevant intervals. Similarities between test interval and Threshold Models of complex events are considered. Similarity values of some complex events such as “Left Object” and “Fight” with the test interval are too low. For this reason, detailed analyses for these complex events are eliminated. However, Threshold Model Similarities of “Walking Together” and “Meeting” are high. The values are as follows:

"Walking Together" $\rightarrow$ *0.66*
"Meeting" $\rightarrow$ *0.65*

These two complex events can be considered as candidate complex events. Then, actions in the test data are determined for BoA and MLN model, using action detection processes of SVAS. BoA models of candidate complex event models are considered with those detected actions. The results of BoA model consideration is as follows:

"Walking Together" $\rightarrow$ 0.64
"Meeting" $\rightarrow$ 0.60

Both of the results are higher than 0.4. For this reason, MLN consideration is done for the candidate complex events. Evidence file named “evidence.db” is prepared for this operation, which includes detected actions in the test interval. The structure of evidence file is similar to training file as shown in Figure 22. Finally, MLN consideration is done by using Tuffy for both candidate events. Tuffy parameters for this operation is as follows:

```
-i weightedWalkingTogether.mln -e evidence.db -queryFile walkingTogetherquery.db
-r walkingTogetherinferout.txt
```

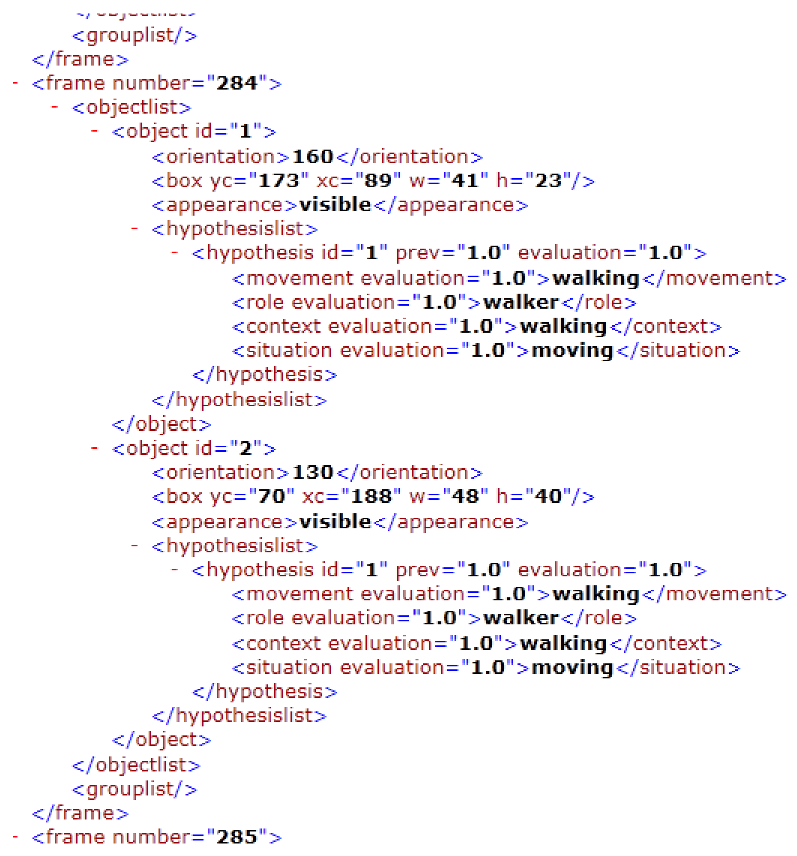
For this query, Tuffy writes results into a file named “walkingTogetherinferout.txt”. The results of inferences are considered for both of the candidate events and complex event which has maximum value is accepted as the detected event.

CHAPTER 4

EXPERIMENTS AND RESULTS

In this chapter, the experiments on the system are described and the results of the experiments are presented. The results of the proposed system are compared with the results of the related studies. The organization of the chapter is as follows: First, the evaluation of CAVIAR dataset is discussed in Section 4.1. Then the evaluation of BEHAVE dataset is discussed in Section 4.2. Both of the sections include evaluation of actions and complex events. In Section 4.3, the evaluation of synthetic dataset is presented. In this section, the tool which is developed for generating synthetic data is also introduced. Section 4.4 gives the results of performance evaluations. Qualitative evaluation is discussed in Section 4.5. Finally, in Section 4.6, Evaluation of Learning Non-Interval-Based Complex Event Models Using Markov Logic Networks is presented using CANTATA Dataset.

In this study, a series of experiments have been conducted and the proposed methods are evaluated using four datasets which are CAVIAR Dataset [115], BEHAVE Dataset [16], CANTATA Dataset [18] and our synthetic dataset. These datasets are used without considering object detection and tracking issues. CAVIAR and BEHAVE datasets are mostly used in the literature for event detection in surveillance domain (e.g. [96], [7], [97], [73], [76], [16], [25], [120], [122], [4] and [67]). Each video in these datasets was manually annotated to provide the ground truth. Both CAVIAR and BEHAVE datasets are in XML format. Figure 24 shows the structure of a sample XML file from CAVIAR Dataset. They provide actors and their pixel positions and blob width and height for each video frame. In SVAS, these datasets are parsed to be used by Trajectory Generation Module according to their XML format.



```
      <grouplist/>
    </frame>
  - <frame number="284">
    - <objectlist>
      - <object id="1">
          <orientation>160</orientation>
          <box yc="173" xc="89" w="41" h="23"/>
          <appearance>visible</appearance>
        - <hypothesislist>
          - <hypothesis id="1" prev="1.0" evaluation="1.0">
              <movement evaluation="1.0">walking</movement>
              <role evaluation="1.0">walker</role>
              <context evaluation="1.0">walking</context>
              <situation evaluation="1.0">moving</situation>
            </hypothesis>
          </hypothesislist>
        </object>
      - <object id="2">
          <orientation>130</orientation>
          <box yc="70" xc="188" w="48" h="40"/>
          <appearance>visible</appearance>
        - <hypothesislist>
          - <hypothesis id="1" prev="1.0" evaluation="1.0">
              <movement evaluation="1.0">walking</movement>
              <role evaluation="1.0">walker</role>
              <context evaluation="1.0">walking</context>
              <situation evaluation="1.0">moving</situation>
            </hypothesis>
          </hypothesislist>
        </object>
      </objectlist>
      <grouplist/>
    </frame>
  - <frame number="285">
```

Figure 24 Sample CAVIAR XML File

To deal effectively with the changes of viewing conditions, the features should be invariant to geometrical transformations such as translation, rotation, scaling and affine transformations. CAVIAR and BEHAVE datasets provide calibration data. Since the scenes in these datasets are not viewed from exactly top center and used cameras are a kind of fish eye camera, it is useful to calibrate the data. In this study, calibration of datasets is considered and calibrated datasets are also evaluated to show that proposed methods are independent of unit.

CAVIAR and BEHAVE datasets are generally challenging because they are not consistent for some conditions. Some event data have different scenarios with inadequate number of videos. In addition, intervals are too short for some events. As an example, the number of run events is not enough in CAVIAR Dataset. As it is stated in [73], meeting scenarios in CAVIAR Dataset vary. Event models are generated differently for datasets. For example, in CAVIAR videos, an object carried by a person is not tracked – only the person who carries it is tracked. The object will be tracked ('appear') if and only if the person leaves it somewhere. This input affects the generated event models.

4.1 Evaluation of CAVIAR Dataset

CAVIAR Benchmark Dataset consists of manually annotated 28 surveillance videos of a public space and contains several scenarios about "Fight", "Left Object" and "Meet" complex events. A sample screenshot of CAVIAR Dataset is shown in Figure 25.

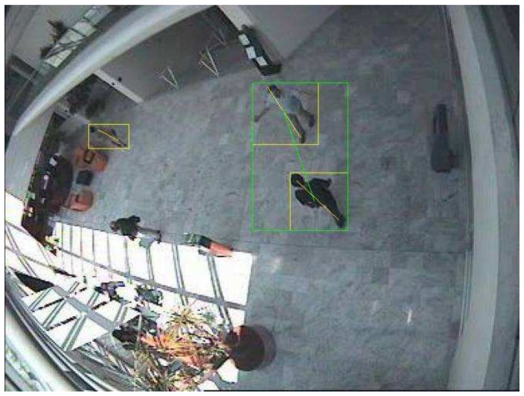


Figure 25 Sample screenshot of CAVIAR Dataset

4.1.1 Action Evaluation of CAVIAR Dataset

In this evaluation, hypothesis values, determined by CAVIAR team, are used. There are 4 types of actions in this dataset which are: "running", "inactive", "walking" and "active". The action "inactive" is considered as a "stand" action and the action "active" is considered as an instant move action. Action evaluations are done by using ten-fold cross validation method. For each action type, we divided the datasets to ten-fold, with nine-fold for training and one-fold for testing.

In CAVIAR Dataset, labeled frame numbers for “running”, “inactive”, “walking” and “active” are 406, 2934, 14134 and 1872 respectively. There are significant differences between the data sizes for action types. This leads to increase in the confusion of actions. For this evaluation, as it is stated in [96], it is required to fix the dataset size. Since the smallest number of training data available is for “run” action, we chose the complete set of “run” action and determined 15 intervals. Each interval is 1 second long and for each other action types, 15 intervals which are 1 second long are selected from the dataset. As a result, dataset size for each action type became equal. Results of confusion matrix evaluation are shown in Table 7.

Table 7 Results of confusion matrix evaluation

	running	**inactive**	**walking**	**active**
running	93.33%	0%	6.67%	0%
inactive	0%	100%	0%	0%
walking	13.33%	0%	73.34%	13.33%
active	0%	0%	6.67%	93.33%

The results of other studies using CAVIAR Dataset are shown in Table 8. [76] has low accuracy values particularly for “active” and “running” action types according to their HMM (Genetic Algorithm) test method for four action types. [96] tries some cluster based methods for this action dataset. Each method marks some actions high while the remaining ones are marked low. Considering all actions, their best evaluation is as follows: 92.3% for “running”, 77.4% for “inactive”, 77% for “walking” and 85.9% for “active”. In [96], only the performance for “walking” is higher than our method. If each frame in the hypothesis dataset is considered separately, then the results in Table 9 are obtained.

Table 8 Comparison of Action Detection Evaluation with other studies in CAVIAR Dataset

	[96]	**[76]**	**Our Results**
Running	92.3%	0%	93.33%
Inactive	77.4%	85%	100%
Walking	77%	88%	73.34%
Active	85.9%	0%	93.33%

Table 9 Results of Frame-based CAVIAR Dataset evaluation

Action Name	**Dataset Count**	**Detection Count**	**Undetection Count**	**Hit Ratio**
Running	406	371	35	91.38 %
Inactive	2934	2930	4	99.86 %
Walking	14134	11806	2328	83.53 %
Active	1872	1866	6	99.68 %

4.1.2 Complex Event Evaluation of CAVIAR Dataset

Hypothesis values, determined by the CAVIAR team, are used in this evaluation too. In CAVIAR Dataset, the number of events is small. For “interacting” (“meeting”) there are 6, for “fight” there are 3 and for “left object” there are 4 examples. To increase test data, we divide test intervals into sub-intervals with 15 frames long. We create 24 intervals for “meeting”, 18 intervals for “fight” and 9 intervals for “left object”. The results of confusion matrix evaluation for CAVIAR complex events are shown in Table 10.

Table 10 Results of confusion matrix evaluation for CAVIAR complex events

	meeting	**fight**	**left object**	**UNFOUND**
meeting	79.17%	12.5%	8.33%	0%
fight	16.67%	77.78%	5.55%	0%
left object	0%	0%	100.0%	0%

In [96], balanced dataset is used for complex event evaluation using decision trees. Comparing with [96], the accuracy of our complex event detections is high. The accuracy result for "meeting" and "fight" are nearly 70, for "left object" is nearly 75 in [96].

In [73], only "meeting" event detection is studied using clustering methods and compared with [7] in which Event Calculus is used. "meeting" accuracy of [7] and [73] are 67% and 89% respectively. Since [73] attacks only one complex event type, the accuracy result is high as expected. When the number of event types increases, confusion problems arise.

In [97], complex event evaluation is provided without low-level action detection using Event Calculus method. [97] uses low-level action values from CAVIAR ground truth. In [97], accuracies are as follows: for "meeting" is 85.5%, for "fighting" is 84.5%, for "left object" is 72.2%, for "walking" is 63.9%. We consider "walking" as action and accuracy value of our study is higher as shown in Section 4.1.1 above. The precision of "left object" event is also higher in our system. On the other hand, results of [97] are better for "fight" and "left object" events. However, since low-level actions are not detected in their work, some errors are inevitable. These comparisons are shown in Table 11.

Table 11 Comparison of Complex Event Detection Evaluation with other studies in CAVIAR Dataset

	[96]	[7]	[97]	[73]	Our Results
meeting	72%	67%	85.5%	89%	79.17%
fight	70%	100%	84.5%	-	77.78%
left object	75%	80%	72.2%	-	100.0%

4.2 Evaluation of BEHAVE Dataset

BEHAVE Dataset [16] consists of four videos and 76,800 frames in total and contains 25 frames per second with a resolution of 640× 480 pixels. It contains several scenarios about "InGroup" (IG), "Approach" (A), "WalkTogether" (WT), "Split" (S), "Ignore" (I), "Following" (F), "Chase" (C), "Fight" (Fi), "RunTogether" (RT) and "Meet" (M) events with a ground truth. Sample screenshots of BEHAVE Dataset are shown in Figure 26.

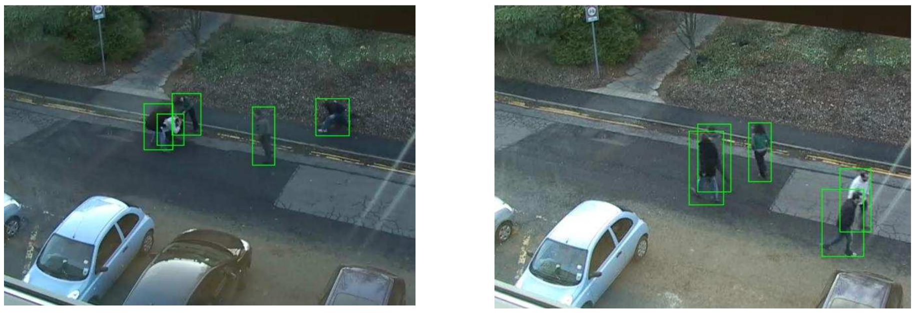


Figure 26 Sample screenshots of BEHAVE Dataset

This dataset is used in many studies in literature such as [25], [120], [122], [4] and [67]. The numbers of datasets for each event type are listed in Table 12.

Table 12 Dataset counts of each event type in BEHAVE Dataset

IG	A	WT	S	I	F	C	Fi	RT	M
35	25	43	23	2	1	10	19	12	1

Evaluations in this section are done using the ground truth values that are determined by BEHAVE team.

4.2.1 Action Evaluation of BEHAVE Dataset

In BEHAVE Dataset, actions are not defined for actors. However, we use individual behaviors of actors for action evaluation. For example, individual behaviors of each actor in “InGroup” and “Meet” events can be considered as “Stand” action. In the same manner, individual behaviors of each actor in “RunTogether”, “WalkTogether” and “Fight” events can be considered as “Run”, “Walk” and “Instant Move” actions, respectively. We generate 32 intervals for “Run”, 106 intervals for “Stand”, 82 intervals for “Walk”, and 19 intervals for “Instant Move” actions by considering each actor behavior in BEHAVE Dataset instances. We use ten-fold cross validation method. For each action type, we divide the interval datasets to ten-fold, with nine-fold for training and one-fold for testing. Our action evaluation results are shown in Table 13. Accuracy values are between 89% and 91%, which are very satisfactory.

Table 13 Confusion matrix of BEHAVE Dataset Action Evaluation

	Run	**Stand**	**Walk**	**Instant Move**
Run	90.63%	0%	9.37%	0%
Stand	0%	91.51%	5.66%	2.83%
Walk	1.22%	3.66%	91.46%	3.66%
Instant Move	0%	10.53%	0%	89.47%

4.2.2 Complex Event Evaluation of BEHAVE Dataset

In this evaluation, complex events are evaluated using ten-fold cross validation method as in Action Evaluation of BEHAVE Dataset. Since the numbers of “Meet”, “Ignore” and “Following” instances are low in dataset, they are not used in the evaluation. In addition, we do not consider group events, so the events “Approach to group” and “Split from group” are not evaluated. Results of confusion matrix evaluation for BEHAVE events are shown in Table 14.

Table 14 Results of confusion matrix evaluation for BEHAVE events

(IG: InGroup, WT: WalkTogether , C: Chase, Fi: Fight, RT: RunTogether)

	IG	**WT**	**C**	**Fi**	**RT**
IG	100%	0%	0%	0%	0%
WT	0%	88.37%	0%	0%	11.63%
C	0%	0%	100%	0%	0%
Fi	0%	5.26%	0%	89.48%	5.26%
RT	0%	25%	0%	0%	75%

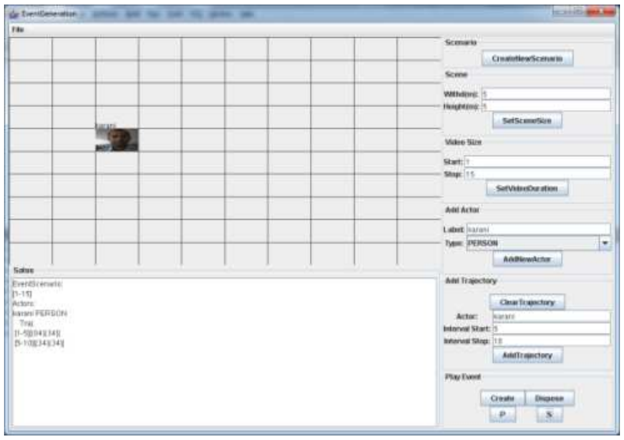
In [16], classification is provided using HMM without considering “Chase” and “RunTogether” events. Their average performance ranges from 80% to 90%. [25] considers only “Fight” and “Meet” events in BEHAVE dataset evaluations using object tracking and classification technique. Their accuracies are as follows: for “Meet” event, it is nearly 85% and for “Fight” event, it is nearly 70%. [120] evaluates “InGroup”, “WalkTogether”, “Fight” and “Split” events without confusion matrix method using Conditional Gaussian Process Dynamic Model. Their accuracies are 94.3%, 92.1%, 95.1% and 93.1%, respectively. In [122], accuracies of detected events in confusion matrix are between 52% and 88% using Multi-Group Causalities method. [4] uses Hierarchical Dirichlet Processes method. In [4], accuracies of detected events in confusion matrix are between 50% and 80%. [67] uses formal knowledge-based reasoning approach and multi-person tracker. In [67], “WalkTogether”, “RunTogether”, “Approach”, “Split” and “InGroup” events are evaluated with accuracies between 60% and 90%. Comparing our results with these studies; some accuracy values of our proposed work are apparently higher than the given values above, as shown in Table 14. Our accuracies are between 75% and 100%. Our average performance is 90.57%. Related comparisons are shown in Table 15.

Table 15 Comparison of Complex Event Detection Evaluation with other studies in BEHAVE Dataset
(IG: InGroup, WT: WalkTogether , C: Chase, Fi: Fight, RT: RunTogether)

	[25]	**[120]**	**[122]**	**[4]**	**[67]**	**Our Results**
IG	-	94.3%	88%	53.73%	90%	100%
WT	-	92.1%	88%	75%	60%	88.37%
C	-	-	52%	-	-	100%
Fi	70%	95.1%	-	80%	-	89.48%
RT	-	-	-	-	60%	75%

4.3 Evaluation of Synthetic Dataset

Synthetic dataset is evaluated in order to consider more event data in various scenarios. Test Data Generation Tool is developed for this purpose. Test data can be prepared for different event types easily by using this tool. Test Data Generation Tool is an application in which scene model and video event scenarios can be created. Various scenes can be designed as the composition of 50 cm * 50 cm grid cells, which approximately determines an effect area of an actor. Scenarios are created for top center view. So, there is no need for calibration. Various actor types can be defined. For each actor, trajectories are determined by giving time intervals. Actors and their trajectories can be determined by marking the route in the tool. Created scenarios can be played for controlling purposes and can be used by Trajectory Generation Module in learning and testing phases of event detection process. In Figure 27, user interface of Test Data Generation Tool is shown.



```
EventScenario:
[1-15]
Actors:
karani PERSON
   Traj:
 [1-5][0:4][3:4]
 [5-10][3:4][3:4]
```

Figure 27 User interface of Test Data Generation Tool

As it is shown in Figure 27 video size and actors can be also determined by using the right panel. For each actor trajectories are determined giving time intervals. Trajectories are determined by selecting grids according to path of the actor. Created scenario can be played by using bottom panel of right component. Scenario information is displayed in status panel which is located at the bottom of the view. By using "File" menu, export and import operations are done. A sample scenario for "meet" event which is prepared using Test Data Generation Tool is shown below:

Video Interval: 1-15
Scene: 5-5
Actors: 2
0-PERSON
1-PERSON
Interval: 0-1-5-0-5-4-5
Interval: 0-5-10-4-5-4-5
Interval: 0-10-15-4-5-0-5
Interval: 1-2-5-9-5-5-5
Interval: 1-5-10-5-5-5-5
Interval: 1-10-15-5-5-9-5

Movements are considered as one second actions so "instant move" action cannot be defined in Test Data Generation Tool currently. By using Test Data

Generation Tool, a total of 135 test data is created for “Run”, “Stand”, “Walk” actions and “Chase”, “Follow”, “Left Object”, “Meet”, “Walk Together”, “Run Together” complex events. By using leave-one-out testing method, detections are correct. However, detection accuracy decreases when some missing values added randomly. We add missing values to each training data such that “1 second missing value per training data” means: for each trajectory in training dataset, we remove randomly 1 second movement from trajectories as if they are occluded.

Table 16 Evaluation of Synthetic Dataset

Missing Value	**Detection Count**	**Hit Ratio**
No missing value	135	100 %
0.5 second missing value per training data	112	82.9 %
1 second missing value per training data	98	72.6 %
2 seconds missing value per training data	67	49.6 %
3 seconds missing value per training data	0	0 %

In Table 16, detection count decreases when missing value duration increases. Since generated test data trajectories are maximum 5 seconds long, the impact of missing value is very high. However, we can conclude that SVAS is robust for missing values nearly 10% of trajectories.

4.4 Performance Evaluation

The effect of the proposed methods on the performance is also evaluated. Both for “Action Detection” and “Complex Event Detection”, the proposed methods provide a great performance gain as discussed below. Calculations are measured by a personal computer which has 8 GB RAM and Intel i5-4210 CPU.

In “Action Detection” phase, Threshold Model elimination increases the performance as shown in Figure 28 and Table 17. One threshold query duration is nearly 0.06 msec., which can be considered as very fast. However, one Bayesian Network query duration is nearly between 4 msec. and 9 msec. Bayesian Network query duration is at least 80 times of the threshold query duration. For this reason, using Bayesian Network query when needed gives a big performance gain.

As listed in Table 17, there is little overhead for the first two rows since all intervals are selected as they contain action data. However, in real videos, actions exist only in a small portion of the video; hence, big performance gain is provided. In this case, only candidate intervals are queried using Bayesian Network.

Table 17 Action Detection Performance Evaluation

Test intervals Count	**Total TH query duration (msec)**	**Total BN query duration (msec)**	**Eliminated Interval Count By Th Model**	**Average 1 Th query duration (msec)**	**Average 1 BN query duration (msec)**	**Estimated Performance gain (msec)**
60	14,88	295,2	0	0,062	4.92	-14,88
150	33	1038	0	0,055	6.92	-33
400	89,6	2702,84	57	0,056	7,88	359,56
732	169,824	5261,76	123	0,058	8,64	892,896
813	178,86	3150,08	385	0,055	7,36	2654,74
800	188,8	172,52	781	0,059	9,08	6902,68

Formulas of calculations are as follows [“Action types” is the number of actions defined in the system which is currently 4.]:

$$\mathit{Average\ 1\ Th\ query\ duration} = \frac{\mathit{Total\ Th\ query\ duration}}{\mathit{Test\ intervals\ count} * \mathit{Action\ types}} \qquad (4.1)$$

$$\begin{pmatrix} \mathit{Average\ 1\ BN} \\ \mathit{query\ duration} \end{pmatrix} = \frac{\mathit{Total\ BN\ query\ duration}}{\mathit{Test\ intervals\ count} - \begin{pmatrix} \mathit{Eliminated\ interval} \\ \mathit{count\ by\ TH\ model} \end{pmatrix}} \qquad (4.2)$$

$$\begin{pmatrix} \mathit{Estimated} \\ \mathit{Performance\ gain} \end{pmatrix} = \left[\begin{pmatrix} \mathit{Eliminated\ Interval} \\ \mathit{count\ by\ TH\ model} \end{pmatrix} * \begin{pmatrix} \mathit{Average\ 1\ BN} \\ \mathit{query\ duration} \end{pmatrix} \right] - \begin{pmatrix} \mathit{Total\ TH} \\ \mathit{query\ duration} \end{pmatrix} \qquad (4.3)$$

In Figure 28, the performance gain increases when the number of eliminated intervals increases. This elimination power shows the necessity and importance of the Threshold Model.

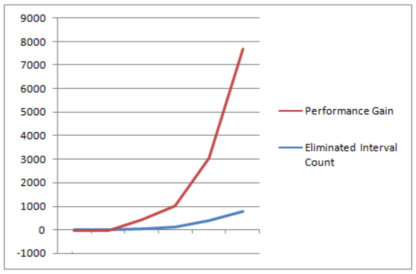


Figure 28 Action Detection Performance Evaluation

In “Complex Event Detection” phase, two-step elimination exists. One of them is Threshold Model while the other is BoA Model. Threshold Model elimination provides a big performance gain in complex event detection phase as in Figure 29 and Table 18. Performance gain of BoA Model is limited but its calculation is very fast such as almost 1 msec. So, BoA Model is also useful.

In this case, performance gain is huge since the inference operation in MLN is nearly between 1.5 sec and 4 sec. By considering this huge MLN query time, the overhead of BoA Model and Threshold Model calculations can be omitted. A small number of MLN queries offer higher performance. In Table 18, there is little overhead for the first three rows since all intervals are selected as they contain complex event data. However, in real videos, complex events occur only in a small portion of video. In this case, a big performance gain is provided.

Table 18 Complex Event Detection Performance Evaluation

Test inter-vals Count	Total TH query dura-tion (msec)	Total Action Detection duration (msec)	Total BoA query dura-tion (msec)	Eliminated Interval Count By Th model	Eliminated Interval Count By BoA model	Total MLN Query Dura-tion (msec)	Average 1 MLN query dura-tion (msec)	Estimated Performan-ce gain for MLN Query Elimination (msec)
6	1	13	1	0	0	10980	1830	-2
12	1	24	1	0	0	23808	1984	-2
14	3	24	1	0	0	55804	3986	-4
97	17	131	2	24	3	193200	2760	74501
223	34	319	6	55	3	268125	1625	94210
438	49	537	7	102	5	785794	2374	253962
721	54	817	12	176	6	1449371	2689	489332

Formulas of calculations are as follows:

$$\begin{pmatrix} Total\ Eliminated \\ Interval\ Count \end{pmatrix} = \begin{pmatrix} Eliminated\ Interval \\ Count\ By\ Th\ model \end{pmatrix} + \begin{pmatrix} Eliminated\ Interval \\ Count\ By\ BoA\ model \end{pmatrix} \qquad (4.4)$$

$$\begin{pmatrix} Average\ 1\ MLN \\ query\ duration \end{pmatrix} = \frac{Total\ MLN\ Query\ Duration}{\begin{pmatrix} Test\ intervals \\ Count \end{pmatrix} - \begin{pmatrix} Total\ Eliminated \\ Interval\ Count \end{pmatrix}} \qquad (4.5)$$

$$Total\ Elimination\ duration = \begin{pmatrix} Total\ TH \\ query\ duration \end{pmatrix} + \begin{pmatrix} Total\ BoA \\ query\ duration \end{pmatrix} \qquad (4.6)$$

$$\begin{pmatrix} Total\ Eliminated \\ MLN\ query\ duration \end{pmatrix} = \begin{pmatrix} Average\ 1\ MLN \\ query\ duration \end{pmatrix} * \begin{pmatrix} Total\ Eliminated \\ Interval\ Count \end{pmatrix} \qquad (4.7)$$

$$\begin{pmatrix} Estimated\ Performance\ gain \\ for\ MLN\ Query\ Elimination \end{pmatrix} = \begin{pmatrix} Total\ Eliminated \\ MLN\ query\ duration \end{pmatrix} - \begin{pmatrix} Total\ Elimination \\ duration \end{pmatrix} \qquad (4.8)$$

In this estimation, Duration of Action Detection can be omitted due to its low computational time.

Figure 29 shows that, the performance increases considerably when the number of eliminated intervals increases. Since the elimination of intervals depends on Threshold Model and BoA Model, the figure also shows the importance of these two models. Without them, MLN alone would be very inefficient.

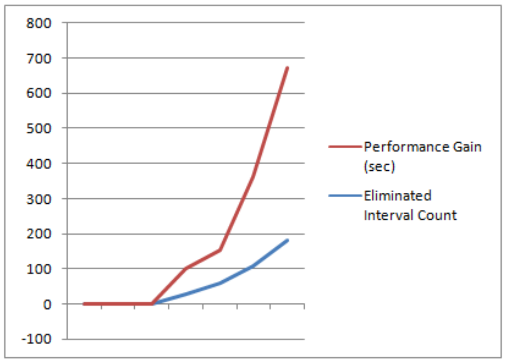


Figure 29 Complex Event Detection Performance Evaluation

4.5 Qualitative Evaluation

SVAS generates semantically meaningful event models in MLN format. Some of the generated complex event models are shown in Table 19. As shown in the table, the models are semantically consistent with expectations.

Table 19 Generated event models ('∧' states 'AND' operator and '!' states 'NOT' operator)

Event	**MLN Model**
Meet(a1, a2, t1)	Person(a1) ∧ Person(a2) ∧ Stand(a1, t1) ∧ Stand(a2, t1) ∧ Near(a1, a2, t1)
Fight(a1, a2, t1)	Person(a1) ∧ Person(a2) ∧ InstantMove(a1, t1) ∧ InstantMove(a2, t1) ∧ !DirectionSimilar(a1, a2, t1) ∧ Near(a1, a2, t1)
Walk Together (a1, a2, t1)	Person(a1) ∧ Person(a2) ∧ Walk(a1, t1) ∧ Walk(a2, t1) ∧ DirectionSimilar(a1, a2, t1) ∧ Near(a1, a2, t1)
Run Together (a1, a2, t1)	Person(a1) ∧ Person(a2) ∧ Run(a1, t1) ∧ Run(a2, t1) ∧ DirectionSimilar(a1, a2, t1) ∧ Near(a1, a2, t1)
Follow(a1, a2, t1)	Person(a1) ∧ Person(a2) ∧ Walk(a1, t1) ∧ Walk(a2, t1) ∧ DirectionSimilar(a1, a2, t1) ∧ Far(a1, a2, t1)
Chase(a1, a2, t1)	Person(a1) ∧ Person(a2) ∧ Run(a1, t1) ∧ Run(a2, t1) ∧ DirectionSimilar(a1, a2, t1) ∧ Far(a1, a2, t1)
LeftObject(a1, a2, t1)	Object(a1) ∧ Person(a2) ∧ Stand(a1, t1) ∧ Walk(a2, t1) ∧ Far(a1, a2, t1)
TakenObject (a1, a2, a3, t2)	Object(a1) ∧ Person(a2) ∧ Person(a3) ∧ Walk(a1, t1) ∧ Walk(a2, t1) ∧ Far(a1, a3, t1) ∧ Near(a1, a2, t1) ∧ Walk(a1, t2) ∧ Walk(a3, t2) ∧ Far(a1, a2, t2) ∧ Near(a1, a3, t2) ∧ (t2 > t1)

For generated models, some spatial relations, such as "Near" and "Far", are unique to the event. These predicates reflect the closeness between actors while the event is taking place. They are defined and handled with Threshold Models. However, for non-generated models which are designed by the user without learning operations, these predicates can be described by giving threshold values explicitly.

Table 19 shows that the generated rules are in FOL format and so they are readable. In addition, a user can manage these rules. The user can change the rules or add new scenarios to any complex event model only by text editing.

For example, the user can add a new scenario for “Left Object” complex event, where previous time periods of the event is considered as follows:

LeftObject(a1, a2, t1, t2) : Object(a1) ∧ Person(a2) ∧ Stand(a1, t1) ∧ Stand(a2, t1) ∧ Near(a1, a2, t1) ∧ Stand(a1, t2) ∧ Walk(a2, t1) ∧ Far(a1, a2, t2) ∧ (t2 > t1)

4.6 Evaluation of Learning Non-Interval-Based Complex Event Models Using Markov Logic Networks

For evaluation of Learning Non-Interval-Based Complex Event Models Using Markov Logic Networks, three event types are tested by using the leave-one-out testing method. For each event type, various different scenarios are considered in automatic model generation. These event types are:

Case 1: “Left Object” event (leftObject(person, object, time1, time2) means that the person leaves the object while the time passes from time1 to time2).

Case 2:“Taking Left Object” event (takingLeftObject(person1, person2, object, time1, time2) means that the owner of the object is person1. Person2 takes the object while the time passes from time1 to time2).

Case 3: “Meet” event (meet(person1, person2, time1, time2) means that person1 and person2 meet while the time passes from time1 to time2).

Eight videos for “left object”, three videos for “taking left object” and four videos for “meet” event are used as test videos. Four videos for “left object” were from CANTATA Dataset [18]. The others are newly created videos, in order to cover more scenarios. Sample image shots from these videos are displayed in Figure 30.

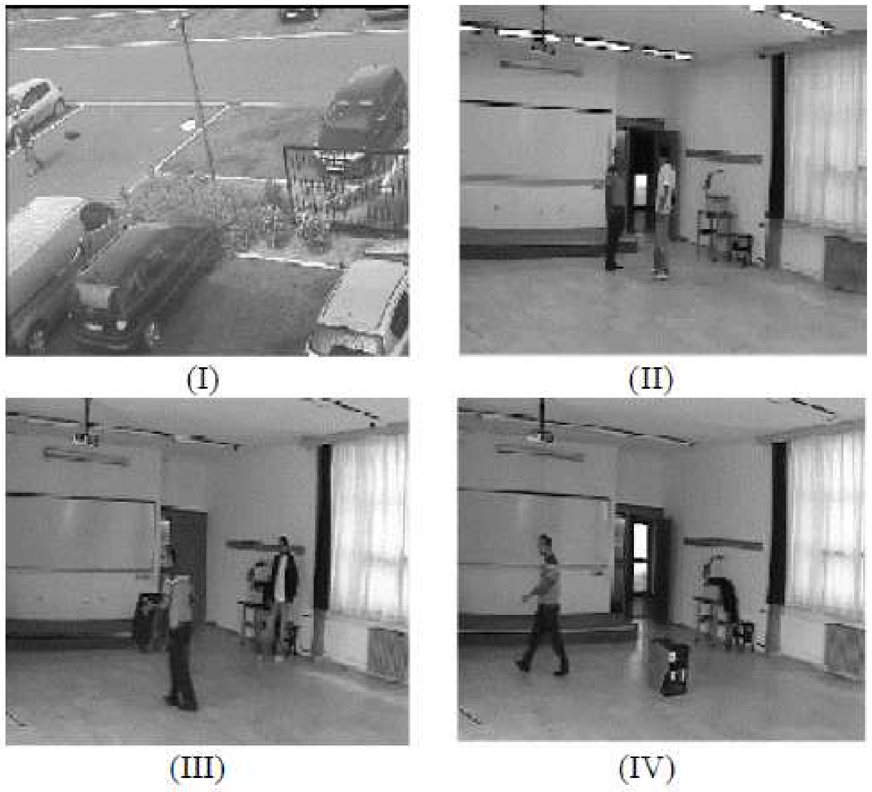


Figure 30 Sample image shots for I) Case 1, II) Case 3, III) Case 2, IV)

For Case 1, in each turn-around of leave-one-out testing method, seven of eight videos are used in event model generation. The generated models are used to detect events in the remaining video by using the "infer" command of Alchemy. After all turn-arounds, all of the possible predicates that can affect the event are listed with their average weights in Table 20. These predicates are chosen according to the parameters of the event. For example "closedistancePP" is not considered because "leftObject" has only one person attribute.

Table 20 Evaluation Results of Learning Non-Interval-Based Complex Event Models

Event Steps	Case 1			
	A	B	C	D
existPerson(p1, t1)	1.22	-1.22	0.14	-0.14
existPerson(p1, t2)	0.38	-0.38	0.42	-0.42
existObject(o1, t1)	1.16	-1.16	0.45	-0.45
existObject(o1, t2)	1.14	-1.14	0.36	-0.36
stopPerson(p1, t1)	0.44	-0.44	0.38	-0.38
stopPerson(p1, t2)	0.54	-0.54	0.48	-0.48
stopObject(o1, t1)	1.22	-1.22	0.44	-0.44
stopObject(o1, t2)	1.14	-1.14	0.29	-0.29
closeDistancePO(p1, o1, t1)	1.37	-1.37	0.67	-0.67
closeDistancePO(p1, o1, t2)	0.34	-0.34	1.31	-1.31
smallObject(o1)	1.17	-1.17	0.48	-0.48
ownedBy(p1, o1)	1.27	-1.27	-0.22	0.22
during(t1, t2)	1.50	-1.50	0.19	-0.19
before(t1, t2)	1.50	-1.50	0.19	-0.19
before(t2, t1)	0.60	-0.60	0.49	-0.49

The event model is created by considering the values of the table as follows ('∧' states 'AND' operator):

existPerson(p1,t1) ∧ existObject(o1,t1) ∧ existObject(o1,t2) ∧ stopObject(o1,t1) ∧ stopObject(o1,t2) ∧ closeDistancePO(p1, o1, t1) ∧ !closeDistancePO(p1, o1, t2) ∧ smallObject(o1) ∧ ownedBy(p1, o1) ∧ during(t1,t2) ∧ before (t1,t2) → leftObject(p1, o1, t1, t2)

The result is consistent with our expectations. For each predicate, A, B, C and D values are considered. If all values are low, the predicate is selected as irrelevant. If only the value of A is high, the predicate is added to the model. If only the value of C is high, the predicate is added to the model with the "not" operator. Both the value of A and the value of D or both the value of B and the value of C are not high for any predicate in this event, because none of the predicates causes the event alone. The values that are considered in the decision of the model are shaded in Table 20. Before the occurrence of the event; p1 and o1 must exist and close to each other. It is not necessary for the person to 'stop'; p1 can be moving while the event occurs. The object must be small and owned by p1. While the event occurs, p1 and o1 must be far and p1 can be out of the scene so "*existPerson(p1,t2)*" is irrelevant. Temporal relations are also correctly detected for the event. Event detections in test videos are also successful. The occurrences of the events are detected at the correct frames of the test videos. For Case 2 and Case 3, the generated event models are listed in Table 21.

Table 21 Generated event models for Case 2 and Case 3

Events	**Models**
takingLeftObject (p1, p2, o1, t1, t2)	*existPerson(p2, t1) ∧ existPerson(p2, t2) ∧ existObject(o1, t1) ∧ existObject(o1, t2) ∧ closeDistancePO(p2, o1, t1) ∧ closeDistancePO (p2, o1, t2) ∧ smallObject(o1) ∧ ownedBy(p1, o1) ∧ !ownedBy(p2, o1) ∧ during(t1, t2) ∧ before(t1, t2) ∧ after(t2, t1)*
meet(p1, p2, t1, t2)	*stopPerson(p1, t1) ∧ stopPerson(p2, t1) ∧ existPerson(p2, t1) ∧ existPerson(p2, t2) ∧ closeDistancePP(p1, p2, t1) ∧ stopPerson(p1, t2) ∧ stopPerson(p2, t2) ∧ closeDistancePP(p1, p2, t2) ∧ during(t1, t2) ∧ before(t1, t2)*

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In this thesis, a Surveillance Video Analysis System (SVAS) is proposed for the surveillance domain in which semantic rules and the definition of the event models can be learned or defined by the user for automatic detection and inference of complex video events. Interval-Based Spatio-Temporal Model (IBSTM) is proposed for event modeling, which fills the semantic gap between humans and video computer systems. By this model, basic spatial, temporal and logical relations in the surveillance domain can be established. Unlike current solutions, generated models are user understandable and manageable since IBSTM is based on first order logic. This modeling technique provides user to interfere generated event models in special conditions. In addition, IBSTM provides users to define their own models in case of unavailable training data.

SVAS does not need any predefined thresholds for scene or event model compared to many studies by its learning abilities. SVAS decreases human intervention through its event model learning ability from training data to ease user operation and prevent user errors. Threshold Models are proposed for learning valid values for features and calculating similarity values in detection phases in order to reflect the spatio-temporal motion analysis. SVAS can learn actions and complex event models using a set of hybrid machine learning techniques including Threshold Models, Bayesian Networks, Bag of Actions, Highly Cohesive Intervals and Markov Logic Networks. In addition, these powerful methods enable SVAS to handle uncertainty in order to be fault-tolerant in noisy conditions. Proposed and implemented algorithms generate probabilistic results to prevent discretization problems.

SVAS is extensively evaluated in different ways using many video data from various datasets such as CAVIAR, BEHAVE, CANTATA and synthetic datasets. Our evaluations show that the proposed approach improves the event recognition performance and precision as compared to the current state-of-the-art approaches in many action and complex event types in different event datasets. Moreover, performance evaluations confirm that SVAS has high performance ability due to its interval-based hierarchical manner and its high performance algorithms.

SVAS is based on the intervals instead of time points and different suitable machine learning techniques are used at different phases of the event detection. In addition, Threshold Models and BoA Model provide great efficiency in both action and complex event detection. These methods eliminate performance problems of MLN method in video domain. In detection phases, Threshold Models and BoA Models eliminate huge irrelevant intervals. Thus, the number of MLN predicates considerably decreases, and minimum MLN graph is created. It is observed that the performance of video event detection is highly increased by the proposed methods due to the interval-based hierarchical detection capability.

SVAS is flexible and extendable so that new features, action types, event types or actor types can be added. Any feature, which comes from the low level, can be used in SVAS. If low-level processes provide attributes such as movements of arm, leg or head, color or shape, these attributes can also be considered in event detection in SVAS.

To sum up, literature survey reveals that SVAS is a unique system, which possesses all key features of video domain needs stated above as a whole. On the one hand it is unique because it decreases human intervention through its learning capabilities, on the other hand it also enables human intervention when necessary through its manageable event model method. The system achieves all of them in the most efficient way through its machine learning methods.

In future work, we plan to test SVAS on more extensive data sets. Moreover, we intend to adapt the system to handle moving camera and multi camera datasets and to include other complex event types relevant for surveillance domain. In addition, the proposed Threshold Model can be used in other domains since it is independent of feature types. Also, the proposed methods can be used for automatic indexing or video browsing. A future direction of research is to focus on extending usage of Threshold Model in different domains.

Highly cohesive interval method can be used in the correction of training data boundaries, which is given by the user during training phase. The inconsistencies in the training boundaries can be eliminated. As another future work, consistency of training data and automatic training data correction can be implemented. In addition, the calibration and noise elimination method used in the current study should be enhanced for complex scenes.

Current prediction capability of SVAS is limited and evaluation of this ability is not implemented yet. Another future direction of research is increasing prediction capability of SVAS with more suitable datasets and comparing with other studies in this field.

SVAS needs a more user-friendly interface. Generated event models can be defined or edited using text editors in current interface of SVAS. The development of a more user-friendly application interface is another future work.

SVAS is currently a single threaded application and runs on a single CPU core. In the future, we have a plan to implement real-time surveillance applications. The proposed Threshold Model algorithms are kept simple in this study, to make them suitable for GPU programming. Implementing the current action detection phase in GPU is another future work which may provide enhanced performance.

REFERENCES

[1] Akdemir, U., Turaga, P., & Chellappa, R. (2008). An ontology based approach for activity recognition from video. In Proceedings of the *ACM International Conference on Multimedia*, pp. 709–712.

[2] Alevizos, E., Skarlatidis, A., Artikis, A., & Paliouras, G. (2015). Complex Event Recognition under Uncertainty: A Short Survey. In Proceedings of the *Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT)*, pp. 97-103.

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

[4] Al-Raziqi,A., & Denzler,J. (2016). Unsupervised Framework for Interactions Modeling between Multiple Objects. In Proceedings of the *11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016)* - Volume 4: VISAPP, pp. 509-516.

[5] Antoniou, G. (2011). Rule-Based Activity Recognition in Ambient Intelligence. *Rule-Based Reasoning, Programming, and Applications*, p. 1.

[6] Apajalahti, K., Hyvönen, E., Niiranen, J., & Räisänen, V. (2016). Combining Ontologies and Markov Logic Networks for Statistical Relational Mobile Network Analysis. *SEMPER ESWC*, pp. 36-45.

[7] Artikis, A., Sergot, M., & Paliouras, G. (2010). A Logic Programming Approach to Activity Recognition. In Proceedings of *ACM International Workshop on Events in Multimedia*, pp. 3-8.

[8] Artikis, A., Sergot, M., & Paliouras, G. (2015). An Event Calculus for Event Recognition. *IEEE Transactions On Knowledge And Data Engineering*, Vol. 27, No. 4, pp. 895–908.

[9] Ayers, D., & Shah, M. (2001). Monitoring human behavior from video taken in an office environment. *Image Vis. Comput*, 19 (12), pp. 833–846.

[10] Azorin-Lopez, J., Saval-Calvo, M., Fuster-Guillo, A., Garcia-Rodriguez, J., Cazorla, M., & Signes-Pont, M. T. (2016). Group activity description and recognition based on trajectory analysis and neural networks. In *Neural Networks (IJCNN), 2016 International Joint Conference*, pp. 1585-1592.

[11] Ballan, L., Bertini, M., Bimbo, A. D., Seidenari, L., & Serra, G. (2011). Event detection and recognition for semantic annotation of video. *Multimed Tools Appl*, 51, pp. 279–302.

[12] Baxter, R., Robertson, N. M., & Lane, D. (2010). Probabilistic Behaviour Signatures: Feature-Based Behaviour. *Information Fusion (FUSION)*, pp. 1-8.

[13] Baxter, R., Robertson, N. M., & Lane, D. (2011). Real-time event recognition from video via a “bag-of-activities”. In Proceedings of the *UAI Bayesian Modelling Applications Workshop*.

[14] Bhargava, M., Chen, C., Ryoo, M. S., & Aggarwal, J. K. (2009). Detection of object abandonment using temporal logic. *Machine Vision and Applications*, 20 (5), pp. 271–281.

[15] Biswas, R., Thrun, S., & Fujimura, K. (2007). Recognizing activities with multiple cues. In *Workshop on Human Motion*. LNCS 4814. Springer, pp. 255–270.

[16] Blunsden, S. J., & Fisher, R. B. (2010). The BEHAVE video dataset: ground truthed video for multi-person behavior classification. Annals of the BMVA, Vol (4), 1-12,

http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS. [Accessed: 05 01, 2018.]

[17] Brendel, W., Fern, A., & Todorovic, S. (2011). Probabilistic Event Logic for Interval-Based Event Recognition. *IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 3329-3336.

[18] CANTATA Video Dataset (http://www.multitel.be/~va/cantata/leftobject/) [Accessed: 02 10, 2017.]

[19] Chong, Y. S., & Tay, Y. H. (2015). Modeling Representation of Videos for Anomaly Detection using Deep Learning: A Review. *Computer Vision and Pattern Recognition (CVPR).*

[20] Cuntoor, N., Yegnanarayana, B., & Chellappa, R. (2005). Interpretation of state sequences in hmm for activity representation. In Proceedings of *IEEE International Conference Acoustics, Speech and Signal Processing*, Vol. 2, pp. 709–712.

[21] Delaitre, V., Laptev, I. & Sivic, J. (2010). Recognizing human actions in still images: a study of bag-of-features and part-based representations. In *BMVC 2010-21st British Machine Vision Conference.*

[22] Domingos, P. (2010). The alchemy tutorial. http://alchemy.cs.washington.edu/ tutorial/tutorial.pdf. [Accessed: 05 01, 2018.]

[23] Dousson, C., & Maigat, P.L. (2007). Chronicle recognition improvement using temporal focusing and hierarchisation. In Proceedings of *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 324-329.

[24] Dubba, K., Santos, P., Cohn, A., & Hogg, D. (2011). Probabilistic Relational Learning of Event Models from Video. *The 21stInternational Conference on Inductive Logic Programming.*

[25] Elhamod, M., & Levine, M. D. (2013). Automated Real-Time Detection of Potentially Suspicious Behavior in Public Transport Areas. *IEEE Transactions on Intelligent Transportation Systems*, pp. 688-699.

[26] Fanello, S.R., Gori, I, & Metta, G. (2013). Keep It Simple And Sparse: Real-Time Action Recognition. *The Journal of Machine Learning Research*, 14(1), pp. 2617-2640.

[27] Felzenswalb, P., McAllester, D., & Ramann, D. (2008). A Discriminatively Trained, Multiscale, Deformable PartModel. In Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-8.

[28] Frank, E., Hall, M. A., & Witten, I. H. (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition. http://www.cs.waikato.ac.nz/ml/weka. [Accessed: 05 01, 2018.]

[29] Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, 29(2-3), pp. 131-163

[30] Fu, Z., Hu, W., & Tan, T. (2005). Similarity based vehicle trajectory clustering and anomaly detection. *IEEE Conference on Image Processing*, vol. 2, pp. 602-605.

[31] Fusier, F., Valentin, V., Bremond, F., Thonnat, M., Borg, M., Thirde, D., & Ferryman, J. (2007). Video understanding for complex activity recognition. *Mach. Vis. Appl.* 18 (3), pp. 167-188.

[32] Gan, C., Wang, N., Yang, Y., Yeung, D., & Hauptmann, A. G. (2015). DevNet: A Deep Event Network for Multimedia Event Detection and Evidence Recounting. In Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2568-2577.

[33] Gayathri, K. S., Easwarakumar, K. S., & Elias, S. (2017). Probabilistic ontology based activity recognition in smart homes using Markov Logic Network. *Knowledge-Based Systems*, 121, pp. 173-184.

[34] Ghanem, N., DeMenthon, D., Doermann, D., & Davis, L. (2004). Representation and recognition of events in surveillance video using Petri nets. *IEEE International Conference on Computer Vision and Pattern Recognition Workshop*, pp. 104–112.

[35] Gilks, W. R. (2005). Markov chain monte carlo. *Encyclopedia of Biostatistics*, 4.

[36] Gong, S., & Xiang, T. (2003). Recognition of group activities using dynamic probabilistic networks. *The 9th International Conference on Computer Vision*, pp. 742-749.

[37] Gupta, H., Yu, L., Hakeem,A., Choe, T. E., Haering, N., & Locasto,M. (2007). Multimodal Complex Event Detection Framework for Wide Area Surveillance. *Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 47-54.

[38] Gutchess, D., Trajkovic, M., Cohen-Solal, E., Lyons, D., & Jain, A.K. (2001). A Background model initialization algorithm for video surveillance. In Proceedings of *IEEE International Conference on Computer Vision (ICCV)*, pp. 733-740.

[39] Hakeem, A., & Shah, M. (2007). Learning, detection and representation of multi-agent events in videos. *Artificial Intelligence*. Volume 171 Issue 8-9, pp. 586-605.

[40] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support Vector Machines. *IEEE Intelligent Systems and their applications*, 13(4), pp. 18-28.

[41] Helaoui, R., Niepert, M., & Stuckenschmidt , H. (2011). Recognizing Interleaved and Concurrent Activities: A Statistical-Relational Approach. In Proceedings of the *9th Annual IEEE International Conference on Pervasive Computing and Communication*, pp. 1-9.

[42] Henry, T., Janapriya, E. & Silva L. (2003). An automatic system for multiple human tracking and actions recognition in office environment. *IEEE Proc. of ICASSP*, pp. 45–48.

[43] Hoey, J. Bertoldi, A., Poupart, P., & Mihailidis, A. (2007). Assisting persons with dementia during handwashing using a partially observable markov decision process. *International Conference on Computer Vision Systems (ICVS).*

[44] Hongeng, S., Nevatia, R., & Bremond, F. (2004). Video-based event recognition: activity representation and probabilistic recognition methods. *Comput. Vis. Image Understand.* 96 (2), pp. 129–162.

[45] Hu, W., Xiao, X., Fu, Z. , Xie, D., Tan, T. & Maybank, S. (2006). A system for learning statistical motion patterns. *IEEE Trans. on Pattern Analysis and Machine Int.*, vol. 28, no. 9, pp. 1450–1464.

[46] Jhuo, I., & Lee, D.T. (2014). Video Event Detection via Multi-modality Deep Learning. *22nd International Conference on Pattern Recognition*, pp. 666-671.

[47] Jiang, F., Wu, Y., & Katsaggelos, A. K. (2009). A Dynamic Hierarchical Clustering Method for Trajectory-Based Unusual Video Event Detection. *IEEE Transactions on Image Processing*, 18(4), pp. 907-913.

[48] Jiang, F., Yuan, J., Tsaftaris, S. A., & Katsaggelos, A. K. (2011). Anomalous video event detection using spatiotemporal context. *Computer Vision and Image Understanding*, vol. 115, pp. 323-333.

[49] Johnson, N., & Hogg, D. (1996). Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, vol. 14, pp. 583–592.

[50] Kapoor, A., Biswas, K. K., & Hanmandlu, M. (2017). Unusual human activity detection using Markov Logic Networks. In *Identity, Security and Behavior Analysis (ISBA), IEEE*, pp. 1-6.

[51] Kardas, K., Ulusoy, İ., & Cicekli, N.K. (2013).Learning Complex Event Models Using Markov Logic Networks. *In Multimedia and Expo Workshops (ICMEW), IEEE International Conference on Multimedia and Expo*, pp. 1-6.

[52] Kembhavi, A., Yeh, T., & Davis, L.S. (2010). Why Did the Person Cross the Road (There)? Scene Understanding Using Probabilistic Logic Models and Common Sense Reasoning. *In European Conference on Computer Vision*, pp. 693-706.

[53] Kim, P. S., Lee, D. G., & Lee, S. W. (2018). Discriminative context learning with gated recurrent unit for group activity recognition. *Pattern Recognition*, 76, pp. 149-161.

[54] Ko, T. (2008). A Survey on Behavior Analysis in Video Surveillance for Homeland Security Applications. *37th IEEE applied imagery pattern recognition workshop*, pp. 1–8.

[55] Kuettel, D., Breitenstein, M., Gool, L., & Ferrari, V. (2010). What's going on? discovering spatio-temporal dependencies in dynamic scenes. In Proceedings *IEEE Conference Computer Vision Pattern Recognition*, pp. 1951–1958.

[56] Kumar, P., Ranganath, Weimin , S. H., & Sengupta, K. (2005). Framework for real-time behavior interpretation from traffic video. *IEEE Trans. On Intelligent Transportation Systems*, vol. 6, no. 1, pp. 43–53.

[57] Lavee, G., Rivlin, E., & Rudzsky, M. (2009). Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video. *Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions*, 39(5), pp. 489-504.

[58] Lavee, G., Rudzsky, M., & Rivlin, E. (2010). Propagating uncertainty in Petri nets for activity recognition. In Proceedings of *International Symposium on Advances in Visual Computing*, pp. 706–715.

[59] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), pp. 436-444.

[60] Li, W., Mahadevan, V., & Vasconcelos N. (2013). Anomaly Detection and Localization in Crowded Scenes. *IEEE transactions on pattern analysis and machine intelligence*, 36(1), pp. 18-32.

[61] Li, X., & Cai, Z. M. (2016). Anomaly detection techniques in surveillance videos. *In Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 54-59.

[62] Lv, F., Song, X., Wu, B., Singh, V. K., & Nevatia, R. (2006). Left-Luggage Detection using Bayesian Inference. In Proceedings *9th IEEE International Workshop Perform. Eval. Tracking Surveillance*, pp. 83-90.

[63] Makris, D., & Ellis, T., (2005). Learning semantic scene models from observing activity in visual surveillance, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 35, no. 3, pp. 397-408.

[64] Martinez-Tomas, R., Rincon, M., Bachiller, M., & Mira, J. (2008). On the correspondence between objects and events for the diagnosis of situations in visual surveillance tasks. *Pattern Recogn. Lett.* 29 (8), pp. 1117-1135.

[65] Mo, X., Monga, V., Bala R. & Fan, Z. (2012). A Joint Sparsity Model for Video Anomaly Detection. In *Signals, Systems and Computers (ASILOMAR),* pp. 1969-1973.

[66] Morariu, V. I., & Davis, L. S. (2011). Multi-agent event recognition in structured scenarios. *Computer Vision and Pattern Recognition (CVPR)*, pp. 3289-3296.

[67] Muench, D., Becker, S., Hubner, W., & Arens, M. (2012). Towards a Real-Time Situational Awareness System for Surveillance Applications in Unconstrained Environments. *7th Security Research Conference, Future Security*, pp. 517-521.

[68] Neuhaus, H. (2008). A Semantic Concept for the Mapping of Low-Level Analysis Data to High-Level Scene Descriptions (Doctoral Dissertation, Technische Universitat Ilmenau, Germany).

[69] Nguyen, N.T., Phung D.Q., Venkatesh, S., & Bui, H. (2005).Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. In Proceedings of *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 955-960.

[70] Oliver, N., & Horvitz, E. (2005). A comparison of HMMs and dynamic bayesian Networks for recognizing office activities. *International conference on user modeling*, Vol. 3538, pp. 199-209.

[71] Oliver, N., Horvitz, E., & Garg, A. (2004). Layered representations for human activity recognition. In *Computer Vision and Image Understanding Journal*, Vol. 96:2, pp. 163–180.

[72] Onal, I., Kardas, K., Rezaeitabar, Y., Bayram, U., Bal, M., Ulusoy, İ., & Cicekli, N.K. (2013). A Framework For Detecting Complex Events In Surveillance Videos. *In Multimedia and Expo Workshops (ICMEW), IEEE International Conference on Multimedia and Expo*, pp. 1-6.

[73] Patino, L. , & FerrymanJ. (2015). Meeting detection in video through semantic analysis. In Proceedings of *12th IEEE International Conference onAdvanced Video and Signal Based Surveillance (AVSS)*, pp. 1-6.

[74] Patterson, D.J., Fox, D., Kautz, H., & Philipose, M. (2005). Finegrained activity recognition by aggregating abstract object usage. In *ISWC ’05: Proceedings of the Ninth IEEE International Symposium on Wearable Computers.: IEEE Computer Society*, pp. 44-51.

[75] Pei, M., Jia, Y., & Zhu, S. (2011). Parsing Video Events with Goal inference and Intent Prediction. *IEEE International Conference On Computer Vision*, pp. 487-494.

[76] Perez, O., Piccardi, M., Garcia, J., & Molina, J. M. (2007). Comparison of Classifiers for Human Activity Recognition. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*: pp. 192-201.

[77] Piciarelli, C., Micheloni, C., & Foresti, G. (2008). Trajectory-based anomalous event detection. *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1544-1554.

[78] PostgreSQL: open source object-relational database system.

https://www.postgresql.org/. [Accessed: 05 01, 2018.]

[79] Reddy, S., Gal,Y., & Shieber,S. (2009). Recognition of Users Activities Using Constraint Satisfaction. *Springer Berlin / Heidelberg*, vol. 5535, pp. 415-421.

[80] Ribeiro, P. C., & Santos-Victor, J. (2005). Human Activity Recognition from Video: modeling, feature, selection and classification arquitecture, HAREM. Oxford, UK : *International Workshop on Human Activity Recognition and Modelling*, pp. 61-78.

[81] Riboni, D., Sztyler, T., Civitarese, G., & Stuckenschmidt, H. (2016). Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning. In Proceedings of the *2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1-12.

[82] Richardson, M., &Domingos, P. (2006). Markov Logic. *Machine Learning*, Vol. 62, pp. 107-136.

[83] Robertson, N., Reid, I., & Brady, M. (2008). Automatic human behaviour recognition and explanation for CCTV video surveillance. 21(3): *Security Journal*, pp. 173-188.

[84] Romdhane, R., Boulay, B., Bremond, F., & Thonnat, M. (2011). Probabilistic Recognition of Complex Event. *ICVS 2011 : 8th International Conference on Computer Vision Systems*, pp. 122-131.

[85] Romdhane, R., Bremond, F., & Thonnat, M. (2010). A framework dealing with uncertainty for complex event recognition. In Proceedings of the *IEEE International Conference on Advanced Video and Signal based Surveillance*, pp. 392-399.

[86] Romdhane, R., Crispim, C. F., Bremond, F., & Thonnat, M. (2013). Activity Recognition and Uncertain Knowledge in Video Scenes. *Advanced Video and Signal Based Surveillance (AVSS)*, pp. 377-382.

[87] Ryoo, M.S., & Aggarwal, J.K. (2008). Recognition of High-level Group Activities Based on Activities of Individual Members. In Proceedings of the *2008 IEEE Workshop on Motion and video Computing*.

[88] Ryoo, M. S., & Aggarwal, J. K. (2009). Stochastic Representation and Recognition of High-level Group Activities. *Computer Vision and Pattern Recognition Workshops*, pp. 1-8.

[89] Saligrama V., & Chen, Z. (2012). Video Anomaly Detection Based on Local Statistical Aggregates. In *Computer Vision and Pattern Recognition (CVPR)*, pp. 2112-2119.

[90] Saligrama, V., Konrad, J. & Jodoin, P. (2010). Video anomaly identification. *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 18-33.

[91] SanMiguel, J.C., Escudero-Viñolo, M., Martínez, J. M., & Bescós, J. (2011). Real-time single-view video event recognition in controlled environments. *Content-Based Multimedia Indexing*, pp. 91-96.

[92] SanMiguel, J. C., & Martínez, J. M. (2008). Robust unattended and stolen object detection by fusing simple algorithms. In *Advanced Video and Signal Based Surveillance. AVSS'08*, pp. 18-25.

[93] SanMiguel, J. C., & Martínez, J. M. (2012). A semantic-based probabilistic approach for real-time video event recognition. *Computer Vision and Image Understanding*, 116(9), pp. 937-952.

[94] Scariaa, E., Tb, A. A., & Isaacc, E. (2016). Suspicious Activity Detection in Surveillance Video using Discriminative Deep Belief Network. *International Journal of Control Theory and Applications*, 9(43), pp. 261-267.

[95] Shet, V., Harwood, D., & Davis, L. (2005). VidMAP: video monitoring of activity with Prolog. *Advanced Video and Signal Based Surveillance IEEE*, pp. 224-229.

[96] Simon, C., Meessen, J., & De Vleeschouwer C. (2010). Visual event recognition using decision trees. *Multimedia Tools Applications*, 50(1), pp. 95-121.

[97] Skarlatidis, A., Artikis, A., Filippou, J., & Paliouras, G. (2015). A probabilistic logic programming event calculus. *Theory and Practice of Logic Programming*, 15(2), pp. 213-224.

[98] Skarlatidis, A., Paliouras, G., Vouros, G. A., & Artikis, A. (2011). Probabilistic Event Calculus based on Markov Logic Networks. *Rule-Based Modeling and Computing on the Semantic Web*, pp. 155-170.

[99] Smullyan, R. M. (1995). First-order logic. *Courier Corporation, North Chelmsford.*

[100] Song, Y. C., Kautz, H., Li, Y., & Luo, J. (2013). A General Framework for Recognizing Complex Events in Markov Logic. In *AAAI Workshop: Statistical Relational Artificial Intelligence*.

[101] Tang, Z., Hwang, J. N., Lin, Y. S., & Chuang, J. H. (2016). Multiple-kernel adaptive segmentation and tracking (MAST) for robust object tracking. In *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference*, pp. 1115-1119.

[102] Tao J., & Tan Y.P. (2003). Color appearance-based approach to robust tracking and recognition of multiple people. In *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference*, 1, pp. 95-99.

[103] Tian, Y., Feris, R.,Liu, H., Humpapur, A., & Sun, M. (2010). Robust Detection of Abandoned and Removed Objects in Complex Surveillance Videos. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(5), pp. 565-576.

[104] Town, C. (2006). Ontological inference for image and video analysis. *Machine Vision and Appliations*. 17(2), pp. 94-115.

[105] Tran, S. D., & Davis, L. S. (2008). Event modeling and recognition using Markov Logic Networks. In *European Conference on Computer Vision*, pp. 610-623.

[106] Tuffy: A Scalable Markov Logic Inference Engine.
http://research.cs.wisc.edu/hazy/tuffy/. [Accessed: 05 01, 2018.]

[107] Vail, D.L., Veloso, M.M, & Lafferty, J.D. (2007). Conditional random fields for activity recognition. *International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, p. 235.

[108] Vishwakarma, S., & Agrawal, A. (2013). A survey on activity recognition and behavior understanding in video surveillance. *Visual Comput.*, 29(10), pp. 983-1009.

[109] Wallach, H. M. (2006). Topic modeling: beyond bag-of-words. In Proceedings of the *23rd International Conference on Machine learning*, pp. 977-984.

[110] Wang X., Tieu K., & Grimson. E. (2006). Learning Semantic Scene Models by Trajectory Analysis. *Computer Vision–ECCV*, pp. 110-123.

[111] Wateosot, C., & Suvonvorn, N. (2017). Fighting detection using interaction energy force. In *International Conference on Robotics and Machine Vision*, pp. 1025304-1025304.

[112] Wu, T., Lian, C., & Hsu, J.Y. (2007). Joint recognition of multiple concurrent activities using factorial conditional random fields. In Proceedings of *AAAI Workshop on Plan, Activity, and Intent Recognition.*

[113] Xiang, T., & Gong, S. (2008). Video behavior profiling for anomaly detection. *IEEE Trans. on Pattern Analysis and Machine Int.*, vol. 30, no. 5, pp. 893-908.

[114] Xu, D., Ricci, E., Yan, Y., Song, J., & Sebe, N. (2015). Learning Deep Representations of Appearance and Motion for Anomalous Event Detection. In Proceedings of *IEEE Computer Vision and Pattern Recognition (CVPR).*

[115] Yang, B., & Nevatia, R. (2012). Multi-Target Tracking by Online Learning of Non-linear Motion Patterns and Robust Appearance Models. In Proceedings of *IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 1918-1925.

http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/ [Accessed: 05 01, 2018.]

[116] Yang, E., Gwak, J., & Jeon, M. (2017). Conditional Random Field (CRF)-Boosting: Constructing a Robust Online Hybrid Boosting Multiple Object Tracker Facilitated by CRF Learning. *Sensors*, 17(3), p. 617.

[117] Yang, E., Gwak, J., & Jeon, M. (2017). Multi-human tracking using part-based appearance modelling and grouping-based tracklet association for visual surveillance applications. *Multimedia Tools and Applications*, 76(5), pp. 6731-6754.

[118] Yun, K., Yoo, Y., & Choi, J. Y. (2017). Motion interaction field for detection of abnormal interactions. *Machine Vision and Applications*, 28(1-2), pp. 157-171.

[119] Yildirim, Y., Yazici, A. & Yilmaz, T. (2013) Automatic Semantic Content Extraction in Videos using a Fuzzy Ontology and Rule-based Model. In *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(1), pp. 47-61.

[120] Yin, Y., Yang, G., Xu, J., & Man, H. (2012). Small Group Human Activity Recognition. In *2012 19th IEEE International Conference on Image Processing (ICIP)*, pp. 2709-2712.

[121] Zeng, Z. Q., Yu, H. B., Xu, H. R., Xie, Y. Q., & Gao, J. (2008). Fast training Support Vector Machines using parallel sequential minimal optimization. In *Intelligent System and Knowledge Engineering (ISKE*): Vol. 1, pp. 997-1001.

[122] Zhang, C., Yang, X., Lin, W., & Zhu J. (2012). Recognizing Human Group Behaviors with Multi-Group Causalities. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT)*: Vol. 3, pp. 44-48.

[123] Zhang, T., Jia, W., Gong, C., Sun, J., & Song, X. (2017). Semi-supervised dictionary learning via local sparse constraints for violence detection. *Pattern recognition letters*, pp. 1-7.

[124] Zhang, T., Lu, H., & Li, S. Z. (2009). Learning semantic scene models by object classification and trajectory clustering. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE*, pp. 1940-1947.

[125] Zhang, T., Yang, Z., Jia, W., Yang, B., Yang, J., & He, X. (2016). A new method for violence detection in surveillance scenes. *Multimedia Tools and Applications*, 75(12), pp. 7327-7349.

[126] Zhang, Y., Zhang, Y., Swears, E., Larios, N.,Wang, Z., & Ji, Q. (2013). Modeling Temporal Interactions with Interval Temporal Bayesian Networks for Complex Activity Recognition, *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 35 (10), pp. 2468-2483.

[127] Zhao, T., & Nevatia, R. (2004). Tracking Multiple Humans in Complex Situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 9, pp. 1208-1221.

[128] Zhong, H., Shi, J. & Visontai M. (2004). Detecting Unusual Activity in Video. In Proceedings of the *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-8.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Kardaş, Karani

Nationality: Turkish (TC)

Date and Place of Birth: 1981, Diyarbakır

Email: karani_kardas@yahoo.com

EDUCATION

Degree	Institution	Year of Graduation
MS	METU, Department of Computer Engineering	2007
BS	Ege University, Department of Computer Engineering	2003

WORK EXPERIENCE

Year	Organization	Position
2003 - Present	HAVELSAN AŞ., Ankara	Senior Software Engineer

PUBLICATIONS

[1] Kardas, K. & Cicekli, N.K. (2017). SVAS: Surveillance Video Analysis System. Expert Systems with Applications.

[2] Akkan, M., Kardas, K. & Karacan, H. (2016). Coğrafi bilgi sistemi istemci sunucu mimarisinde ağ trafiği optimizasyonu. SAVTEK.

[3] Kardas, K., Ulusoy, İ., & Cicekli, N.K. (2013). Learning Complex Event Models Using Markov Logic Networks. In Multimedia and Expo Workshops (ICMEW), IEEE International Conference on Multimedia and Expo. (pp. 1-6). IEEE.

[4] Onal, I., Kardas, K., Rezaeitabar, Y., Bayram, U., Bal, M., Ulusoy, İ., & Cicekli, N.K. (2013). A Framework For Detecting Complex Events In Surveillance Videos. In Multimedia and Expo Workshops (ICMEW), IEEE International Conference on Multimedia and Expo. (pp. 1-6). IEEE.

[5] Kardas, K., Turhan, E. Z., Korkmaz, H. (2012). Görev Planlama İçin GPS Analizi. I. Ulusal Havacılık Teknolojisi Ve Uygulamaları Kongresi.

[6] Kardas, K. & Senkul, P. (2007). Enhanced Semantic Operations for Web Service Composition. In Computer and information sciences, 2007. ISCIS 2007. 22nd international symposium on (pp. 1-6). IEEE.

[7] Karakoc, E., Kardas, K. & Senkul, P. (2006). A Workflow-based Web Service Composition System. In Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology (pp. 113-116). IEEE Computer Society

TEZ FOTOKOPİSİ İZİN FORMU

ENSTİTÜ

Fen Bilimleri Enstitüsü	x
Sosyal Bilimler Enstitüsü	
Uygulamalı Matematik Enstitüsü	
Enformatik Enstitüsü	
Deniz Bilimleri Enstitüsü	

YAZARIN

Soyadı :	KARDAŞ
Adı :	KARANİ
Bölümü :	BİLGİSAYAR MÜHENDİSLİĞİ

TEZİN ADI (İngilizce): SEMANTIC VIDEO ANALYSIS FOR SURVEILLANCE SYSTEMS

TEZİN TÜRÜ : Yüksek Lisans [ ] Doktora [X]

1. Tezimin tamamından kaynak gösterilmek şartıyla fotokopi alınabilir.	X
2. Tezimin içindekiler sayfası, özet, indeks sayfalarından ve/veya bir bölümünden kaynak gösterilmek şartıyla fotokopi alınabilir.	
3. Tezimden bir (1) yıl süreyle fotokopi alınamaz.	

TEZİN KÜTÜPHANEYE TESLİM TARİHİ: