Expert Systems With Applications 89 (2017) 343–361

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

SVAS: Surveillance Video Analysis System

Karani Kardas*, Nihan Kesim Cicekli

Department of Computer Engineering, Middle East Technical University, Universiteler Mahallesi Dumlupinar Bulvarı No:1, 06800, Ankara, Turkey

ARTICLE INFO

Article history:
Received 19 January 2017
Revised 28 July 2017
Accepted 29 July 2017
Available online 29 July 2017

Keywords:
Event detection
Markov logic networks
Video surveillance
Event model learning
Event inference

ABSTRACT

This paper introduces a Surveillance Video Analysis System, called SVAS, for surveillance domain, in which the 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 understandable and manageable IBSTM event models. Hybrid machine learning methods are proposed and used. 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 manner. 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 the 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 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.

1. Introduction

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. Al-Raziqi & Denzler, 2016; Artikis, Sergot, & Paliouras, 2010; Blunsden & Fisher, 2010; Elhamod & Levine, 2013; Muench, Becker, Hubner, & Arens, 2012; Patino & Ferryman, 2015; Perez, Piccardi, Garcia, & Molina, 2007; Simon, Meessen, & De Vleeschouwer, 2010; Skarlatidis, Artikis, Filippou, & Paliouras, 2015; Yin, Yang, Xu, & Man, 2012; Zhang, Yang, Lin, & Zhu, 2012) 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. Antoniou, 2011; Lv, Song, Wu, Singh, & Nevatia, 2006; Neuhaus, 2005; Robertson, Reid, & Brady, 2008; SanMiguel, Escudero-Viñolo, Martínez, & Bescós, 2011; Skarlatidis et al., 2015) 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 levels and high levels. 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.

* Corresponding author.
E-mail addresses: kkardas@havelsan.com.tr (K. Kardas), nihan@ceng.metu.edu.tr (N.K. Cicekli).

http://dx.doi.org/10.1016/j.eswa.2017.07.051
0957-4174/

There are various successful studies taking low levels into consideration (e.g. Felzenswalb, McAllester, & Ramann, 2008; Gutchess, Trajkovic, Cohen-Solal, Lyons, & Jain, 2001; Zhao & Nevatia, 2004) 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 is not the main concern of our study, studies on low level will also be briefly discussed in the related work section of this paper.

High levels include video event detection, inference of events and prediction of possible events. Finding semantic relations between actors and detection of 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 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. In many different domains, various event types are analyzed, compared and categorized. The literature survey on video event recognition (which is discussed in Section 2) reveals that event modeling methods have to possess some important features.

As the first feature, event modeling methods should deal with uncertainty in order to be fault-tolerant (Lavee, Rivlin, & Rudzsky, 2009). 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. It is apparent that automatic inference of event models from data is essential for adaptability and scalability of event understanding systems. As a result, learning ability from training data is another important feature of event model methods.

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 (Lavee et al., 2009). It should be semantically meaningful for the user and enable user intervention when needed.

This paper introduces a Surveillance Video Analysis System, shortly called SVAS, which aims at solving the above 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 method, 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 disappears. IBSTM uses Markov Logic Networks (MLN) (Richardson & Domingos, 2006) 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 rest of the paper is organized as follows: In Section 2, related work on Complex Event Detection and recognition is discussed. The overall architecture of SVAS is introduced in Section 3. In Section 4, Trajectory Generation Module is presented. In Section 5, Event Model Learning is explained in detail, which includes Action Model Learning, Action Detection and Complex Event Model Learning processes. Complex Event Detection is discussed in Section 6. Experiments and their results are provided in Section 7. Section 8 concludes the paper with possible future studies.

2. Related work

Research on Complex Event Detection and recognition is 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.

Pixel-based operations can be considered as the lowest level jobs in complex event recognition process. 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. Bhargava, Chen, Ryoo, & Aggarwal, 2009; Jiang, Yuan, Tsaftaris, & Katsaggelos, 2011; Tian, Feris, Liu, Humpapur, & Sun, 2010). In 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 sub-

traction, object detection, object tracking and object recognition (e.g. Felzenswalb et al., 2008; Gutchess et al., 2001; Zhao & Nevatia, 2004). Since, at high level, there are no pixel-based operations directly, this kind of studies is not discussed here in detail.

The studies at high levels can be grouped according to the methods used. Rule-based methods, such as Antoniou (2011), Neuhaus (2005) and Robertson et al. (2008), can not handle uncertainty since they are not probabilistic. Yildirim, Yazici, & Yilmaz (2013) extends rule-based system with fuzzy-based manner. Rules, which compose 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. Lv et al., 2006), Bayesian Network (BN), Dynamic Bayesian Networks (DBN) (e.g. Gong & Xiang, 2003; SanMiguel et al., 2011; SanMiguel & Martínez, 2012; Town, 2006), Hidden Markov Model (HMM) (e.g. Cuntoor, Yegnanarayana, & Chellappa, 2005; Hoey, Bertoldi, Poupart, & Mihailidis, 2007; Nguyen, Phung, Venkatesh, & Bui, 2005; Oliver & Horvitz, 2005; Oliver, Horvitz, & Garg, 2004; Patterson, Fox, Kautz, & Philipose, 2005), Conditional Random Fields (CRF) (e.g. Liao, Fox, & Kautz, 2006; Vail, Veloso, & Lafferty, 2007; Wu, Lian, & Hsu, 2007) 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 (Simon et al., 2010). They also can not 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. Ayers & Shah, 2001; Martinez-Tomas, Rincon, Bachiller, & Mira, 2008) and Decision Trees (e.g. Simon et al., 2010). FSM is a deterministic model and provide computationally efficient solutions. On the other hand, FSM can not have hidden states and can not 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. Ghanem, DeMenthon, Doermann, & Davis, 2004; Lavee, Rudzsky, & Rivlin, 2010), grammar models, constraint satisfaction (e.g. Akdemir, Turaga, & Chellappa, 2008; Fusier et al., 2007; Reddy, Gal, & Shieber, 2009), and logic-based approaches can be considered in this group. 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. Pei, Jia, & Zhu, 2011; Ryoo & Aggarwal, 2008, 2009) 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.

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. Romdhane, Boulay, Bremond, & Thonnat, 2011; Romdhane, Bremond, & Thonnat, 2010; Romdhane, Crispim, Bremond, & Thonnat, 2013) that try to compose probabilistic constraint satisfaction to add an uncertainty handling mechanism.

Logic-based models have well-defined understandable structure. In this type of approaches (e.g. Dousson, & Maigat, 2007; Shet, Harwood, & Davis, 2005), 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 can not 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 can not handle uncertainty. There are also some studies which can handle uncertainty (Hongeng, Nevatia, & Bremond, 2004). In these studies, probability is integrated to handle noisy conditions. Some authors also proposed new combined models in the literature. Markov Logic Network (MLN) (Richardson & Domingos, 2006) 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 Onal et al. (2013), Tran and Davis (2008), Biswas, Thrun, and Fujimura (2007), Gupta et al. (2007), Kembhavi, Yeh, and Davis (2010) and Helaoui, Niepert, and Stuckenschmidt (2011) 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.

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. 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. Artikis et al. (2010), Skarlatidis, Paliouras, Vouros, and Artikis (2011), Artikis, Sergot, and Paliouras (2015) and Skarlatidis et al. (2015) 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. Baxter, Robertson, & Lane, 2011; Baxter, Robertson, & Lane, 2010; Kuettel, Breitenstein, Gool, & Ferrari, 2010). 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. 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, Zhang et al. (2013), Hakeem and Shah (2007) and Brendel, Fern, and Todorovic (2011) are the studies in which event recognition processes are extended in an interval-based manner, along with MLN as in Song, Kautz, Li, and Luo (2013) and Morariu and Davis (2011).

Nowadays, methods that use Deep Learning become increasingly popular (e.g. Chong & Tay, 2015; Gan, Wang, Yang, Yeung, & Hauptmann, 2015; Jhuo & Lee, 2014; Xu, Ricci, Yan, Song, & Sebe, 2015). Methods based on Deep Learning have achieved promising performance in image classification and action recognition tasks and are generally used for anomaly detection.

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. Dubba, Santos, Cohn, & Hogg, 2011; Kardas, Ulusoy, & Cicekli, 2013; Patino & Ferryman, 2015).

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 Ribeiro and Santos-Victor (2005), Bayesian classifier method is used. Perez et al. (2007) 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 Lv et al. (2006), events are recognized using Bayesian Inference after trajectory smoothing is done using median filter. They use Pets2006 Dataset. In SanMiguel et al. (2011), Bayesian Inference is used to detect “LeaveObject”, “getobject”, “use object”, “walking” and “handup”. In both of these studies, there is no Event Model Learning. Instead, predefined models and thresholds are used.

In Simon et al. (2010), classifier-based approach is used for recognizing high-level events in CAVIAR Dataset. After clustering operation, classification is done using decision trees. “meeting”, “pocket picking”, “fighting”, “leaving bag”, “forbidden zone” are considered events. Created event models are not understandable as models defined using MLN.

In Romdhane et al. (2010) and Romdhane et al. (2013), 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 Artikis et al. (2010) event calculus is used. Event calculus can represent interval-based relations well but can not 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 Skarlatidis et al. (2011), 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 Skarlatidis et al. (2015), 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 Patino and Ferryman (2015), 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.

Tran and Davis (2008) is a MLN study to probabilistically infer activities in a parking lot. Domain knowledge is defined as MLN rules without learning. In Biswas et al. (2007), 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 Gupta et al. (2007), 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 Song et al. (2013), MLNs are used in an interval-based manner for cooking plan event such as “make tea” and “make coffee”. In Morariu and Davis (2011), Allen’s Interval Logic (Allen & Ferguson, 1994) 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 Zhang et al. (2013), 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. 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 understandable. In Brendel et al. (2011), probabilistic event logic (PEL) is presented, which uses weighted event-logic formulas to represent probabilistic constraints among events. They consider only the recognition of primitive events of basketball game.

In Dubba et al. (2011), ILP-based Event Model Learning is used. They use MLNs for event models and interval-based predicates can be defined. Experiments are done for only events in airport domain such as “aircraft arrival”, “positioning” and “departure”. This method is not suitable for domains in which there is no tree like object type hierarchy because of great increase in search space.

Baxter et al. (2011) 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 and some detailed surveys about methods for detection, recognition of video events, and video event modeling (e.g. Alevizos, Skarlatidis, Artikis, & Paliouras, 2015; Ballan, Bertini, Bimbo, Seidenari, & Serra, 2011; Ko, 2008; Lavee et al., 2009; Vishwakarma & Agrawal, 2013) 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 robust representational capability including semantic relations. It should be semantically meaningful for the user, it should also have learning ability from 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 recognition algorithms should have almost real time high performance. According to the studies surveyed, 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 effi-

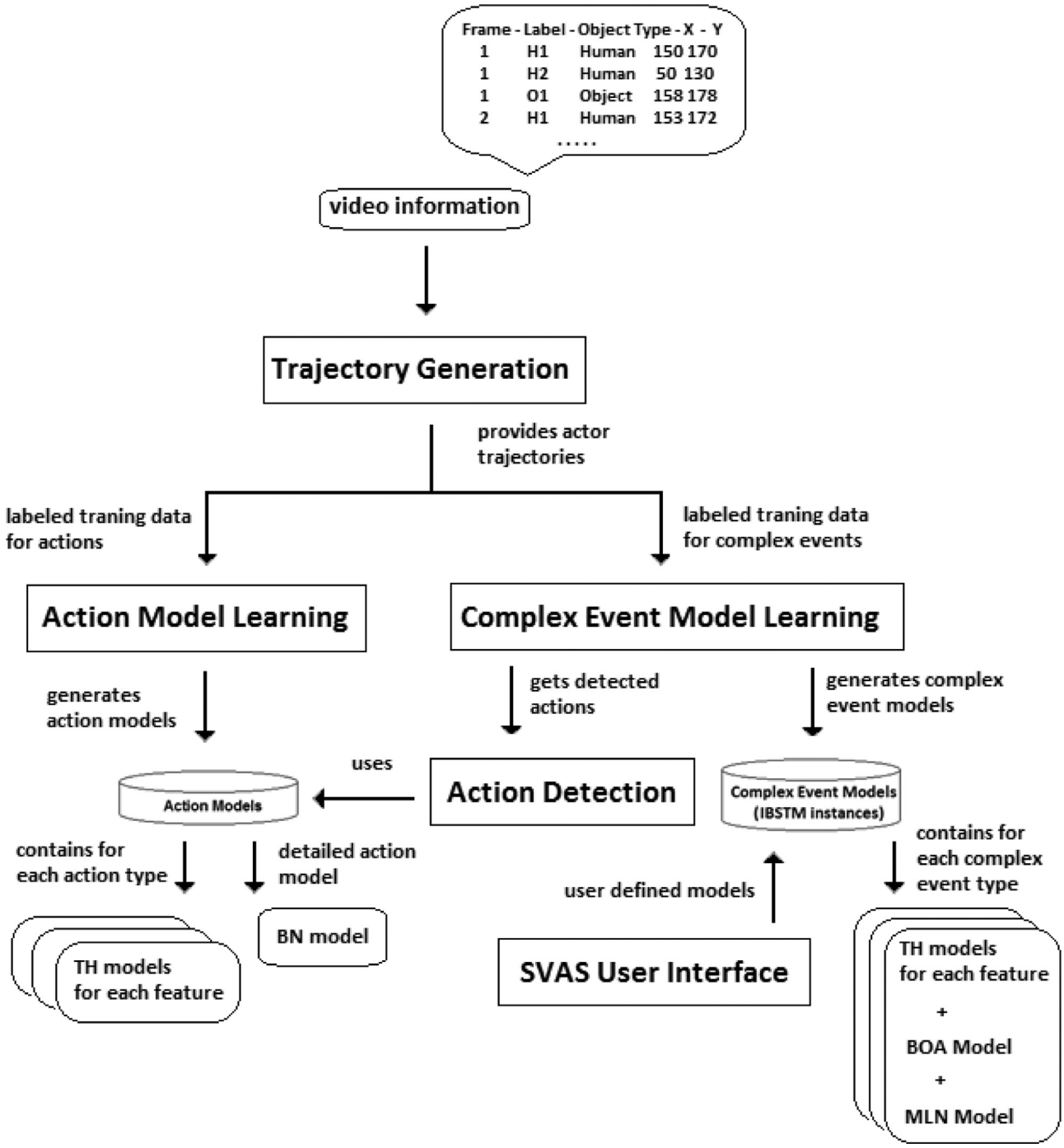

Fig. 1. Event Model Learning Process.

cient video analysis system for surveillance domain by considering video domain needs according to the survey.

3. SVAS architecture

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 (Fig. 1) and event detection (Fig. 2).

In Event Model Learning (Fig. 1), actor Trajectory Generation is the first operation using Trajectory Generation Module. Trajectory Generation Module parses and prepares video data for processing. Event Model Learning process 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.

In Fig. 1, 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 gen-

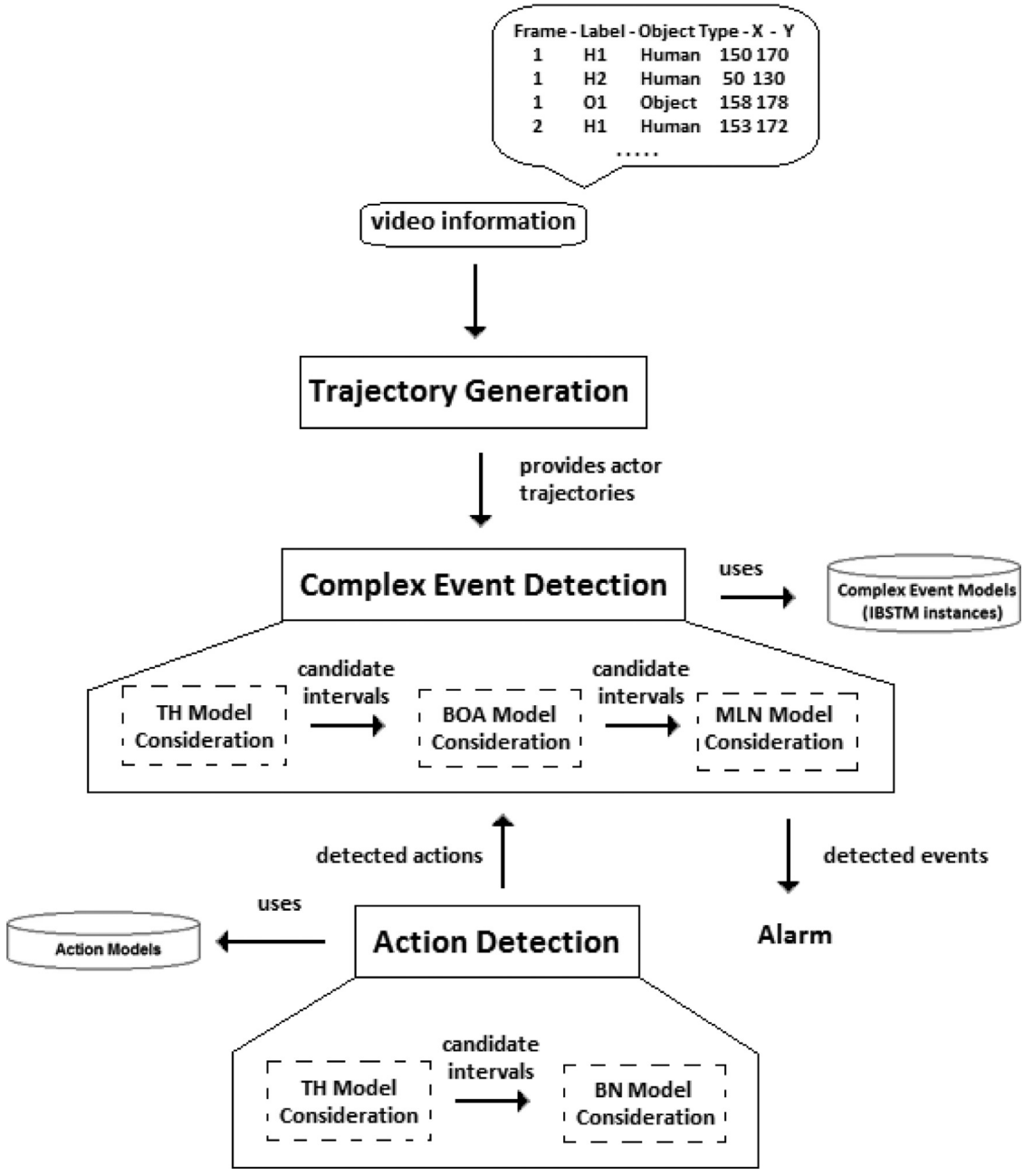

Fig. 2. Event Detection Process.

erated models via SVAS User Interface Module if desired. In addition, the user can create his/her own event models by using the same module.

In Event Detection Process (Fig. 2), 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 analyzed for each complex event type that is defined in the system. Alarms are generated for detected events.

SVAS has a high performance event detection capability due to its interval-based hierarchical manner. In Fig. 2, both Action Detection and Complex Event Detection include more than one model consideration. 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.

4. Trajectory Generation

Trajectory Generation generates actor trajectories from input videos for further processing. In this study, CAVIAR (Yang & Nevatia, 2012) and BEHAVE (Blunsden & Fisher, 2010) 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 Fig. 3:

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),

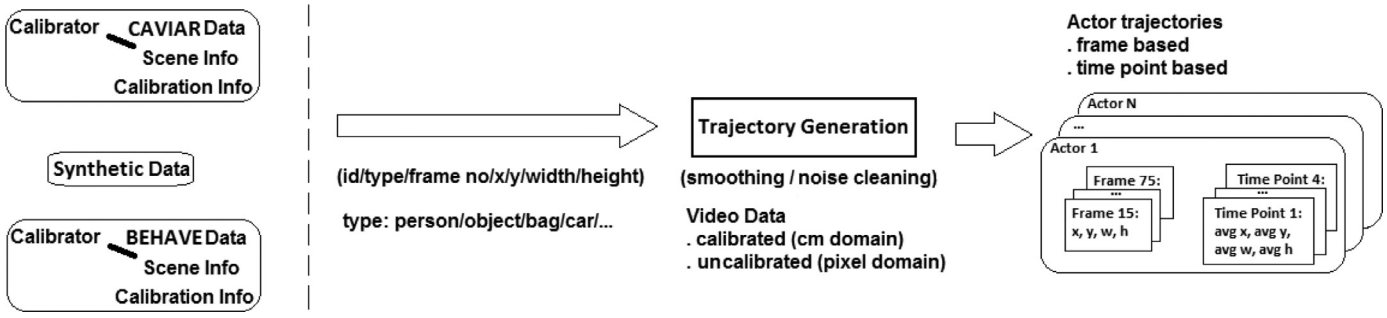

Fig. 3. Trajectory Generation.

width, height, type⟩ is obtained. The meanings of these attributes are as follows:

- 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 s. The number 15 was selected for grouping frames since actions within 0.5 s are generally visible to the human eye. This indicates that some features such as speed and direction can be considered at 0.5 s intervals.

SVAS can generate the scene structure using trajectories. Movable and occlusion areas are 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.

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. 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. 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 are 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.

5. Event Model Learning

An event is something happening in a location at a given time. 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. This process 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. SVAS does not need any predefined thresholds for scene or event type contrary to many studies in the literature (e.g. Antoniou, 2011; Lv et al., 2006; Neuhaus, 2005; Robertson et al., 2008; SanMiguel et al., 2011; Skarlatidis et al., 2015). It can learn required thresholds as Threshold Models.

In this section, Event Model Learning processes are presented in detail (see Fig. 1). 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 process, the input is the labeled training data. In Complex Event Model Learning process, 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 processes only require positive examples.

5.1. Action Model Learning

An action is a simple event performed by a single actor. Actions can be short body movements. Specifically, “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 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

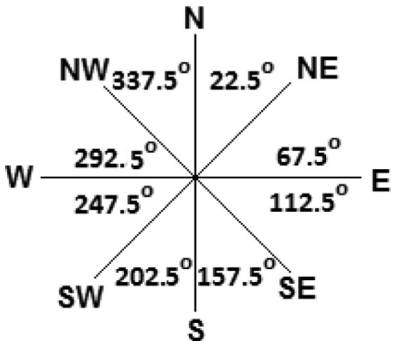

Fig. 4. Direction information.

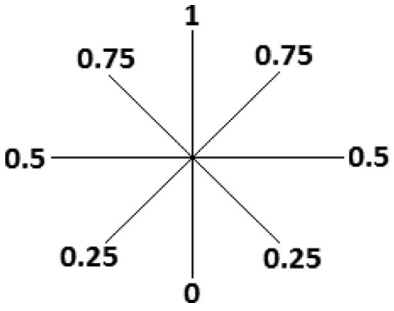

Fig. 5. Direction similarity degree.

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 Fig. 4. The direction information is prepared for slices of 45°. For example, the angle between 337.5° and 22.5° 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 Fig. 5.

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 Fig. 6.

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 Fig. 7. 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 can not 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 Fig. 8. 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 models are needed for a correct and efficient computation.

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:

$$W_{proportion} = 1 - [(\text{Test Value} - -\text{avg})/(\max - \text{avg})] * 0.5$$

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:

$$W_{frequency} = 0.5 + \left(\text{Value}_{frequency}/2 * \text{TotalFrequencyCount}\right)$$

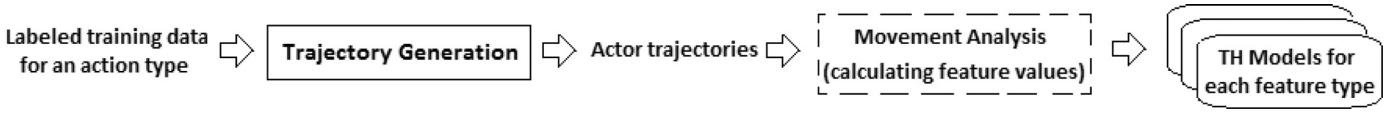

Fig. 6. Threshold Model (TH Model) learning.

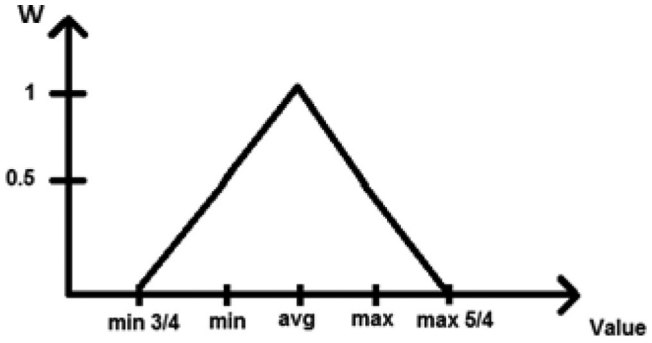

Fig. 7. Proportion model.

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

Fig. 8. Frequency table.

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.

$$\text{Weight Of Test Value} = \text{Max}\left(W_{proportion}, W_{frequency}\right)$$

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 using WEKA (Frank, Hall, & Witten, 2016) BayesNet classification algorithm. 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 shown in Fig. 1.

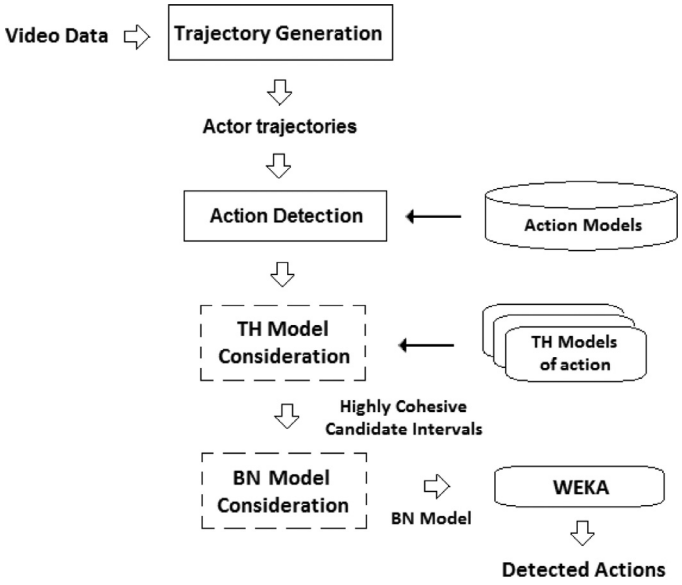

Fig. 9. Action Detection.

5.2. Action Detection

Action Detection is a process of finding actions in a video data using pre-learned action models. 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 Fig. 9.

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:

```
Input: VideoPart videoPart;
Output: DetectedActions
1. TimePoint[] timepoints = TrajetoryGenerator.createTimePoints(videoPart);
2. CandidateIntervals[] candidateIntervals = new CandidateIntervals[]{};
3. For each Action a in ActionListInTheSystem{
4.
candidateIntervals.addAll(findCandidateIntervals(a.getThresholdModelSet(),
timepoints));
5.}
6. TestInputForWekatestInputForWeka = generateBNTestData
(candidateIntervals);
7. Weka.BayesianNetworkConsideration(testInputForWeka, BayesianNetwork
ModelForActions);
8. DetectedActions[] detectedActions = ParseWekaResultsAnd
GetDetectedActions();
9. return detectedActions;
```

In detection phase, three seconds long video parts with 0.5 s 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 fea-

ture 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]. WEKA 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 WEKA 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:

```
Input:  ThresholdModelSet thresholdModelSet,
        TimePoint[] timepoints,
Output: CandidateIntervals
1.  CandidateTimePoint [] candidateTimePoints = new CandidateTimePoint[]{};
2.  for each TimePoint tp in timepoints{
3.          double weightvalue = 0;
4.          int featureCount = 0;
5.          For each Feature f in FeatureTypesInTheSystem {
6.                  ThresholdModel thresholdModel = thresholdModel
    Set.getRelatedThresholdModel(f);
7.                  if(thresholdModel.getType().isTimePointFeature()){
8.                  weightvalue += thresholdModel.calculateSimilarity
    Value(tp);
9.                  featureCount++;
10.                 }
11.         }
12.         weightvalue = weightvalue / featureCount;
13.         if(weightvalue > 0.4){
14.                 candidateTimePoints.add(tp);
15.         }
16.  }
17.  Intervals[] intervals = generateAllIntervalsFromConsecutive
    TimePoints(candidateTimePoints);
18.  Intervals[] candidateIntervals = new Intervals[]{};
19.  For each Interval inter in intervals{
20.         double weightvalue = 0;
21.         intfeatureCount = 0;
22.         For each feature f in FeatureTypesInTheSystem{
23.                 ThresholdModel thresholdModel = thresholdModel
    Set.getRelatedThresholdModel(f);
24.                 weightvalue += thresholdModel.calculateSimilarity
    Value(inter);
25.                 featureCount++;
26.         }
27.         weightvalue = weightvalue / featureCount;
28.         if(weightvalue > 0.4){
29.                 candidateIntervals.add(inter);
30.         }
31.  }
32.  candidateIntervals = prepareUnionsAndIntersections(candidateIntervals);
33.  return 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 sub-interval is kept and the interval which contains the sub-interval, is replaced with two separate sub-intervals 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:

$$\begin{aligned} \text{newWeight} = &(\text{occuredEvent\_i.getWeight}\ () * \\ &\text{occuredEvent\_i.getInterval}().\text{length}() \\ &+ \text{occuredEvent\_j.getWeight}\ () * \\ &\text{occuredEvent\_j.getInterval}().\text{length}()) / \\ &(\text{occuredEvent\_i.getInterval}().\text{length}() \\ &+ \text{occuredEvent\_j.getInterval}().\text{length}()) \end{aligned}$$

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

5.3. Complex Event Model Learning

In Complex Event Model Learning, complex event models are generated as IBSTM models which reflect spatio-temporal relations and Threshold Models of events according to the training data. Complex events are events which include more than one actor who reside in a determined closeness. Complex events are learned by using actor trajectories and spatial relations between them. They can be learned without any predefined values or thresholds. "Meet", "Fight", "Walk Together", "Run Together", "Follow", "Chase", "LeftObject", "Taken Object" are examples of complex 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, the properties of our Interval-based Spatio-Temporal Model are described; then the model learning process is presented.

5.3.1. 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 these basic properties.

1. Domain convenience: IBSTM is suitable for surveillance domain. Various events with different scenarios can be learned and de-

fined 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 (Allen & Ferguson, 1994) are defined in IBSTM. 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.

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 which 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.

5.3.2. 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 Fig. 1). 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 Fig. 5.

In the second step, Complex Event Model Learning Module determines actions in training data using Action Detection Module (Fig. 1). 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 and kept without considering temporal information 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 in a role-based manner. 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 as follows (Table 1):

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 by using Alchemy (Domingos, 2010) which is an open-source MLN software. For Alchemy, first, a file with db extension is created from the training data. By using “learnwts” command of Alchemy, weighted MLN file is created. These steps are summarized in a flow diagram in Fig. 10.

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 7.

6. Complex Event Detection

Complex Event Detection is a process of finding complex events in a video using pre-learned IBSTM event models described in the previous section. Fig. 11 shows the flow diagram of the operation.

Pseudocode of this operation is shown below:

```
Input: VideoPart videoPart;
Output: DetectedEvents
1.  TimePoint[] timepoints = TrajetoryGenerator.createTimePoints(videoPart);
2.  CandidateIntervals[] candidateIntervals = new CandidateIntervals[]{};
3.  PredicateList predicates = new PredicateList();
4.  For each ComplexEvent ce in ComplexEventListInTheSystem{
5.  THModelSet thModelSet = ce.getThresholdModelSet();
6.  candidateIntervals.addAll(findCandidateIntervalsForTHModel(thModelSet,
    timepoints));
7.  DetectedActions actions = detectActions(candidateIntervals);
8.  candidateIntervals = boaElimination(actions, candidateIntervals,
    ce.getBoAModel());
9.  predicates.addAll(createMLNPredicates(actions, candidateIntervals,
    ce.getMLNModel()));
10. }
11. actFile f = CreateMLNFactFile(predicates);
12. AlchemyConsiderationWithMLNModel(f);
13. DetectedEvents[] detectedEvents = parseAlchemyResultsAndGet
    DetectedComplexEvents();
14. return detectedEvents;
```

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 per-

Table 1
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)

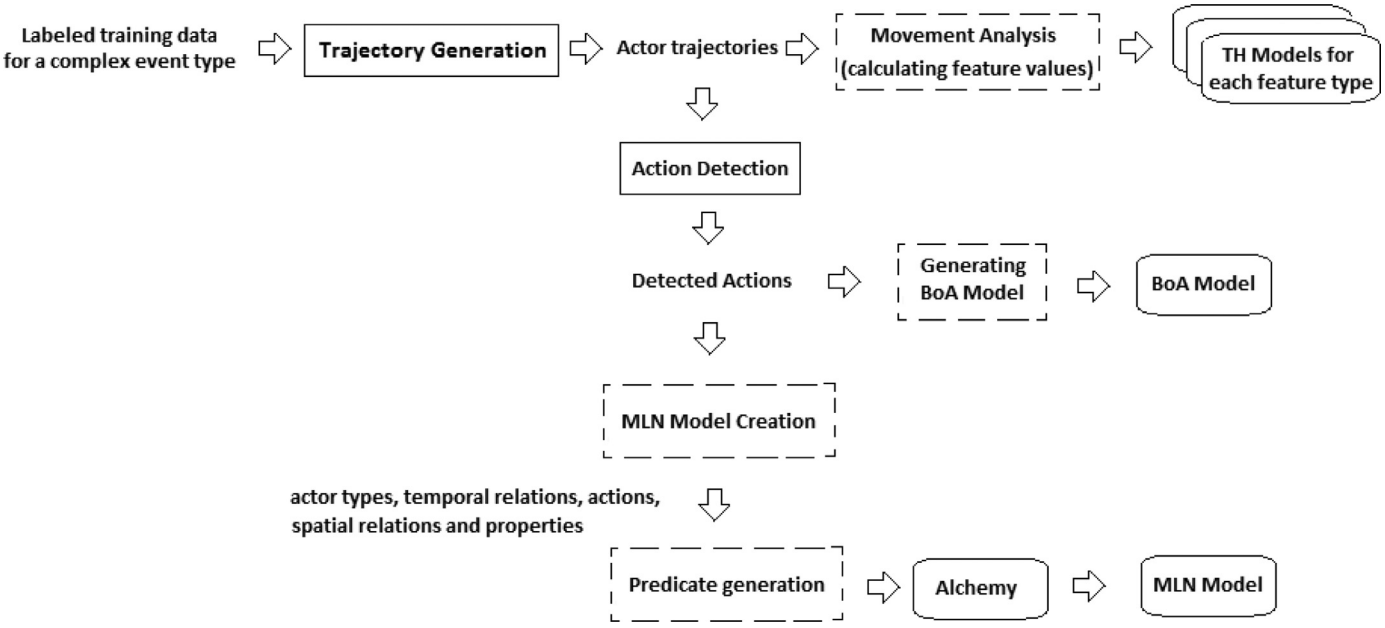

Fig. 10. Complex Event Model Learning.

formed 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 gives 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 which have higher weights as discussed in Section 5.2.

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. 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 variable count [Pseudocode: 9]. As a result, the performance of MLN algorithm increases. For the remaining candidate intervals, MLN fact file is created [Pseudocode: 11]. Remaining candidate intervals are queried with Alchemy by using MLN event models which are learned in the training phase and created MLN fact file. Intervals in

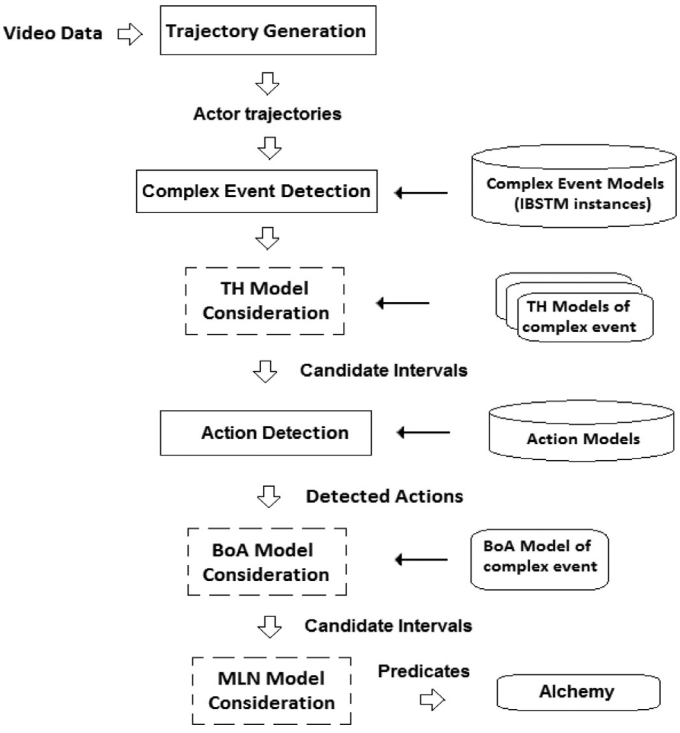

Fig. 11. Complex Event Detection.

which Alchemy gives higher results are chosen as complex event alarm [Pseudocode: 12–14].

7. Experimental evaluation

In this study, a series of experiments is made and proposed methods are evaluated using three datasets which are CAVIAR Dataset (Yang & Nevatia, 2012), BEHAVE Dataset (Blunsden & Fisher, 2010) 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 of surveillance domain (e.g. Al-Raziqi & Denzler, 2016; Artikis et al., 2010; Blunsden & Fisher, 2010; Elhamod & Levine, 2013; Muench et al., 2012; Patino & Ferryman, 2015; Perez et al., 2007; Simon et al., 2010; Skarlatidis et al., 2015; Yin et al., 2012; Zhang et al., 2012). Each video in these datasets is manually annotated to provide the ground truth and both CAVIAR and BEHAVE datasets are in XML format. Low-level image processing tasks are performed and video blobs are extracted. They provide actors and their pixel positions and blob width and height for each video frame. In SVAS, these datasets are parsed and become ready to use by Trajectory Generation Module according to their XML format.

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 Patino and Ferryman (2015), 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.

Table 2
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%

Table 3
Comparison of Action Detection evaluation with other studies in CAVIAR Dataset.

	(Simon et al., 2010)	(Perez et al., 2007)	Our results
Running	92.3%	0%	93.33%
İnactive	77.4%	85%	100%
Walking	77%	88%	73.34%
Active	85.9%	0%	93.33%

7.1. Evaluation of CAVIAR Dataset

CAVIAR Benchmark Dataset consists of 28 surveillance videos of a public space and contains several scenarios about "Fight", "Left-Object" and "Meet" complex events.

7.1.1. 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". "inactive" can be considered as a stand action and "active" can be considered as an instant move action. Action evaluations are done by using the 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, 14,134 and 1872 respectively. There are significant differences between CAVIAR Dataset sizes for action types. This leads to increase in confusion of actions. For this evaluation, as it is stated in Simon et al. (2010), it is required to fix the dataset size. Since the least 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 s long and for each other action types, 15 intervals which are 1 s long are selected from their dataset. As a result, dataset size for each action type became equal. Results of confusion matrix evaluation are shown in Table 2.

Results of studies using CAVIAR Dataset are shown in Table 3. Perez et al. (2007) has low accuracy values particularly for "active" and "running" action types. Simon et al. (2010) tries some 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 Simon et al. (2010), 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 4 are obtained.

7.1.2. 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

Table 4
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	14,134	11,806	2328	83.53%
Active	1872	1866	6	99.68

Table 5
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%

3 and for "left object" there are 4 event data. 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". Results of confusion matrix evaluation for CAVIAR complex events are shown in Table 5.

In Simon et al. (2010), balanced dataset is used for complex event evaluation. Comparing with Simon et al. (2010), 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 Simon et al. (2010).

In Patino and Ferryman (2015), only "meeting" event detection is studied and compared with Artikis et al. (2010). "meeting" accuracy of Artikis et al. (2010) and Patino and Ferryman (2015) are 67% and 89% respectively. Since Patino and Ferryman (2015) attacks only one complex event type, the accuracy result is high as expected. When the number of event types increases, confusion problems arise.

In Skarlatidis et al. (2015), complex event evaluation is provided without low-level Action Detection. Skarlatidis et al. (2015) uses low-level action values from CAVIAR ground truth. In Skarlatidis et al. (2015), 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 7.1.1 above. The precision of "left object" event is also higher in our system. On the other hand, results of Skarlatidis et al. (2015) 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 6.

7.2. Evaluation of BEHAVE Dataset

BEHAVE Dataset (Blunsden & Fisher, 2010) 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. This dataset is used in many studies in literature such as Elhamod and Levine (2013), Yin et al. (2012), Zhang et al. (2012), Al-Raziqi and Denzler (2016) and Muench et al. (2012). The numbers of datasets for each event type are listed in Table 7.

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

7.2.1. 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 8. Accuracy values are between 89% and 91%, which are very satisfactory.

7.2.2. 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 9.

In Blunsden and Fisher (2010), classification is provided using HMM without considering "Chase" and "RunTogether" events. Their average performance ranges from 80% to 90%. Elhamod and Levine (2013) considers only "Fight" and "Meet" events in BEHAVE dataset. Their accuracies are as follows: for "Meet" event, it is nearly 85% and for "Fight" event, it is nearly 70%. (Yin et al., 2012) evaluates "InGroup", "WalkTogether", "Fight" and "Split" events without confusion matrix method. Their accuracies are 94.3%, 92.1%, 95.1% and 93.1%, respectively. In Zhang et al. (2012), accuracies of detected events in confusion matrix are between 52% and 88%. In Al-Raziqi and Denzler (2016), accuracies of detected events in confusion matrix are between 50% and 80%. In Muench et al. (2012), "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 9. Our accuracies are between 75% and 100%. Our average performance is 90.57%. Related comparisons are shown in Table 10.

7.3. 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 learn-

Table 6
Comparison of Complex Event Detection evaluation with other studies in CAVIAR Dataset.

	(Simon et al., 2010)	(Artikis et al., 2010)	(Skarlatidis et al., 2015)	(Patino & Ferryman, 2015)	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%

Table 7
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

Table 8
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%

Table 9
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%

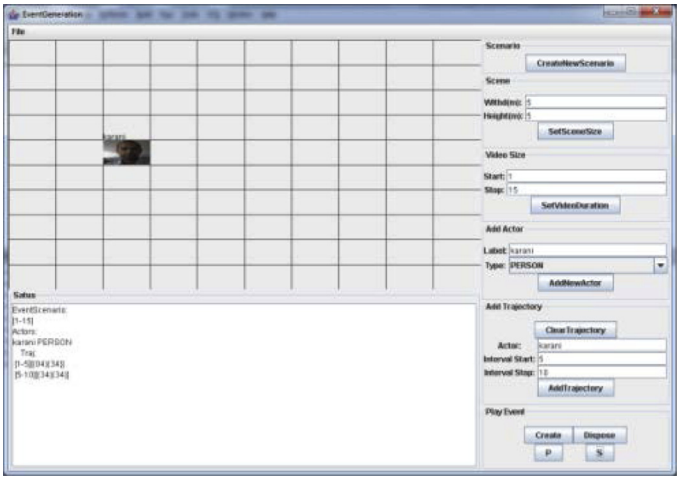

Fig. 12. User interface of Test Data Generation Tool.

Table 11
Evaluation of synthetic dataset.

Noise	Detection count	Hit ratio
No noise	135	100%
0.5 s noise per training data	112	82.9%
1 s noise per training data	98	72.6%
2 s noise per training data	67	49.6%
3 s noise per training data	0	0%

ing and testing phases of event detection process. In Fig. 12, user interface of Test Data Generation Tool is shown.

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

In Table 11, detection count decreases when noise duration increases. Since generated test data trajectories are maximum 5 s long, the impact of noise is very high. However, we can conclude that SVAS is robust for noises nearly 10% of trajectories.

7.4. 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 8GB RAM and Intel i5-4210 CPU.

In "Action Detection" phase, Threshold Model elimination increases the performance as shown in Fig. 13 and Table 12. One threshold query duration is nearly 0.06 ms, which can be considered as very fast. However, one Bayesian Network query duration is nearly between 4 ms and 9 ms 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 12, 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 10
Comparison of Complex Event Detection Evaluation with other studies in BEHAVE Dataset (IG: InGroup, WT: WalkTogether, C: Chase, Fi: Fight, RT: RunTogether).

	(Elhamod & Levine, 2013)	(Yin et al., 2012)	(Zhang et al., 2012)	(Al-Raziqi & Denzler, 2016)	(Muench et al., 2012)	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%

Table 12
Action Detection performance evaluation.

Test intervals count	Total TH query duration (ms)	Total BN query duration (ms)	Eliminated Interval Count By Th Model	Average 1 Th query duration (ms)	Average 1 BN query duration (ms)	Estimated performance gain (ms)
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

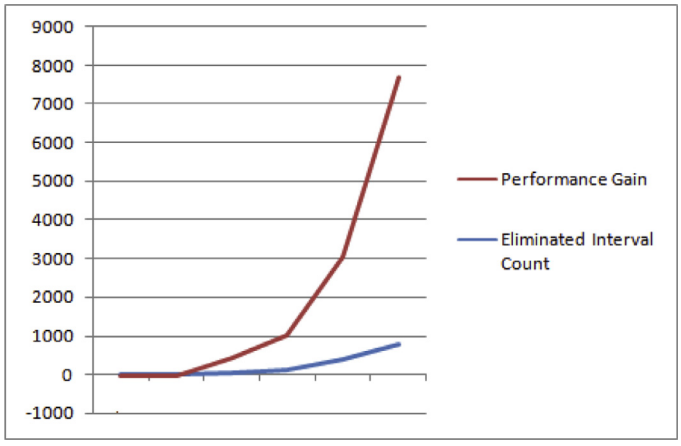

Fig. 13. Action Detection performance evaluation.

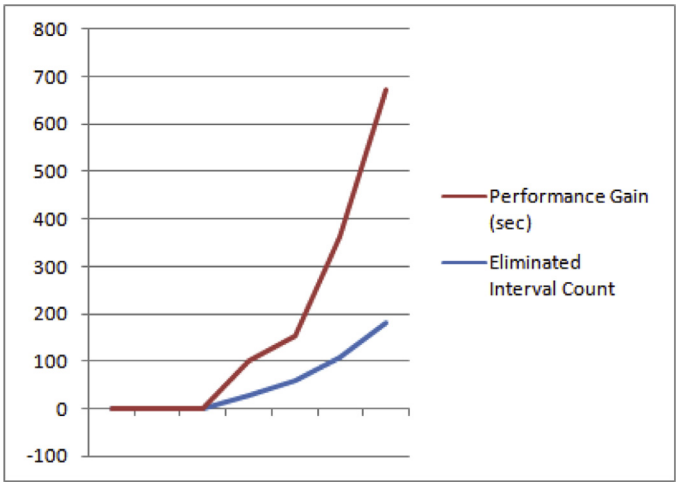

Fig. 14. Complex Event Detection performance evaluation.

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

Average 1 *Th query duration* :

$$\frac{\text{Total Th query duration}}{\text{Test intervals count} * \text{Action types}}$$

Average 1 *BN query duration* :

$$\frac{\text{Total BN query duration}}{\text{Test intervals count} - \text{Eliminated interval count by TH model}}$$

Estimated Performance gain :

(Eliminated Interval Count By Th model ∗
Average 1 BN query duration) − Total TH query duration

In Fig. 13, the performance gain increases when the number of eliminated intervals increases. This elimination power shows the necessity and importance of the Threshold Model.

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 Fig. 14 and Table 13. Performance gain of BoA Model is limited but its calculation is very fast such as almost 1 ms So, BoA Model is also useful.

In this case, performance gain is huge since the inference operation in MLN is nearly between 1.5 s and 4 s. 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 13, 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.

Formulas of calculations are as follows:

Total Eliminated Interval Count :
Eliminated Interval Count By Th model
+ Eliminated Interval Count By BoA model

Average 1 MLN query duration :

$$\frac{\text{Total MLN Query Duration}}{\text{Test intervals Count} - \text{Total Eliminated Interval Count}}$$

Total Elimination duration :
Total TH query duration + Total BoA query duration

Total Eliminated MLN query duration :
Average 1 MLN query duration ∗
Total Eliminated Interval Count

Estimated Performance gain for MLN Query Elimination :
Total Eliminated MLN query duration
− Total Elimination duration

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

Fig. 14 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.

Table 13
Complex Event Detection performance evaluation.

Test inter-vals count	Total TH query duration (ms)	Total Action Detection duration (ms)	Total BoA query duration (ms)	Eliminated Interval Count By Th model	Eliminated Interval Count By BoA model	Total MLN Query Duration (ms)	Average 1 MLN query duration (ms)	Estimated Performance gain for MLN Query Elimination (ms)
6	1	13	1	0	0	10,980	1830	−2
12	1	24	1	0	0	23,808	1984	−2
14	3	24	1	0	0	55,804	3986	−4
97	17	131	2	24	3	193,200	2760	74,501
223	34	319	6	55	3	268,125	1625	94,210
438	49	537	7	102	5	785,794	2374	253,962
721	54	817	12	176	6	1,449,371	2689	489,332

7.5. Qualitative evaluation

SVAS generates semantically meaningful event models in MLN format. Some of the generated complex event models are shown in Table 14. As shown in the table, the models are semantically consistent with expectations.

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 14 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 “LeftObject” complex event, where previous time periods of the event is considered as follows:

$$\begin{aligned}&\text{LeftObject}(a1, a2, t1, t2) :\\ &\quad \text{Object}(a1) \wedge \text{Person}(a2) \wedge \text{Stand}(a1, t1) \wedge \text{Stand}(a2, t1) \wedge\\ &\quad \text{Near}(a1, a2, t1) \wedge \text{Stand}(a1, t2) \wedge \text{Walk}(a2, t1) \wedge\\ &\quad \text{Far}(a1, a2, t2)(t2 > t1)\end{aligned}$$

8. Conclusion

In this paper, 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. Generated models are user understandable and manageable. In addition, SVAS does not need any predefined thresholds for scene or event model compared to many studies. A set of hybrid machine learning techniques including Threshold Models for features, Bayesian Networks, Bag of Actions, Highly Cohesive Intervals and Markov Logic Networks are proposed and used.

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 since it is based on the intervals instead of time points. In addition, Threshold Models and BoA Model 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 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, but 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

Table 14
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)
Taken Object(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)

camera and multi camera datasets and to include other actions and 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.

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.

The proposed Threshold Model algorithms are designed simple in this study to make them suitable for GPU programming. Implementing the Action Detection phase in GPU is another future work which may provide enhanced performance.

Acknowledgments

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.

References

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).

Alevizos, E., Skarlatidis, A., Artikis, A., & Paliouras, G. (2015). Complex event recognition under uncertainty: a short survey. *The workshop proceedings of the EDBT/ICDT joint conference*.

Allen, J. F., & Ferguson, G. (1994). Actions and events in interval temporal logic. *Journal of Logic and Computation, 4*(5), 531–579.

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).

Antoniou, G. (2011). Rule-Based Activity Recognition in Ambient Intelligence. Rule-based reasoning, programming, and applications. *RuleML*.

Artikis, A., Sergot, M., & Paliouras, G. (2010). A logic programming approach to activity recognition. In *Proceedings of ACM international workshop on events in multimedia*.

Artikis, A., Sergot, M., & Paliouras, G. (2015). An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering, 27*(4), 895–908.

Ayers, D., & Shah, M. (2001). Monitoring human behavior from video taken in an office environment. *Image and Vision Computing, 19*(12), 833–846.

Ballan, L., Bertini, M., Bimbo, A. D., Seidenari, L., & Serra, G. (2011). Event detection and recognition for semantic annotation of video. *Multimedia Tools and Applications, 51*, 279–302.

Baxter, R., Robertson, N. M., & Lane, D. (2010). Probabilistic behaviour signatures: Feature-based behaviour. *Information fusion (FUSION)*.

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*.

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

Biswas, R., Thrun, S., & Fujimura, K. (2007). Recognizing activities with multiple cues. In *Workshop on human motion: Vol. 4814* (pp. 255–270). Springer.

Blunsden, S. J., & Fisher, R. B. (2010). The BEHAVE video dataset: Ground truthed video for multi-person behavior classification. *Annals of the BMVA, 4*, 1–12. http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS. [Accessed: 05 01, 2016.].

Brendel, W., Fern, A., & Todorovic, S. (2011). Probabilistic event logic for interval-based event recognition. *IEEE computer vision and pattern recognition (CVPR)*.

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)*.

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).

Domingos, P. (2010). *The alchemy tutorial*. http://alchemy.cs.washington.edu/tutorial/tutorial.pdf [Accessed: 05 01, 2016].

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).

Dubba, K., Santos, P., Cohn, A., & Hogg, D. (2011). Probabilistic relational learning of event models from video. *The 21st international conference on inductive logicprogramming*.

Elhamod, M., & Levine, M. D. (2013). Automated real-time detection of potentially suspicious behavior in public transport areas. *IEEE Transactions on Intelligent Transportation Systems, 14*(2), 688–699.

Felzenswalb, P., McAllester, D., & Ramann, D. (2008). A discriminatively trained, multiscale, deformable partmodel. *IEEE CVPR*.

Frank, E., Hall, M. A., & Witten, I. H. (2016). The WEKA workbench. *Online appendix for "data mining: practical machine learning tools and techniques* (4th ed.). Morgan Kaufmann *http://www.cs.waikato.ac.nz/ml/weka*.

Fusier, F., Valentin, V., Bremond, F., Thonnat, M., Borg, M., Thirde, D., et al. (2007). Video understanding for complex activity recognition. *Machine Vision and Applications, 18*(3), 167–188.

Gan, C., Wang, N., Yang, Y., Yeung, D., & Hauptmann, A. G. (2015). DevNet: A deep event network for multimedia event detection and evidence recounting. *Computer vision and pattern recognition (CVPR)*.

Ghanem, N., DeMenthon, D., Doermann, D., & Davis, L. (2004). Representation and recognition of events in surveillance video using Petri nets. In *IEEE international conference on computer vision and pattern recognition workshop* (pp. 104–112).

Gong, S., & Xiang, T. (2003). Recognition of group activities using dynamic probabilistic networks. *The 9th international conference on computer vision*.

Gupta, H., Yu, L., Hakeem, A., Choe, T. E., Haering, N., & Locasto, M. (2007). Multimodal complex event detection framework forwide area surveillance. *Computer vision and pattern recognition workshops (CVPRW)*.

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).

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

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*.

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)*.

Hongeng, S., Nevatia, R., & Bremond, F. (2004). Video-based event recognition: Activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding, 96*(2), 129–162.

Jhuo, I., & Lee, D. T. (2014). Video event detection via multi-modality deep learning. *22nd international conference on pattern recognition*.

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

Kardas, K., Ulusoy, İ., & Cicekli, N. K. (2013). Learning complex event models using markov logic networks. *Multimedia and expo workshops (ICMEW), IEEE international conference on multimedia and expo*.

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*.

Ko, T. (2008). A survey on behavior analysis in video surveillance for homeland security applications. In *37th IEEE applied imagery pattern recognition workshop* (pp. 1–8).

Kuettel, D., Breitenstein, M., Gool, L., & Ferrari, V. (2010). What's going on? Discovering spatio-temporal dependencies in dynamic scenes. In *Proceedings IEEE conference on computer vision pattern recognition* (pp. 1951–1958).

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 on, 39*(5), 489–504.

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).

Liao, L., Fox, D., & Kautz, H. (2006). Hierarchical conditional random fields for GPS based activity recognition. *Robotics research the twelfth international symposium (ISRR-05)*.

Lv, F., Song, X., Wu, B., Singh, V. K., & Nevatia, R. (2006). Left-luggage detection using bayesian inference. In *Proceedings 9th IEEE international workshop performance evaluation of tracking and surveillance* (pp. 83–90).

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 Recognition Letter, 29*(8), 1117–1135.

Morariu, V. I., & Davis, L. S. (2011). Multi-agent event recognition in structured scenarios. *Computer vision and pattern recognition (CVPR)*.

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*.

Neuhaus, H. (2005). *A semantic concept for the mapping of low-level analysis data to high-level scene description*.

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).

Oliver, N., & Horvitz, E. (2005). A comparison of HMMs and dynamic bayesian networks for recognizing office activities. In *International conference on user modeling: Vol. 3538* (pp. 199–209).

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

Onal, I., Kardas, K., Rezaeitabar, Y., Bayram, U., Bal, M., Ulusoy, İ., et al. (2013). A framework for detecting complex events in surveillance videos. *Multimedia and expo workshops (ICMEW), IEEE international conference on multimedia and expo*.

Patino, L., & Ferryman, J. (2015). Meeting detection in video through semantic analysis. In *Proceedings of 12th IEEE international conference on advanced video and signal based surveillance (AVSS)*.

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.

Pei, M., Jia, Y., & Zhu, S. (2011). Parsing video events with goal inference and intent prediction. *IEEE international conference on computer vision*.

Perez, O., Piccardi, M., Garcia, J., & Molina, J. M. (2007). Comparison of classifiers for human activity recognition. In *Lecture notes in computer science: Vol. 4528* (pp. 192–201).

Reddy, S., Gal, Y., & Shieber, S. (2009). In *Recognition of users activities using constraint satisfaction: Vol. 5535* (pp. 415–421). Berlin / Heidelberg: Springer.

Ribeiro, P. C., & Santos-Victor, J. (2005). Human activity recognition from video: Modeling, feature, selection and classification arquitecture, HAREM. *International workshop on human activity recognition and modelling*.

Richardson, M., & Domingos, P. (2006). Markov Logic. *Machine Learning, Vol. 62*, 107–136.

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

Romdhane, R., Boulay, B., Bremond, F., & Thonnat, M. (2011). Probabilistic recognition of complex event. *ICVS 2011: 8th international conference on computer vision systems*.

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).

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)*.

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*.

Ryoo, M. S., & Aggarwal, J. K. (2009). Stochastic representation and recognition of high-level group activities. *Computer vision and pattern recognition workshops*.

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*.

SanMiguel, J. C., & Martínez, J. M. (2012). *A semantic-based probabilistic approach for real-time video event recognition*.

Shet, V., Harwood, D., & Davis, L. (2005). VidMAP: Video monitoring of activity with Prolog. *Advanced video and signal based surveillance IEEE*.

Simon, C., Meessen, J., & De Vleeschouwer, C. (2010). Visual event recognition using decision trees. *Multimedia Tools and Applications, 50*(1), 95–121.

Skarlatidis, A., Artikis, A., Filippou, J., & Paliouras, G. (2015). A probabilistic logic programming event calculus. *Theory and Practice of Logic Programming, 15*(2), 213–245.

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*.

Song, Y. C., Kautz, H., Li, Y., & Luo, J. (2013). A general framework for recognizing complex events in markov logic. *AAAI workshop: Statistical relational artificial intelligence*.

Tian, Y., Feris, R., Liu, H., Humpapur, A., & Sun, M. (2010). *Robust detection of abandoned and removed objects in complex surveillance videos*.

Town, C. (2006). Ontological inference for image and video analysis. *Machine Vision and Applications, 17*(2), 94–115 4.

Tran, S. D., & Davis, L. S. (2008). Event modeling and recognition using Markov logic networks. In *Proceedings of the 10th ECCV: Part II* (pp. 610–623). Springer-Verlag.

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)*.

Vishwakarma, S., & Agrawal, A. (2013). A survey on activity recognition and behavior understanding in video surveillance. *The Visual Computer, 29*(10), 983–1009.

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*.

Xu, D., Ricci, E., Yan, Y., Song, J., & Sebe, N. (2015). Learning deep representations of appearance and motion for anomalous event detection. *Computer Vision and Pattern Recognition (CVPR)*.

Yang, B., & Nevatia, R.. *Multi-target tracking by online learning of non-linear motion patterns and robust appearance models* [Accessed: 05 01, 2016.] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/.

Yildirim, Y., Yazici, A., & Yilmaz, T. (2013). Automatic semantic content extraction in videos using a fuzzy ontology and rule-based model. *IEEE Transactions on Knowledge and Data Engineering, 25*(1), 47–61.

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).

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). IEEE.

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), 2468–2483.

Zhao, T., & Nevatia, R. (2004). Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(9), 1208–1221.