Video-Based Event Recognition: Activity Representation and Probabilistic Recognition Methods

Somboon Hongeng, Ram Nevatia and Francois Bremond *

University of Southern California
Institute for Robotics and Intelligent Systems
Los Angeles, California 90089

E-mail:{hongeng| nevatia}@iris.usc.edu

* Present Contact: INRIA Project ORION, 2004 Route des Lucioles BP 93, 06902 Sophia Antipolis Cedex, France

Abstract

We present a new representation and recognition method for human activities. An activity is considered to be composed of action threads, each thread being executed by a single actor. A single thread action is represented by a stochastic finite automaton of event states, which are recognized from the characteristics of the trajectory and shape of moving blob of the actor using Bayesian methods. A multi-agent event is composed of several action threads related by temporal constraints. Multi-agent events are recognized by propagating the constraints and likelihood of event threads in a temporal logic network. We present results on real-world data and performance characterization on perturbed data.

Key words: video-based event detection, event mining, activity recognition

1 Introduction

Automatic event detection in video streams is gaining attention in the computer vision research community due to the needs of many applications such as surveillance for security, video content understanding and human-computer interaction. The type of event to be recognized can vary from a small scale action such as facial expressions, hand gestures and human poses to a large scale activity that may involve a physical interaction among locomotory objects moving around in the scene for a long period of time. There also may be interactions between moving objects and other objects in the scene, requiring static scene understanding. Addressing all the issues in event detection is thus enormously challenging and a major undertaking. In this paper, we focus on the detection of large scale activities where some knowledge of the scene (e.g., the characteristics of the objects in the environment) is known. One characteristic of activities of our interest is that they exhibit some specific patterns of whole-body motion. For example, consider a group of people stealing luggage left unattended by the owners. One particular pattern of the *"stealing"* event may be: two persons approach the owners and obstruct the view of the luggage, while another person takes the luggage. In the following, the word "event" and "activity" are used to refer to a large scale activity.

The task of activity recognition is to bridge the gap between numerical pixel level data and a high level abstract activity description. A common approach involves first detecting and tracking moving object features from image sequences. The goal of this step is to transform pixel level data into low-level features that are more appropriate for activity analysis. From the tracked features, the type of moving objects and their spatio-temporal interaction are then analyzed [1–4]. There are several challenges that need to be addressed to achieve this task:

- Motion detection and object tracking from real video data are often unstable due to poor video quality, shadows, occlusion and so on. A single view constraint common to many applications further complicates these problems.
- The interpretation of low-level features (e.g., appearance of objects) may be dependent on the view point.
- There is spatio-temporal variation in the execution style of the same activity by different actors, leading to a variety of temporal durations.
- Repeated performance by the same individual can vary in appearance.
- Similar motion patterns may be caused by different activities.

Therefore, there is need for a generic activity representation as well as a robust recognition mechanism that handle both data and event variations. The representation must be able to describe a simple action as well as a complicated,

cooperative task by several actors. For a pragmatic system, the representation should also be easily modified and extended by a user. Recognition methods must handle the probabilities accurately at all processing levels.

A large number of activity detection systems have been developed in the last decades. Details of some of the current approaches are given in section 2. One deficiency of most approaches is that they are developed for events that suit the goal in a particular domain and lack genericity. Many event representations (e.g., image-pixel based representation) cannot be extended easily. Most event detection algorithms are only for simple events (e.g., *"walking"* or *"running"*) performed by a single actor, or for specific movements such as periodic motion. Some of them rely on the accuracy of motion sensors and do not provide a measure of confidence of the results, which is crucial for discriminating similar events in a noisy environment.

In this paper, we present a system that overcomes some of these deficiencies. We model scenario events from shape and trajectory features using a hierarchical activity representation extended from [5], where events are organized into several layers of abstraction, providing flexibility and modularity in modeling scheme. The event recognition methods described in [5] is based on a heuristic method and could not handle multiple-actor events. In this paper, an event is considered to be composed of action threads, each thread being executed by a single actor. A single thread action is represented by a stochastic finite automaton of event states, which are recognized from the characteristics of the trajectory and shape of the moving blob of the actor based on rigourous Bayesian analysis. A multi-agent event is represented by an event graph composed of several action threads related by logical and temporal constraints. Multi-agent events are recognized by propagating the constraints and likelihood of event threads in the event graph. Our earlier papers [6,4] have described these components to some extent. This paper integrates all the materials and provides more details and performance evaluation.

The organization of the paper is as follows: Related work is discussed in section 2. An overview of our event detection system is described in section 3. Our tracking approach based on ground plane location is in section 4. The extended hierarchical representation is in section 5. Algorithm for recognizing scenarios is described in detail including experimental results in section 6 and 7. Performance characterization of the algorithm is in section 8.

2 Related Work

During the last decade, there has been a significant amount of event understanding research in various application domains [7,8]. A review of the current

approaches in motion analysis can be found in [9]. Most of the current approaches to activity recognition are composed of defining models for specific activity types that suit the goal in a particular domain and developing procedural recognition methods. In [10], simple periodic events (e.g., walking) are recognized by constructing dynamic models of the periodic pattern of people's movements and is highly dependent on the robustness of the tracking.

Bayesian networks have been used to recognize static postures (e.g., *"standing close to a car"*) or simple events (e.g., *"sitting"*) from visual evidence gathered during one video frame [1,11,3]. The use of Bayesian networks in these approaches differs in the way they are applied (e.g., what data is used as evidential input and how this data is computed, the structures of the networks, etc.). One of the limitations of using Bayesian networks is that they are not suitable for encoding the dynamic of long term activities.

Inspired by applications in speech recognition, *Hidden Markov Model*(HMM) formalism has been extensively applied to activity recognition [12–16]. In one of the earlier attempts [12], discrete HMMs are used as representation of tennis strokes. Feature vectors of a tennis stroke are defined directly from the pixel values of a subsampled image. A tennis stroke is recognized by computing the probability that the model produces the sequence of feature vectors observed during the action. Parameterized-HMM [14] and coupled-HMM [16] were introduced later to recognize more complex events such as an interaction of two mobile objects. In [2], a stochastic context-free grammar parsing algorithm is used to compute the probability of a temporally consistent sequence of primitive actions recognized by HMMs. Even though HMMs are robust against various temporal segmentations of events, the structure and probability distributions are not transparent and need to be learned using iterative methods. For complex events, such networks and the parameter space to be learned may become prohibitively large.

There is only a limited amount of research on multi-agent events [17,3] as the tracking of multiple objects in a natural scene is difficult and it is difficult to maintain the parameters of the fine temporal granularity of the event models such as HMMs. In [3], a complicated Bayesian network is defined together with specific functions to evaluate some temporal relationships among events (e.g., *before* and *around*) to recognize actions involving multiple agents tracked manually in a football match. Generalizing this system for other tasks than those of a football match may require substantial development.

In recent years, there has also been a significant amount of work toward the fusion of multimodal information (e.g., color, motion, acoustic, speech, text) for event and action recognition. Most approaches [18–20] rely on contextual knowledge and are limited to specific domains (e.g., offices, classrooms, TV programs).

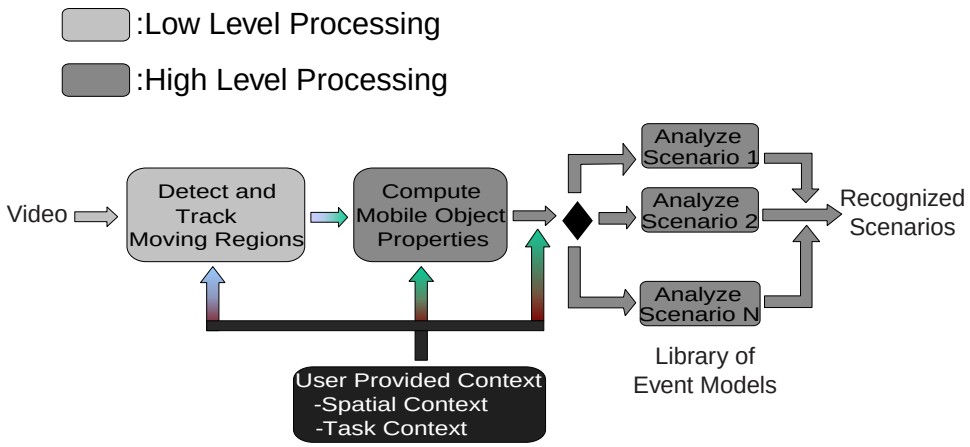


Fig. 1. *Overview of the system*

Our approach is closely related to the work by Ivanov et. al. [2] in the sense that external knowledge about the problem domain is incorporated into the expected structure of the activity model. In [5], we introduced a hierarchical activity representation that allows the recognition of a series of actions performed by a single mobile object. Image features are linked explicitly to a symbolic notion of activity through several layers of more abstract activity descriptions. Rule-based methods are used to approximate the belief of the occurrence of activities. A set of rules are defined at each recognition step to verify whether the properties of mobile objects match their expected distributions (represented by a mean and a variance) for a particular action or event. This method often involves a careful hand-tuning of parameters such as threshold values. In this paper, we extend the representation described in [5] and present a recognition algorithm that computes the probabilities of activities in a more rigorous way using Bayesian and logical methods.

3 Overview of the System

Figure 1 shows schematically our approach to recognize the behavior of moving objects from an image sequence and available context. **Context** consists of associated information, other than the sensed data, that is useful for activity recognition such as a spatial map and prior activity expectation (task context). Our system is composed of two modules: 1) Motion detection and tracking (shown in blue); 2) Event Analysis (shown in green).

Our tracking system is augmented from a graph-based moving blob tracking system described in [5]. A stationary single-view camera is used in our experiments. Background pixels are learned statistically in real time from the input video streams. Moving regions are segmented from background by detecting changes in the intensity. Knowledge of the ground plane, acquired as a spatial

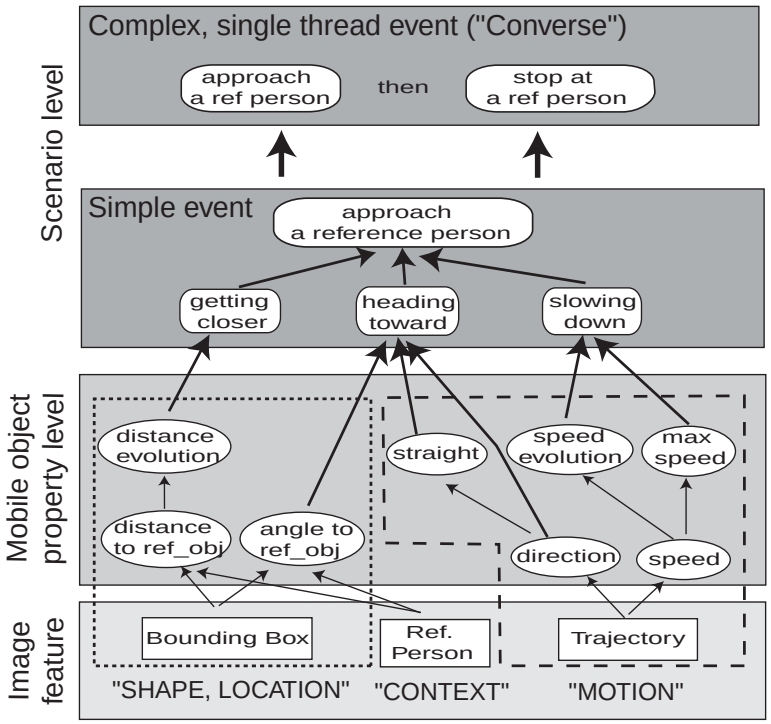


Fig. 2. *A representation of the complex event "converse".*

context, is used to filter moving regions and track objects robustly. Shape and trajectory features of moving objects are then computed by some low level image processing routines and used to infer the probability of potential events defined in a library of scenario event models.

Events in the scenario library are modeled using a hierarchical event representation, in which a hierarchy of **entities** is defined to bridge the gap between a high level event description and the pixel level information. Figure 2 shows an example of this representation for the event *"converse"*, which is described as *"a person approaches a reference person from a distance and then stops at the reference person when he arrives"*. **Image features** are defined at the lowest layer of the event representation. Several layers of more abstract **mobile object properties** and **scenarios** are then constructed explicitly by users to describe a more complex and abstract activity shown at the highest layer.

Mobile object properties are general properties of a mobile object that are computed over a few frames. Some properties can be elementary such as width, height, color histogram or texture while the others can be complex (e.g., a graph description for the shape of an object). Properties can also be defined with regard to the context (e.g., *"locate in the security area"*). In figure 2, mobile object properties are defined based on spatio-temporal characteristics of the corresponding shapes of moving blobs (shown in light blue) and the trajectories (shown in dark blue). The links between a mobile object property at a higher layer to a set of properties at the lower layers represent some

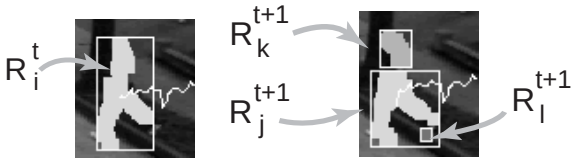
relation between them (e.g., taking a ratio of width and height properties to compute the aspect ratio of the shape of mobile objects). Typically, approximately three layers of properties are defined and used in a broad spectrum of applications. A filtering function and a mean function are applied to property values collected over time to minimize the errors caused by environmental and sensor noise.

Scenarios correspond to long-term activities described by the classes of moving objects (e.g., human, car or suitcase) and the event in which they are involved. Both the class of an object and the event have a confidence value (or a probability distribution) attached to them based on statistical analysis. Scenarios are defined from a set of properties or sub-scenarios. The structure of a scenario is thus hierarchical. We classify scenario events into a *single thread* and *multiple thread* event. In a **single thread event**, relevant actions occur along a linear time scale. Single thread events are further categorized into a simple or complex event. Simple events are defined as a short coherent unit of movement (e.g., *"approaching a reference person"*) and can be verified from a set of sub-events (*"getting closer to the reference person"*, *"heading toward"*, etc.) and mobile object properties. Complex events are a linearly ordered time sequence of simple events (or other complex events), requiring a long term evaluation of sub-events. In figure 2, the complex event *"converse"* is a sequential occurrence of *"approach a reference person"* and *"stop at the reference person"*. A **multiple thread event** is composed of several action threads, possibly performed by several actors. These action threads are related by some logical and time relations. In a typical application, there are about two to four layers of single-thread events and another two to four layers of multiple-thread events. Since our event representation at the scenario level maps closely to how human would describe events, little expertise is expected from the users. The users only need to have basic understanding of our event taxonomy.

Recognition process begins with the evaluation of the evidence (image features) and the computation of the probabilities of simple events at each time frame based on Bayesian analysis. The probabilities of simple events are then combined in a long term to recognize and segment complex events. Multiple thread events are recognized by combining the probabilities of complex event threads whose temporal segmentations satisfy the logical and time constraints. We describe in detail each component of our system in the following.

4 Detection and Tracking

Activity recognition by computer involves the analysis of the spatio-temporal interaction among the trajectories of moving objects [3,6,1]. Robust detection



(a) frame 74 (b) frame 75

Fig. 3. *Splitting of moving regions and noise.*

and tracking of moving objects from an image sequence is therefore an important key to reliable activity recognition. In the case of a static camera, detection of moving regions is relatively easier to perform, often based on background modeling and foreground segmentation. However, noise, shadows and reflections often arise in real sequences, causing detection to be unstable. For example, moving regions belonging to the same object may not connect or may merge with some unrelated regions. Tracking moving objects involves making hypotheses about the shapes of the objects from such unstable moving regions and track them correctly in the presence of partial or total occlusions. If some knowledge about the objects being tracked or about the scene is available, tracking can be simplified [21]. Otherwise, correspondence between regions must be established based on pixel level information such as shape and texture [5]. Such auxiliary knowledge other than the sensed data is called **context**. In many applications, a large amount of context is often available. In this paper, we demonstrate the use of the ground plane information as a constraint to achieve robust tracking.

Robust tracking often requires an object model and a sophisticated optimization process [22]. In the case that a model is not available or the size of the image of an object is too small, tracking must rely on the spatial and temporal correspondence between low level image features of moving regions. One difficulty is that the same moving regions at different times may split in several parts or merge with some other objects nearby due to noise, occlusion and low contrast. Figure 3 illustrates one of these problems, where the moving region ${R_i}^t$ at time t (a human shape) splits into two smaller regions ${R_j}^{t+1}$ and ${R_k}^{t+1}$ and noise detected at time $t+1$. The image correlation between the moving region ${R_i}^t$ and ${R_j}^{t+1}$ or ${R_k}^{t+1}$ by itself is often low and creates an incorrect trajectory. Filtering moving regions is therefore an important step of a reliable trajectory computation.

4.1 Ground Plane Assumption for Filtering

Let us assume that objects move along a known ground plane. An estimate of the ground plane location of the lowest point of a moving region can be used as a spatial constraint to find the best candidate region that corresponds

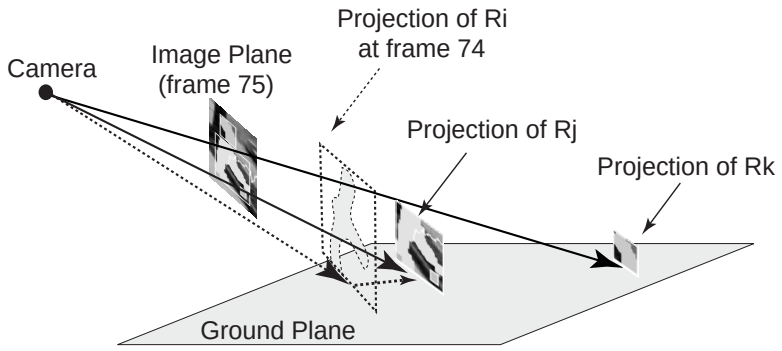


Fig. 4. *Projection of the bottom points of moving regions on to the ground plane.*

to ${R_i}^t$. This is illustrated in figure 4. The dotted line shows the projection of moving region ${R_i}^t$ of frame 74 on the ground plane, whose correspondence is to be found at frame 75. The solid lines show the projection of two moving regions detected at frame 75. Given that objects move on the ground, ${R_k}^{t+1}$ is unlikely to correspond to ${R_i}^t$ (the head blob) as they would be located too far apart on the ground plane. The best candidate region ${R_j}^{t+1}$ (the body blob) can then be selected as the most likely region correspondence accordingly. Other moving blobs for which no correspondences are made, are tracked by their relationship to the body blob.

To accurately compute the world coordinates of a point on an image plane, we need the parameters of a camera model. However, if we only need to estimate the 3D locations of points of moving regions on a ground plane, we can choose the world coordinate system such that $Z = 0$ as we are only interested in (X, Y) positions. This reduces the 3x4 perspective camera transformation matrix to a 3x3 projective transformation matrix (or plane homography) as follows:

$$\begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} = \begin{bmatrix} h_{11}h_{12}h_{13} \\ h_{21}h_{22}h_{23} \\ h_{31}h_{32}h_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

.

Since the non-singular homogeneous matrix H has 8 degrees of freedom, four or more points correspondence (x, y) to (X, Y) are enough to determine it uniquely, where $x = x_1/x_3$ and $y = x_2/x_3$. For the scene shown in figure 5 where the distance of an object from the camera is approximately 37 meters, we collected 8 points correspondence and solved for H with an error margin of $15cm$.

4.2 Merging Regions Using K-S Statistics

The best region candidate chosen based on a ground plane location may not be of the correct shape as shown by R_j. It may need to be merged with other sub-regions subject to some similarity measure. This merging process is iterated until no further merging occurs. In the final step, regions that are too small to represent valid objects can be disregarded.

The merging process between the best candidate sub-region ${R_j}^{t+1}$ and sub-region ${R_k}^{t+1}$ is performed as follows. We first make a hypothesis that both regions belong to the same object represented by $R_{j,k}^{t+1}$ and that their pixel data are drawn from the same distribution as those of ${R_i}^t$. To test this hypothesis, we compute the spatial similarity between $R_{j,k}^{t+1}$ and ${R_i}^t$. We base the test on the distribution of gray intensity. The Kolmogorov-Smirnov (K-S) statistics [23] provides a simple measure of the overall difference between two cumulative distribution functions. Let us assume that the intensity distribution of a region R_m is modeled by a Gaussian distribution ($N(x, \mu_m, \sigma_m)$). K-S statistics (D) of two regions R_m and R_n can be computed as follows:

$$\begin{aligned} D(R_m, R_n) = \max_{0 \le x \le 255} | \int_0^x \frac{1}{\sqrt{2\pi}\sigma_m} e^{-\frac{(x-\mu_m)^2}{2\sigma_m^2}} dx| \\ - | \int_0^x \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(x-\mu_n)^2}{2\sigma_n^2}} dx| \end{aligned} \quad (1)$$

The significance level of D is then computed as $Q_{ks}([N_e + 0.12 + 0.11/\sqrt{N_e}]D)$, where N_e is the effective area of the regions ($N_e = \frac{size(R_m) * size(R_n)}{size(R_m) + size(R_n)}$) and

$$Q_{ks}(\lambda) = 2\Sigma_{j=1}^{\infty}(-1)^{j-1} e^{-2j^2\lambda^2} \quad (2)$$

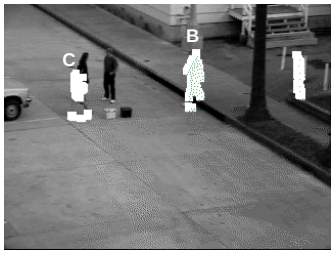
Regions are merged if the significance level and the K-S statistics of ${R_i}^t$ and $R_{j,k}^{t+1}$ are lower than those of ${R_i}^t$ and ${R_j}^{t+1}$. If the type of objects are known (e.g., a human), a constraint on the size of the moving regions after merging can also be used as a criteria to reject incorrect hypotheses.

Figure 5 shows the detection of moving regions of the "stealing" sequence at different times. To track objects in the sequence, a graph representation is used [5]. Figure 6 shows the graphs used for tracking object "D" before and after applying our filtering process. The nodes at each layer represent the moving regions detected at one video frame. An edge indicates a hypothesized correspondence between two regions of different frames. The nodes that are linked from the same parent represent the moving regions detected within the neighborhood of the parent node. The red nodes in the figure show moving regions of another object being tracked nearby. The numbers associated with

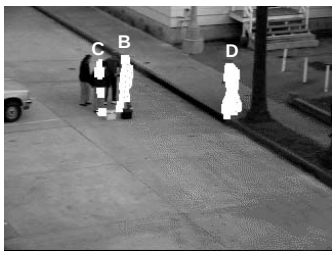
(a) frame 104

(b) frame 241

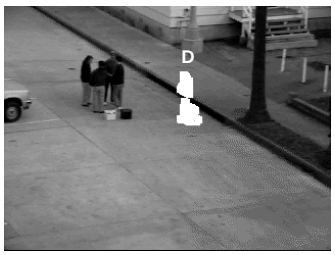
(c) frame 311



(d) frame 340

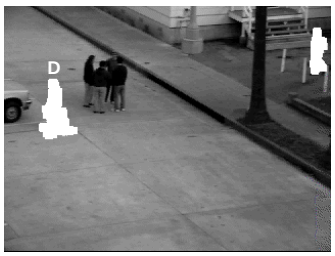


(e) frame 405



(f) frame 436

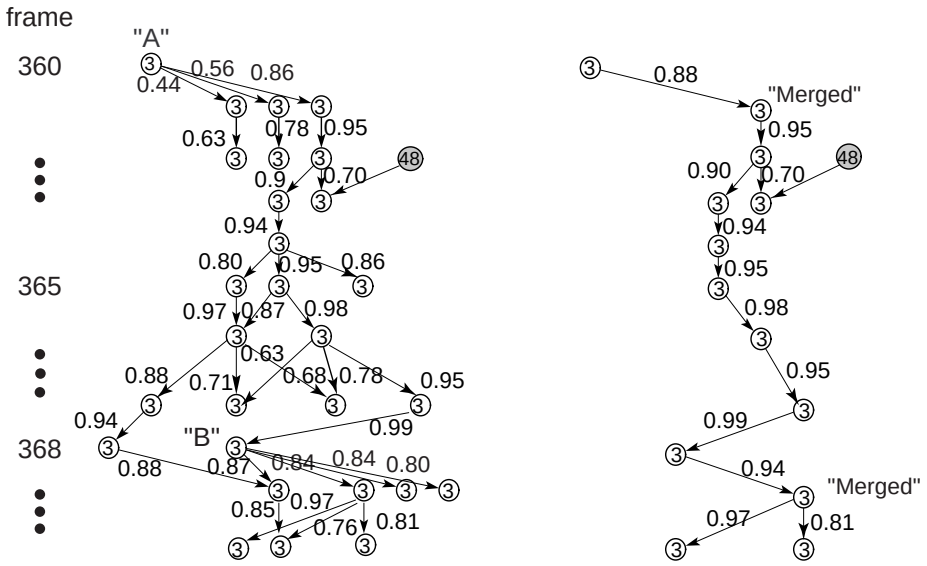
(g) frame 490



(h) frame 557

Fig. 5. *The "Stealing by Blocking" sequence. "A" approaches a reference object (a person standing in the middle with his belongings on the ground). "B" and "C" then approach and block the view of "A" and the reference person from their belongings. In the mean time, "D" comes and takes the belongings.*

the edges indicate the similarity of the region distribution based on K-S test ranging from 0 to 1. In figure 6(a), several hypotheses can be made about the possible tracks of "D" as indicated by numerous branching edges along the path from frame 360. However, none of these represents a correct track since most of the edges coming from a single parent node or going to a single child node are the results of the splitting and merging of moving regions as shown in figure 3. For example, at frame 360 and 368, "a" and "b" split into three



(a) Before (b) After

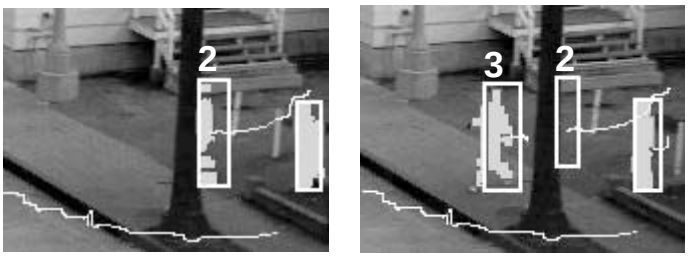
Fig. 6. *A graph representation of the possible tracks of object "D". (a) Without using the ground plane knowledge, several hypotheses can be made about the possible tracks of the object. (b) After filtering, regions are merged or disregarded, decreasing the ambiguity.*

sub-regions at the following frames. Figure 6 (b) shows the tracking graph after filtering regions based on the ground plane locations and K-S statistics. At frame 361 and 369, moving regions are merged and produce a higher similarity level to the moving region of the previous frame. At frame 365, 366 and 367, some moving regions are discarded as noise as they contradict with the ground plane assumption or when merged with the best candidate region, the similarity level decreases. After filtering, some tracking ambiguity may still remain in the graph. For example, the node at frame 362 can be associated with two regions in the following frame, one of which is also associated with the moving region of another object nearby. Such ambiguity can be removed based on other criteria such as event context.

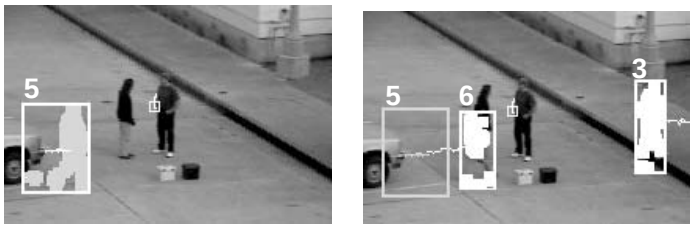
4.3 Resolving the Discontinuity of Object Trajectories

The discontinuity of tracks of moving objects in an outdoor scene often arises when moving objects are not detected for a few frames, such as from total occlusion, or when all regions do not satisfy the ground plane assumption. This is shown in figure 7. In this case, hypotheses about connections of fragments of tracks need to be made. We verify a hypothesis about the track correspondence based on the similarity of the intensity distribution of the moving blobs. We also evaluate events with various possible mergers and choose the one that gives the highest value. Figure 8 shows the final results of our track-

ing approach, where the correct shapes of moving objects are recovered and the confusion of possible multiple tracks is reduced.

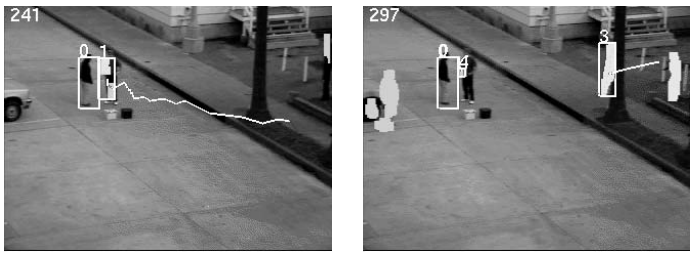


(a) Object 2 is totally occluded by the tree.

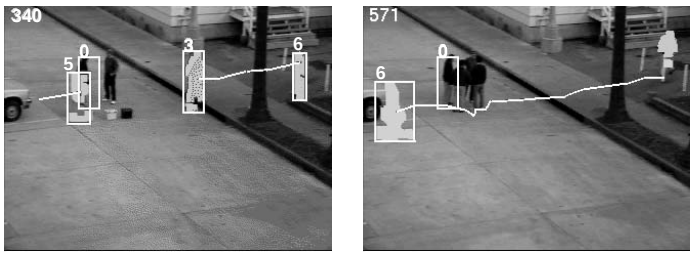


(b) Discontinuing tracks between Object 5 and 6.

Fig. 7. *(a) Object 3 is considered a different object from object 2 due to the total occlusion. (b) Due to the reflection on a car surface, the moving region of object 5 is larger than that of object 6, causing the discontinuity of the trajectory.*



(a) frame 241 (b) frame 297



(c) frame 340 (d) frame 571

Fig. 8. *Trajectories of the objects tracked using the ground plane knowledge and K-S statistics.*

5 Event Classification and Representation

Events are viewed as consisting of *single* or *multiple* **threads**. In a **single thread event**, relevant actions occur along a linear time scale (as is the case when only one actor is present). In **multiple thread events**, multiple actors may be present. A single thread event may be further categorized into a simple or complex event. We describe in detail how these events are defined and modeled in the following.

5.1 Simple, Single Thread Events (or Simple Events)

Simple events are defined as a short coherent unit of movement described by a concurrent *logical constraint* on a set of sub-events (other simple events) or directly from a set of mobile object properties. For example, in figure 2, simple event *"a person is approaching a reference person"* is described by the occurrence of three sub-events: *"getting closer to the reference person"*, *"heading toward it"*, and *"slowing down"*. These sub-events are defined from the properties of the object trajectory. The representation of a simple event can be viewed as an instantiation of a Bayesian network. Bayesian networks allow users to directly determine the causal relations between mobile object properties and simple events, reducing the complexity and redundancy of the representation. Simple events are verified at every frame from the probability distributions of the sub-events at that instance.

5.2 Complex, Single Thread Events (or Complex Events)

Complex events correspond to a linearly ordered time sequence of simple or other complex events. They may take place over long sequences (100 frames or more) and are usually over at least 10 frames. For example, *"converse"* is a linear occurrence of two consecutive simple events: *"a person approaches the reference person"*, then *"stops at the reference person"*. We propose to use a finite state automaton to represent a complex event as it allows a natural description [6]. Figure 9 shows an automaton representation of *"converse"*, where each automaton state represents a sub-event.

5.3 Multiple Thread Events

Multiple thread events correspond to two or more single thread events with some logical and time relations between them. Each composite thread may be

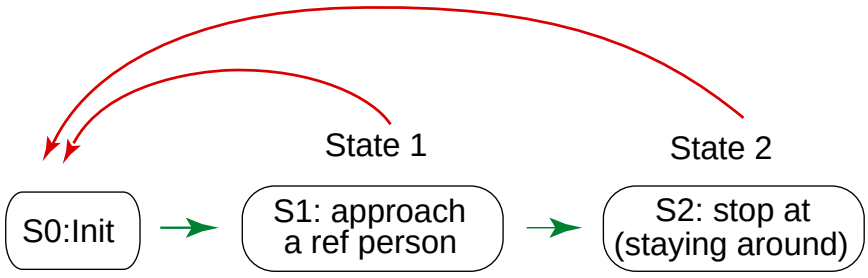


Fig. 9. *A finite-state automaton that represents the complex event "converse".*

performed by different actors. In such case, multiple thread events can be considered to model interactions among actors. In [16], interactions among actors are modeled by coupled HMMs. In coupled HMMs, interactions are defined at a fine temporal granularity and are represented by probabilistic transitions from a hidden state of one HMM to other hidden states of another HMM. In our case, we consider event threads to be a large scale event and propose to use the binary interval-to-interval relations, first defined by Allen [24], such as "before", "starts", "during" and "overlap", to describe temporal relations between sub-events of a multi-thread event. For example, *"event A must occur before event B"* and *"event A or event D occurs"*.

A multi-thread event is represented by an event graph similar to interval algebra networks [24]. One particular specification of an event that we call *"stealing"* may be composed of several threads of complex events performed by four actors, as described in Table 1. The temporal constraints among these events are that

1) *"converse"* occurs before *"approach1"* and *"approach2"*, 2) *"blocking"* starts right after or some time after *approach1* or *approach2*, and 3) *"taking_object"* occurs during *"blocking"*. Figure 10 shows a graphical representation of these events. The symbols "b", "d" and "s" stand for interval temporal constraints "before", "during" and "starts" respectively.

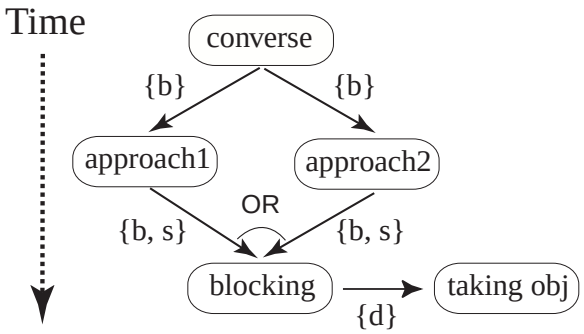


Fig. 10. *A graphical description of a multi-thread event.*

Description of sub-events of *"stealing"*	
converse	$actor_1$ approaches his friend
	and drops a suitcase on the ground.
approach1	$actor_2$ approaches and
	stops between $actor_1$ and the suitcases.
approach2	$actor_3$ approaches and stops
	between $actor_1$'s friend and the suitcases.
blocking	$actor_2$ or $actor_3$ are blocking
	the view of the suitcases.
taking_object	$actor_4$ approaches
	and takes the suitcases away.

Table 1
Description of the multiple thread event "stealing".

6 Single-Thread Event Recognition

To recognize which event model matches the video images the best, the entities in the representation must be computed. Computations at each level are inherently uncertain, hence a formal uncertainty reasoning mechanism is needed. We propose the use of Bayesian methods for making an inference about single-thread events, which are then combined to compute the probabilities of the multiple-thread events.

Suppose we have a set of competing event scenarios $S = S_1, ..., S_N$. Let O_{t_i} represents a set of mobile object properties computed at time frame t_i. Given a temporal sequence of observations $O_{1,t} = O_1 O_2 ... O_t$ of moving objects, we want to make a decision as to which event has occurred and when it starts and finishes. To make such a decision, we want to compute $\forall i, P(S_i|O_{1,t})$ and find the event with the maximal value. $P(S_i|O_{1,t})$ can be computed by inferring the distribution of sub-event values at the lower layers and propagating them towards the top layer. In particular, the event recognition process starts with the computation of the likelihood of single-thread events from the properties defined for the moving object of interest. The probabilities of these single-thread events are then combined to verify a multi-agent event. In this section, we discuss the Bayesian inference of $P(S_i|O_{1,t})$ where S_i is a single-thread event.

6.1 Object Class and Simple Event Recognition

The class of an object and its simple events are both represented by a Bayesian network and are inferred from the mobile object properties computed during a time frame. In Bayesian terms, the input entities are viewed as providing the evidence variables and the task is to compute the probability distribution of an output entity hypothesis. If the entities to be combined are statistically independent (given the scenario), a simple *naive* Bayes' classifier can be used to compute the distribution of the combined result. When the entities are not conditionally independent, Bayesian networks offer an efficient representation that can represent dependence by decomposing the network into conditionally independent components. To effectively use Bayesian networks, we need the knowledge about the network structure (i.e., which entities are directly related or linked to which entities) and the conditional probabilities associated with the links.

6.1.1 The Structure of Bayesian Networks

In our case, the structure of the network is derived from the knowledge about the domain. For example, logical constraints of sub-events that represent the recognition of a particular event indicate the direct causal link between them (i.e., the sub-events are the consequences of that event). By defining each event such that its sub-events are conditionally independent of each other given the event values, the hierarchy such as the one in figure 2 can be converted naturally into a Bayesian network which is composed of several layers of naive Bayesian classifiers (i.e., no hidden nodes). Each layer in the hierarchy can be viewed as having several naive Bayesian classifiers (one classifier per each scenario). Belief propagation is performed in one direction from the bottom layer to the top layer.

In figure 2, at the top layer, the parent event *"approach a reference person"*, is linked to three child events: *"getting closer to the reference person"* (e_1), *"heading toward the reference person"*(e_2), and *"slowing down"* (e_3). These child events form another layer of three naive Bayesian classifiers (e.g., e_1 becomes a parent of *"distance evolution"* (the difference of the distance to the reference object at frame t and $t-1$). The probability distribution over the parent event values in Bayesian classifiers ($P(H|e_1, e_2, e_3)$) is inferred from the distribution of child event values and the conditional probabilities of child events given the values of the parent event as shown in figure 11. Given that e_1, e_2 and e_3 are conditionally independent given H, the belief is propagated from the sub-events e_1, e_2 and e_3 to infer the probability distribution of H

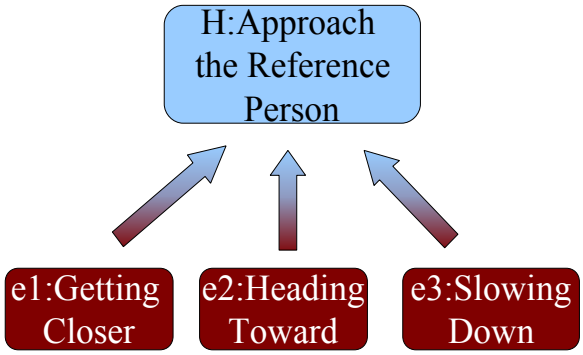


Fig. 11. *A detailed illustration of a naive Bayesian classifier that is used to infer "approach the reference person" in figure 2.*

(i.e., $P(H|e_1, e_2, e_3)$) by applying Bayes' rule as follows:

$$P(H|e_1e_2e_3) = \alpha P(e_1|H)P(e_2|H)P(e_3|H)P(H)$$
$$P(\neg H|e_1e_2e_3) = \alpha P(e_1|\neg H)P(e_2|\neg H)P(e_3|\neg H)P(\neg H),$$

where α is a normalizing constant such that $P(H|e_1e_2e_3) + P(\neg H|e_1e_2e_3) = 1$.

6.1.2 Parameter Learning

Parameters to be learned in a Bayesian network are the conditional probabilities of child events given the values of the parent event. Traditionally, these parameters are learned using the Expectation-Maximization (EM) algorithm when hidden nodes are present. In our case where all nodes are transparent (e.g., we can observe whether the object is moving towards another object or whether it is slowing down), these conditional probabilities (i.e., $P(e_i|H)$ and $P(e_i|\neg H)$)can be learned from image sequences directly such as by making a histogram of observed values of the evidence variables, e_1, e_2, e_3, given the value of a given hypothesis, H. In the case where a single Gaussian distribution can be assumed, the Gaussian parameters (μ and σ) can be computed easily.

6.2 Complex Event Recognition

Complex events, defined as a temporal sequence of sub-events, are represented by a finite state automaton as shown in figure 9 for the event *"converse"*. The dynamics of the complex event is modeled by the probabilistic transitions among the event states, as shown by the arrows in the figure, where state i either advances to state $i+1$ or remains in the same state. Both sub-events of *"converse"* are simple events. The recognition process consists of decoding the pattern of sub-events (which, in our case, are inferred from

observed motion and shape features) and segmenting them into the corresponding states. As the occurrence of sub-events are uncertain, the decision on the transition between states also becomes uncertain. We define the probability of a multi-state complex event (MS) as $P(MS^*|O)$: *the probability that the complete automaton state sequence of MS occurs with the most likely state transition timing given the sequence of observations $O = O_{(1,t)}$*. This can be computed as follows:

$$P(MS^*|O) = \max_{\forall(t_1,t_2,...,t_N)} P(S_{1_{(t_1,t_2-1)}} S_{2_{(t_2,t_3-1)}} ... S_{N_{(t_N,t)}}|O), \tag{3}$$

where t_i refers to the time that the transition to state i from state $i-1$ occurs and $S_{i_{(t_i,t_{i+1}-1)}}$ means that scenario event i occurs during t_i and t_{i+1} - 1.

For compactness, we drop the timing notation symbols and let S_i be $S_{i_{(t_i,t_{i+1}-1)}}$ and S_1^N be $S_{1_{(t_1,t_2-1)}}\ S_{2_{(t_2,t_3-1)}}\ ...S_{N_{(t_N,t)}}$. Similarly, let O_i be $O_{(t_i,t_{i+1}-1)}$ and O_i^j be $O_iO_{i+1}...O_j$. The computation of $P(S_1^N|O)$ can be decomposed into the recognition results of sub-events by the following steps. First, from the conditional probability axiom, we have that,

$$P(S_1^N|O) = P(S_N, S_1^{N-1}|O_N, O_1^{N-1}) \tag{4}$$

$$= P(S_1^{N-1}|S_N, O_1^{N-1}, O_N)P(S_N|O_1^{N-1}, O_N). \tag{5}$$

By making an assumption that given O_N, S_N is independent of O_1^{N-1} and vice versa, Eq. 5 can be written as:

$$P(S_1^N|O) = P(S_1^{N-1}|S_N, O_1^{N-1})P(S_N|O_N). \tag{6}$$

From the conditional probability axiom, Eq. 6 becomes,

$$P(S_1^N|O) = \frac{P(S_N|S_1^{N-1}, O_1^{N-1})P(S_1^{N-1}|O_1^{N-1})P(S_N|O_N)}{P(S_N|O_1^{N-1})}. \tag{7}$$

By making an assumption that S_N and O_1^{N-1} are independent of each other, we can write Eq. 7 as:

$$P(S_1^N|O) = P(S_N|O_N)[\frac{P(S_N|S_1^{N-1})}{P(S_N)}]P(S_1^{N-1}|O_1^{N-1}). \tag{8}$$

By making a Markov assumption that the probability of being in state i at time t depends only on the state at time $t-1$ (i.e., $P(S_N|S_1^{N-1})$=$P(S_N|S_{N-1_{t_{N-1}}})$), the terms in the bracket in Eq. 8 can be written as $a_{N,N-1}$ (the probability of the path being taken from S_{N-1} to S_N normalized by all the possible paths from S_{N-1}). Eq. 8 can now be written as:

$$P(S_1^N|O) = P(S_N|O_N)a_{N,N-1}P(S_1^{N-1}|O_1^{N-1}). \tag{9}$$

We can recursively expand the term $P(S_1^{N-1}|O_1^{N-1})$ using equations 4 to 9 and finally get that,

$$P(S_1^N|O) = \prod_{1 \le i \le N} a_{i,i-1} P(S_{i_{(t_i, t_{i+1}-1)}}|O_{(t_i, t_{i+1}-1)}) \tag{10}$$

where $t_{N+1} = t$ and $a_{i,i-1}$ is assumed to be constant for all scenarios. By substitute Eq. 10 into Eq. 3, we have:

$$P(MS^*|O) = \max_{\forall(t_1,...,t_N)} \prod_{1 \le i \le N} a_{i,i-1} P(S_{i_{(t_i, t_{i+1}-1)}}|O_{(t_i, t_{i+1}-1)}). \tag{11}$$

We describe next the algorithm to find the values of $t_1, t_2, ..., t_N$ that maximize $P(MS^*|O)$.

Complex Event Recognition Algorithm

The direct computation of $P(MS^*|O)$ at time T using Eq. 11 involves an operation of $O(T^N)$ complexity since there are $O(T^N)$ combination of values of $t_1, t_2, ..., t_N$. However a more efficient recursive algorithm based on dynamic programming can be applied. This algorithm is an adaptation of the Viterbi's algorithm used for HMMs to our finite-state automaton. Let $R_i(t)$ be the likelihood that MS occupies state i at time t with the most likely transition timing between states given the observation sequence $O_{(t_1,t)}$. That is,

$$\begin{aligned} R_i(t) &= \max_{\forall(t_1,t_2,...,t_i)} P(S_{1_{(t_1,t_2-1)}} S_{2_{(t_2,t_3-1)}} ... S_{i_{(t_i,t)}}|O_{(t_1,t)}) \qquad \dots(a) \\ &= \max_{\forall(t_1,t_2,...,t_i)} \prod_{1 \le j \le i} a_{j,j-1} P(S_{j_{(t_j,t_{j+1}-1)}}|O_{(t_j,t_{j+1}-1)}) \qquad \dots(b) \end{aligned} \tag{12}$$

The solution for Eq. 3 is, therefore, equivalent to $R_N(t)$. $R_i(t)$ can be derived from previously recognized state $i-1$ as follows. For short, let A_j be the term on the right hand side of the product.

$$\begin{aligned} R_i(t) &= \max_{\forall(t_1,t_2,...,t_i)} \prod_{1 \le j \le i} A_j \qquad \dots(a) \\ &= \max_{t_{i-1} \le t_i \le t} [A_i \max_{\forall(t_1,t_2,...,t_{i-1})} \prod_{1 \le j \le i-1} A_j] \qquad \dots(b) \\ &= \max_{t_{i-1} \le t_i \le t} A_i R_{i-1}(t_i - 1) \qquad \dots(c) \end{aligned} \tag{13}$$

By substitute A_i back in Eq. 13 (c), we have that,

$$R_i(t) = \max_{t_{i-1} \le t_i \le t} a_{i,i-1} P(S_{i_{(t_i,t)}}|O(t_i,t)) R_{i-1}(t_i - 1) \tag{14}$$

$$t_{i_{best}} = \underset{t_{i-1} \le t_i \le t}{argmax} a_{i,i-1} P(S_{i_{(t_i,t)}}|O_{(t_i,t)}) R_{i-1}(t_i - 1) \tag{15}$$

At time t, starting from state 1 where $R_0(t)$ is always 1, Eq. 14 are recursively processed until the final state N is reached, where $R_N(t)$ represents the probability of the sequence of states occurs with optimal transition timing $t_{1_{best}}, t_{2_{best}}, ..., t_{N_{best}}$ (i.e., $P(MS^*|O)$).

Finding $t_{i_{best}}$ that Maximizes $R_i(t)$

To compute $R_i(t)$ in Eq. 14, two issues need to be addressed: how $P(S_{i_{(t_i,t)}}|O_{(t_i,t)})$ is computed and how to search for t_i that maximizes $R_i(t)$.

One way to derive $P(S_{i_{(t_i,t)}}|O_{(t_i,t)})$ is to write it out in terms of state transitional probabilities $(a_{i,i})$ and the probability of the state S_{it_j} given O_{t_j} for all $t_i \leq t_j \leq t$:

$$P(S_{i_{(t_i,t)}}|O_{(t_i,t)}) = (\prod_{t_i \leq t_j \leq t} P(S_{it_j}|O_{t_j}))(a_{i,i})^{t-t_j)}(1-a_{i,i}) \tag{16}$$

This is similar to the probability computation method in the traditional HMM decoding except for the alternate role of S_{it_j} and O_{t_j}. One weakness of Eq. 16 is that the event duration is inappropriately modeled by an exponential function $(a_{i,i})^{t-t_j)}(1-a_{i,i})$. Unlike the case of speech recognition where the duration of a phoneme (e.g., the sound */th* in *"the"*) is restrictively short, the duration of an activity is highly variable and can last for several minutes. Therefore, the exponential model of event duration may not give accurate results as

A more accurate way of computing $P(S_{i_{(t_i,t)}}|O_{(t_i,t)})$ is to model the event duration explicitly (also known as a semi-HMM) [25]. Eq. 14 can be thought as one implementation of the semi-HMM, where the best sequence of events are searched in a longer time scale. Decoding a semi-HMM is NT times more complex than a conventional HMM (where N is the number of event states and T is the total time frames), since we need to re-evaluate $P(S_{i_{(t_i,t)}}|O_{(t_i,t)})$ for all possible t_i at each time frame t. In our experiment, we approximate the decoding in the semi-HMM as follows.

We assume that all event durations are equally likely and, therefore, are canceled out when alternative event paths are compared during the same period of time. We approximate $P(S_{i_{(t_i,t)}}|O_{(t_i,t)})$ by computing a temporal mean (expected value) of the distribution of $P(S_{it_m}|O_{t_m})$ collected from frame $t_m = t_i$ to $t_m = t$ to average out the noise.

To find the optimal t_i (i.e., $t_{i_{best}}$) that maximizes $R_i(t)$, we would have to investigate all possible values of each t_i. However, there is a criteria to disregard many of these values as a candidate for $t_{i_{best}}$. Let t_i' be a possible value of t_i. One indicator for t_i' to be disregarded as a potential candidate for t_i is the fact that the accumulative probability that scenario i does not occur during t_i' and t is greater than the accumulative probability that scenario i does occur, which indicates that scenario i does not occur during t_i' and t. In general, only a certain numbers, say k, of such t_i candidates $(t_i^k = t_{i_1}, ..., t_{i_k})$ that compute the highest $R_i(t)$ can be maintained.

The algorithm for maintaining the set t_i^k is summarized in the following. Let

- $S_i^+(t'_i, t)$ be the accumulative probability density of the scenario (above threshold δ which is inversely proportionate to the degree of noise in the image sequences) from time t'_i to t,
- $S_i^-(t'_i, t)$ be the accumulative probability density when the scenario is not recognized (i.e., under the threshold δ) from time t'_i to t, and
- $E[S_i]_{(t'_i,t)}$ be the expected recognition value of event S_i during t'_i and t. It represents the temporal mean of the distribution $P(S_{i_{(t'_i,t)}}|O_{(t'_i,t)})$ used in Eq. 14 and 15.

$S_i^+(t'_i, t)$ and $S_i^-(t'_i, t)$ are computed as follows:

$$S_i^+(t'_i, t) = \sum_{t'_i \le j \le t, P(S_{i_j}) > \delta} P(S_{i_j}),$$
$$S_i^-(t'_i, t) = \sum_{t'_i \le j \le t, P(S_{i_j}) \le \delta} (1 - P(S_{i_j}))$$

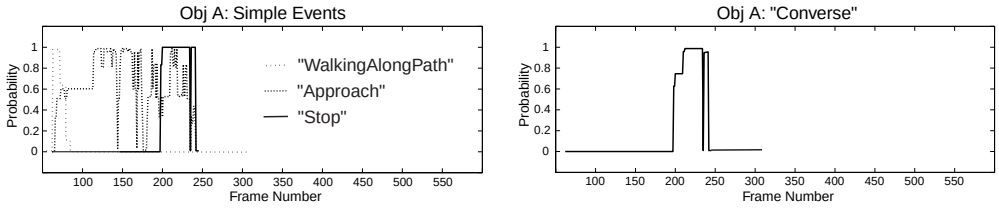
$E[S_i]_{(t'_i,t)}$ is computed from $S_i^+(t'_i, t)$ and $S_i^-(t'_i, t)$ as follows:

If $(S_i^+(t'_i, t) > S_i^-(t'_i, t))$,
then $E[S_i]_{(t'_i,t)} = \frac{S_i^+(t)}{t - t'_i}$,
else, $E[S_i]_{(t'_i,t)} = 0$ and t'_i is discarded from the potential candidate values for $t_{i_{best}}$.

In term of complexity, if only a certain number k of t_i are maintained, we need to update k number of temporal means to compute $R_i(t)$. Since this process is repeated for all N states, multi-state complex event recognition algorithm requires $O(NT)$ operations, which is as efficient as the Viterbi algorithm used in the HMM approach. However, the construction of the model, and the initialization and learning of the parameters can be much easier in our case since the nodes of the network are transparent.

6.3 Analysis Results of Single-Thread Events

We constructed eighteen Bayesian networks similar to the one shown in figure 2 to represent simple events. These events consist of actions related to both the shape of a moving object (e.g., *"stand"*, *"crouch"*) and object's trajectory (e.g., *"approach"*, *"stop at"*, *"slow down"*). Some simple events are defined with regard to a geometrical zone (e.g., a road) such as *"moving along the path"*. Eight complex events are modeled by constructing an automaton of simple events or other complex events. Parameters of each network are assumed to be Gaussian distributions and are learned from a training data set composed of up to 5 pre-segmented sequences (approximately 600 frames). Training sequences are staged performance by different pedestrians and con-



a) Simple events b) Complex event: "Converse"

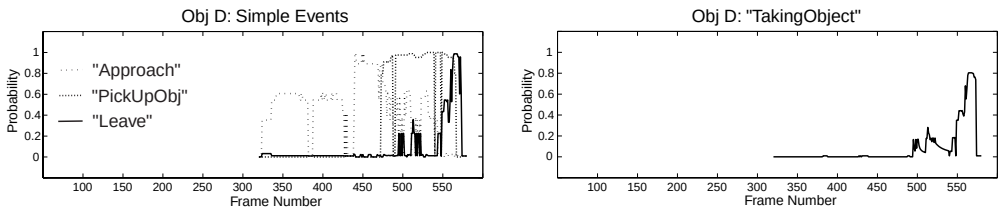
Fig. 12. *Analysis of single-thread events for object "A".*

tain about half of positive and half of negative examples. They are taken in different days, weather conditions and locations. A priori probabilities of all events are assumed to be equal. These event models constitute the scenario library in figure 1. We tested our single-thread event recognition algorithm on video streams collected in various domains: parking lots, street corners and airborne platforms. In the following, the algorithm is first demonstrated using the analysis results of the actions of two objects in "stealing" sequence (figure 5). Then, we show that our system can discriminate and correctly recognize competing scenarios using videos obtained from a different scene than the "stealing" sequence.

6.3.1 Recognition of "Converse" *and* "Taking Object"

- Complex event *"converse"* is a sequential occurrence of two simple events: *"approach the reference person"* and *"stop at the reference person"*. This is, indeed, the action performed by object "A" in the "stealing" sequence. We process the "stealing" sequence by analyzing all simple and complex events modeled in the library. Figure 12(a) shows the evolution of the probabilities of four most significant simple events. The probabilities of other simple events related to the trajectory are lower than 0.2 at all time frames. These results correspond with the human observation of the action of object "A". Object "A" first appears on the pedestrian path (defined by the user using a polygon drawn on the ground plane) on the right and moves along it for about thirty frames. It then proceeds toward the reference object and stops. It can be noticed that the probabilities of *"approach the reference person"* are considerably noisy compared to *"stop at the reference person"*. This is because the recognition of *"approach the reference person* depends on instantaneous direction and speed of the objects, of which the measurements are not stable. In contrary, *"stop at"* event varies smoothly on the distance measurements. That is, once the object appears near the reference person, *"stop at"* will remains high as long as the object does not appear to jump a distance.

  Figure 12(b) shows the evolution of complex event *"converse"* which is recognized by detecting the sequential occurrence of *"approach the reference object"* and *"stop at"*. It can be noticed that even though *"converse"*



a) Sub-events of "Taking Object" b) Complex event: "Taking Object"

Fig. 13. *Analysis of single-thread events for object "D".*

depends on the recognition of the noisy *"approach the reference person"*, its probability is more stable. This is because the temporal mean algorithm averages out the noise.

- Complex event *"taking Object"* is a sequential occurrence of *"approach the reference person"*, *"pick up object"* and *"leave"*. It is modeled in a similar way to *"converse"*. However, while all sub-events in *"converse"* are simple events, *"pick up object"* is defined as a complex event *"bend down"* conditioned on the fact that the suitcase has disappeared. *"Bend down"* is modeled by an automaton composed of five shape-related event states: *"stand"*, *"height decreasing"*, *"crouch"*, *"height increasing"* and *"stand"*.

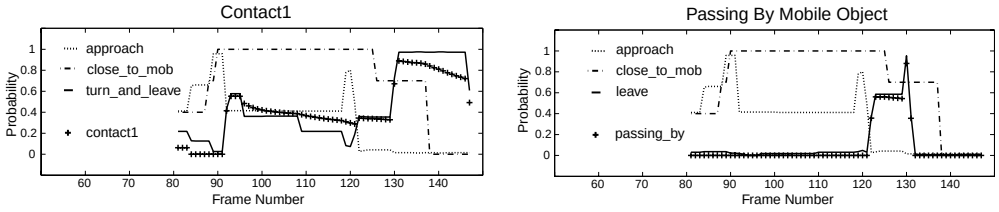
  Figure 13(a) shows the probabilities of the sub-events of *"taking object"* for the object "D", which match with the human observation. In figure 13(b), *"taking object"* is recognized when its sub-events have sequentially occurred. We note that *"pick up object"* is, in fact, an articulated motion that, in an ideal case, should be recognized by tracking movements of human limbs. However, in many applications (in particular, the surveillance of an outdoor scene), moving objects may appear very small and tracking body configuration is not possible. We have shown that, in such cases, the approximate movement of shape (i.e., the change of the bounding box) may be combined with an event context (i.e., the fact that the suitcase is missing) to recognize a more sophisticated activity.

6.3.2 Recognizing Competing Events in a Parking Lot

We demonstrate the efficacy of our single-thread event recognition algorithm in a different domain by analyzing ten video streams (digitized at 15 Hz) that capture activities in two different parking lot areas. The parameters of the networks are learned from sequences taken at the first parking lot. We tested the network on the sequences taken from the both parking lots; the second lot has a different street structure (the streets are of high curvature). We show results of two test sequences called sequence **A** and **B** shown in figure 14 (a) and 15 (a) taken in the second parking lot.

frame A.63 frame A.111 frame A.157

(a) Detection and tracking of moving regions for "contact1".



I) Output of the "contact1" II) Output of the "passing_by"

(b) Recognition results of two competing activities.

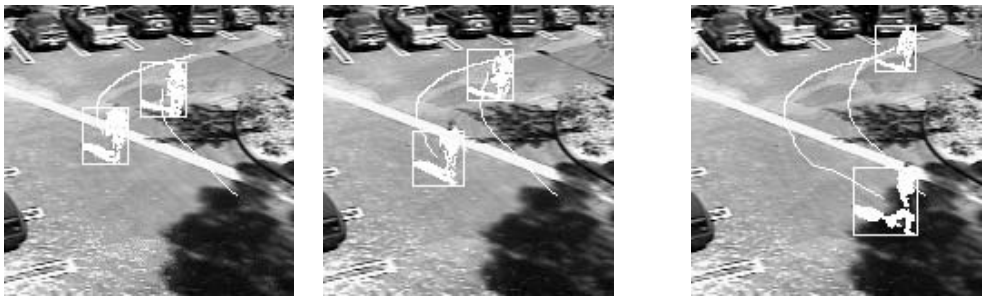
Fig. 14. *(a) Input sequence* **A** *shows a complex event "contact1". Object 1 (at the top) approaches object 2 (at the bottom), makes contact (both objects have merged as they meet), turns around and leaves. (b) "Contact1" is recognized with* $P(MS^*|O)$ *= 0.7. On the other hand, "passing by" is recognized with lower probability (almost 0 at the end) since sub-event "leaving without turning around" is not established.*

Description of Competing Events

We model two competing events to be detected using our hierarchical representation. First activity, *"contact1"* is a complex event defined as an automaton of three sub-events: *"two men approach each other, make contact, turn around and leave"*. These sub-events are then described by the network similar to the one in figure 11. The second activity, *"passing by"* is also a complex event defined as *"A man approaches another man, walks past the other, and then leaves"*. *"passing by"* is described by a network similar to that for *"contact1"*, but with different sub-events.

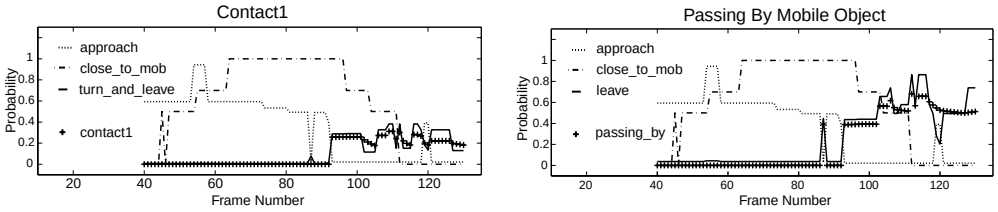
Recognition Results

The analysis results for sequence **A** are shown in figure 14 (b). The dotted, dotted-dashed, and solid lines show the likelihood of the sub-events which are derived using Bayesian networks described in section 6.1. The line marked with "+" shows $P(MS^*|O)$ computed based on our complex event recognition algorithm. The results show that *"contact1"* is recognized with higher



frame B.97 frame B.108 frame B.138

(a) Detection and tracking of moving regions for scenario "passing by"



I) Output of the"contact1" II) Output of the "passing by"

(b) Recognition results of two competing activities.

Fig. 15. *(a) Input sequence* **B** *shows a complex event "passing by":Object 1 walks past object 2. (b) "passing by" is recognized with higher confidence* ($P(MS^*|O) = 0.6$) *than "contact1"*($P(MS^*|O) = 0.2$).

confidence ($P(MS^*|O) = 0.7$) compared to *"passing by"* ($P(MS^*|O) = 0$).

Figure 15 (b) shows the analysis results for sequence **B**. This sequence depicts the event *"Passing By"* which is correctly recognized with probability of 0.6, while the event *"Contact1"* is poorly recognized at the lower value of 0.2.

7 Multi-Thread Event Recognition

Multi-thread events are recognized by evaluating the likelihood values of the temporal and logical relations among event threads. Constraint satisfaction and propagation techniques have been used in the past [24] based on the assumption that events and their durations are deterministic. We present in this section an algorithm to verify and propagate temporal constraints when events are uncertain.

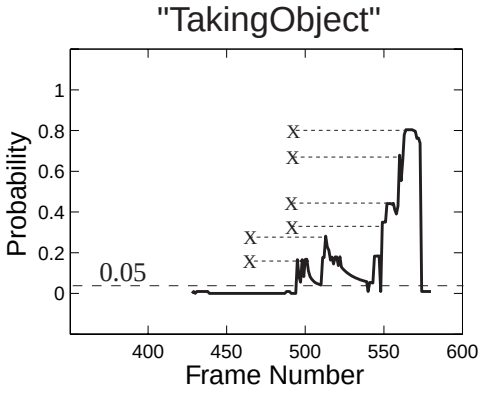


Fig. 16. *The likelihood of the complex event inferred by a probabilistic finite state automaton. The event, at different times, may have different likely start times as illustrated by "x", depending on the most likely transition timings between states considered up to the current frame.*

7.1 Evaluation of Temporal Relations

Suppose that we have two complex events: *event A* and *event B*, each with a likelihood distribution similar to the one shown in figure 16. Computing the likelihood of these events satisfying a temporal constraint requires an exploration of the combination of all possible event intervals of both events that may occur during time frame 1 and the current frame. For example, to compute *"A before B"*, we need to find a time t' frame such that *event A* started and ended before t' and *event B* started after t' and may still occur. The event intervals of A and B that give the maximal combined likelihood define the occurrence of the multi-thread event.

In our case, the interval of an event can be inferred from the finite state automation by keeping track of the transition timings between states and by making an assumption that the end point of an event can be determined when the likelihood of the event becomes very low. For example, in figure 16, assuming that an event ends when the likelihood becomes lower than 0.05, we have three possible end times of this event at frame 511, 542, and 575. From our experiment, the number of possible end points of a complex event is relatively small compared to the length of video data. Suppose that event threads are independent of each other. After the constraint on the interval of two events are verified, we can compute the likelihood of the multi-thread event as the product of their probabilities. This computation requires a search operation of $O(k^{R+1})$ complexity if there is an average of k possible starting points for an event model and R is the number of temporal relations in a series of nodes in the event graph.

Other temporal relations can also be computed similarly. For example, to compute *"A during B"*, we first find all possible event intervals of A and B.

We then search for a combination of these events that produces the maximal likelihood subject to a constraint that the start and end times of A must be during the interval of B.

Event threads that are defined using a logical constraint such as "and" and "or" can be combined much easier than a temporal constraint, as we do not need to verify a temporal relation. For *"A and B"*, we compute the product of the event likelihood. For *"A or B"*, we choose the event with a higher likelihood.

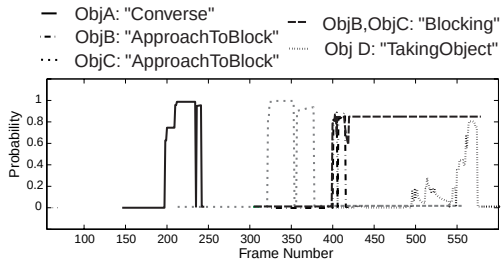
7.2 Inferring a Multi-Thread Event

A multi-thread event that is described by more than two event threads can be inferred by propagating the temporal constraints and the likelihood degrees of sub-events along the event graph. For example, suppose that we have two event constraints: *"A before B"* and *"B before C"*. To evaluate *"B before C"*, we need to consider an extra constraint that "event B" occurs after "event A".

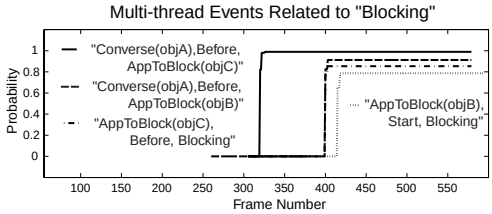
7.3 Multi-Thread Event Analysis Results

We have constructed four multi-thread event models, with the most complicated event model being the *"stealing"* scenario. In this section, We illustrate our multi-agent event recognition algorithm using the sequence *"stealing"*(figure 5). Figure 17 (a) shows the recognition of the most significant complex events (modeled by a probabilistic automaton) evaluated on object A, B, C and D. The plots in the figure correspond to the detection of event nodes in figure 10. The results of simple events are omitted for clarity. Object A is recognized to be performing *"converse"* at approximately frame 200 after it approaches and stops at the reference person. Objects B and C are recognized to be performing *"approach to block"* at frame 400 and 320, respectively. These events are an instantiation of *"approach1"* and *"approach2"* in figure 10. The *"blocking"* event is detected as two objects approach each other and stop in the view blocking zone at approximately frame 400. Finally, Object D is recognized to be performing *"taking objects"*. Figure 17 (b) shows the recognition of the temporal relations defined in *"stealing"* together with the assignment of the actors. In figure 17 (c), the solid and dotted-dashed lines show the probabilities of event *"stealing"*, where the sub-event *"approach to block"* is performed by object B and object C, respectively.

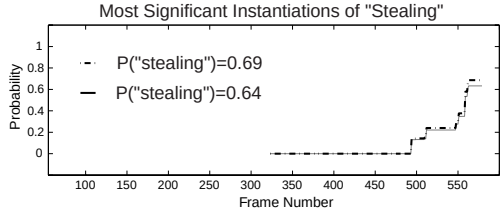
We note that as the complexity of the event model increases, the probability of the event tends to decrease. This is due to the introduction of more param-



(a) Recognition of single-thread events.



(b) Multi-thread events evaluated related to "blocking" and the corresponding assignment of actors.



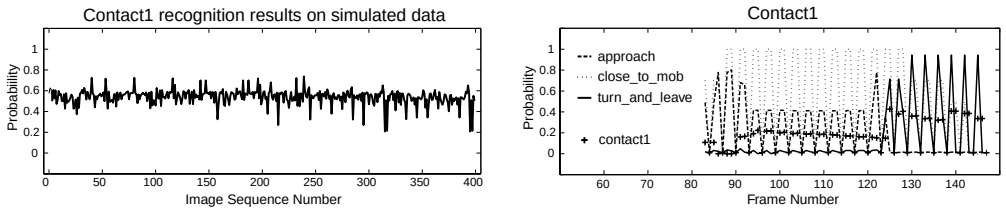
(c) "Recognition results of "stealing", each with different actor and temporal event combinations.

Fig. 17. *Event analysis results of the sequence "stealing".*

eters, and hence the ambiguity arises. To accurately select a rare but complex scenario, an alternative model selection criteria that takes into consideration of the model complexity is necessary.

8 Performance Characterization

We have tested our system successfully on real video streams containing a variety of events in addition to the ones shown in section 6.3 and 7.3. These events are mainly for surveillance purposes such as *"a person drops off a package"*, *"a person (a car) follows another person (another car)"* and *"people cooperatively exchange objects"*. Evaluating the performance of an event detection system, however, requires a much more extensive collection of data. In an ideal case, the data set should contain all possible variations of any particular event captured at various time, under different environments and possibly in a crowd.



(a) Results on perturbed sequences. (b) Example of perturbed sequence.

Fig. 18. *(a) "Contact1" recognition values of 400 perturbed image sequences. (b) The recognition result on an example of perturbed sequence with the loss of tracking on every three frames. The recognition of sub-events becomes zero every time the mobile object is lost. "Contact1", however, is still correctly recognized.*

Obtaining such data is possible if the action to be recognized is confined in a highly structured scene and has little variation in the execution style as in the case of sport events. However, significant scenarios such as *"stealing"* occur rarely and sometimes have execution styles that are difficult to anticipate. Abstract activities also allow for more variation in temporal constraints. For example, the path and the duration that a thief takes to steal a suitcase can vary depending on the location of the suitcase. Simulated data of "variations" may help in such cases.

There are several variations that may affect the performance of our recognition algorithm:

- Bayesian networks can be affected by noise and error in motion detection and tracking;
- Probabilistic event automaton can be sensitive to the variable durations of sub-events in addition to tracking noise;
- The performance of event graphs can be affected further by the variable timing of event relations due to different execution styles.

In the following, we characterize the performance of our recognition methods using spatio-temporal perturbation of tracking data computed from real video streams that simulates these noise and variations.

8.1 Loss of Tracking

Suppose we are to detect event *"contact1"* described in section 6.3. We generate four hundred perturbed sequences from sequence **A** shown in figure 14 by randomly inserting and deleting the tracked mobile objects as follows. A random number uniformly distributed between 1 and 20 is generated and used as a timing point where the sequence is perturbed. For each perturbed sequence, three such random numbers are selected and used as timing point to duplicate

or delete a frame or to remove the information on the tracked blobs. For example, if the random number is 3, a frame may be inserted once in every three frame and so on. The smaller the random number is the more the sequence becomes perturbed. As a result, mobile objects may appear to be stationary or lost from time to time. The probabilities of *"contact1"* at the final frame of each processing are shown in figure 18 (a). It shows that on average *"contact1"* is recognized with a probability of 0.55 (with the highest value of 0.74 and lowest value of 0.21). Figure 18 (b) shows the result of recognition on a sequence where the tracked moving blobs were removed once in every three frames. From the stability of the line marked with "+", our method still infers the correct scenario.

8.2 Levels of Noise

Suppose we are to recognize and discriminate between *"contact1"*(figure 14) and *"passing by"*(figure 15). We generate noisy sequences from sequence **A** and **B** by corrupting the tracking data with Gaussian noise. First, we compute the mean ($\mu = 13.01 cm/frame$) and variance ($\sigma = 6.68 cm/frame$) of the speed of walking people detected in both sequences. Tracking data of the original sequence is then corrupted with a Gaussian noise $G(\mu = 0, \omega \cdot \sigma)$ to simulate a noisy sequence, where ω is the level of noise. The larger ω is, the noisier the tracking data becomes corrupted.

In the following experiment, we test our algorithm on $\omega = 1, 3$ and 7. Forty sequences of noisy *"contact1"* and *"passing by"* are evaluated for each level of noise. If the probability of the event model corresponding to the sequence is lower than the threshold value (τ), we say that it produces a negative result. If the probability of the competing event model is higher than τ, then we say that it produces a false alarm. In an ideal case, the system would produce zero negative results and zero false alarm. We characterize results in terms of missing rate and false alarm rate. Missing rate is the ratio of negative results to all the positive sequences and false alarm rate is the ratio of the false alarms to all the positive sequences. A trade-off between the missing rate and the false alarm rate can be exercised by varying the threshold value. An optimal threshold can be selected to give the desired trade-off for a given application based on some criteria. To help make such a decision, it is common to plot a trade-off curve by varying threshold values. This curve is commonly called a *Receiver Operating Characterizing* (ROC) curve.

Figure 19 shows ROC curves for each noise level for the detection of *"contact1"* and *"passing by"*. It can be seen that when the variance of the random Gaussian noise of tracking data is below $20.04 cm/frame$ (three times as much as the walking speed variance of humans), it is still possible to maintain both the

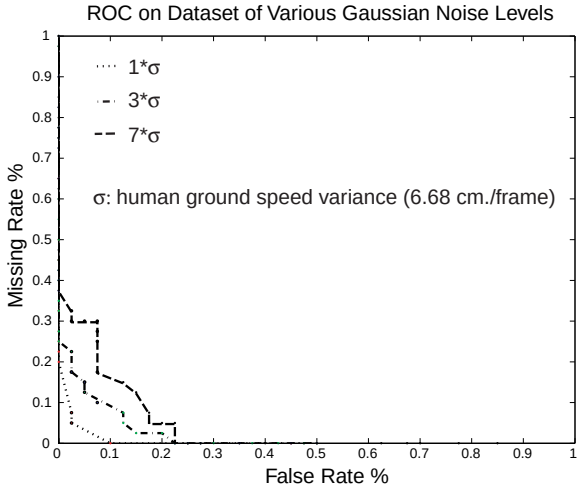


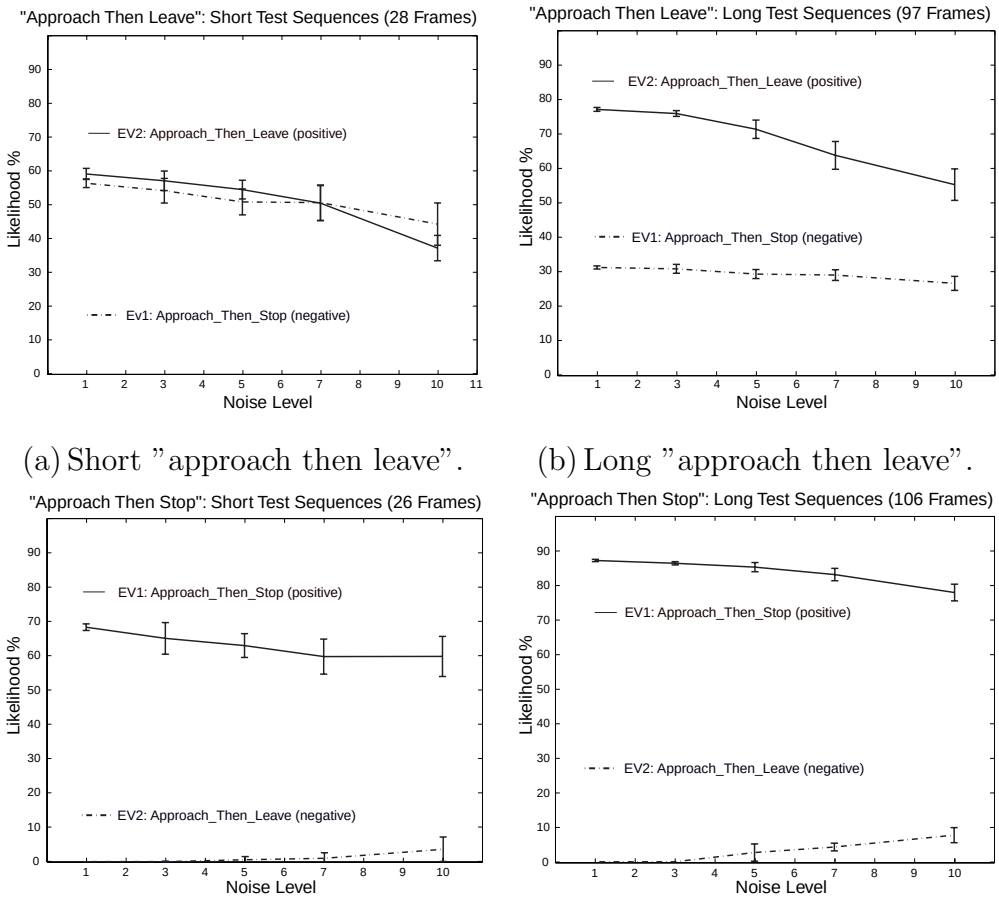
Fig. 19. *ROC curves for a data set with various noise levels.*

missing rate and the false alarm rate under 10%. This is because the Gaussian noise only locally corrupts the ground location of an object frame by frame. The overall shape of the trajectory is, therefore, still preserved. Such local noise is averaged out during the temporal mean computation. The error occurs when the noise repeatedly causes a large displacement while the person still stops at the reference person, causing the recognition of *"stops at"* to be weak for a long period of time.

8.3 Variable Event Durations

In this section, we examine how the duration of sub-events affect the recognition of complex events. We model two competing complex events. The first event (EV_1), *"approach a reference person and then stop"*, is composed of two sub-events: *"approach"* and *"stop"*. The second event (EV_2), *"approach a reference person and then leave"*, is also composed of two sub-events: *"approach"* and *"leave"*. We simulated, for each complex event, a test data set of twenty sequences for five different noise levels ($\omega = 1, 3, 5, 7$ and 10) using the method described in section 8.2. Half of the test data set (i.e. ten sequences) has a shorter duration of the second sub-event.

For each noise level, we analyze the test data set and compute the mean and standard deviation of the probabilities of both EV_1 and EV_2. Figures 20(a) and (b) show the results of the *"approach then leave"* test sequences with different lengths of the second sub-event (*"leave the reference person"*); 28 and 97 frames, respectively. The average probabilities of the positive event model EV_2 are shown in solid lines with the standard deviation displayed with $\vdash\dashv$. The dotted lines are the probabilities of the negative event model EV_1. Similar to figures 20(a) and (b), figures 20(c) and (d) show the analysis



(a) Short "approach then leave". (b) Long "approach then leave".

(c) Short "approach then stop". (d) Long "approach then stop".

Fig. 20. *Discriminating "approach then leave" and "approach then stop" in test sequences of various lengths.*

results of the *"approach then stop"* test sequences with different lengths of the second sub-events; 97 and 106 frames, respectively. In (c) and (d), EV_1 (solid lines) become the positive event model.

It can be seen that the system can discriminate between competing event models much better when the sub-events are observed for a longer period of time. That is, a larger gap between the solid and dotted lines is observed at every noise level in the figures on the right than those on the left. Especially in the "approach then leave" test sequences, the system discriminates poorly between EV_1 and EV_2 when the event *"leave"* is not observed long enough. This, in fact, corresponds with human observation because as observers get confused when event duration is one or two seconds long.

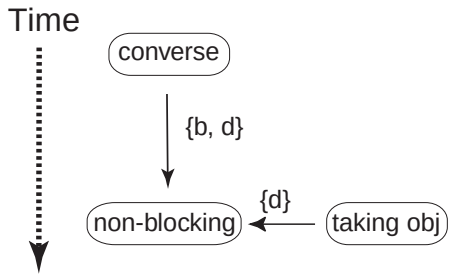


Fig. 21. *A graphical description of event "cooperative object transfer".*

8.4 Varying Execution Styles

It is conceivable that the recognition of a multi-agent event may vary as a result of a change in the pattern of execution by an actor. Such variation, however, should not be significant in a robust system as long as the temporal relations among sub-events do not change. One of the greatest difficulties in performance analysis of a multi-agent event detection system is to find an extensive set of real test data that includes all possible action execution styles by individuals. Analysis of ROC curves on synthetic data similar to the case of single-thread events may help.

We define two competing multi-agent events for ROC analysis. The first event is *"stealing"*, of which the event graph is described in section 5. The second event is *"cooperative object transfer"* and is described as a person approaching to converse with another person whose luggage is left on the ground. Then, another person comes and takes the luggage away. Discriminating these two events is extremely challenging because they are very similar. The difference between them is whether or not the view of the belongings is obstructed while there is a transfer of the ownership. The event graph representation of *"cooperative object transfer"* is shown in figure 21.

We simulate a test data set composed of twenty one *"stealing"* sequences and twenty one *"cooperative object transfer"* sequences. Simulated sequences are generated as follows. First, the trajectory of each object in the original sequence is extracted and smoothed out to obtain a principal curve. Then, four points on the principal curve are selected such that that curve is segmented into five segments of an equal length (k). The four points are then randomly perturbed with Gaussian noise $G(\mu = 0, \sigma = 2k)$ and fitted on a smooth B-spline curve. Finally, tracking data (points along the B-spline curve) is assigned according to the estimated walking speed ($13.01cm/frame$) and variance ($6.68cm/frame$) of a human. The variable lengths of the perturbed trajectories introduce variations in the timing among event threads. Therefore, we manually classify and correctly label each of the perturbed sequences.

Figure 22 shows the ROC curve for the test data set. Even though the two

events are very similar, by choosing an appropriate threshold, we can achieve the detection rate as high as 81% while keeping the false alarm rate at 16%. The main reason for misdetection is the critical event *"blocking"* that helps discriminate between *"stealing"* and *"cooperative object transfer"* can not be recognized when the persons who perform *"blocking"* action move abruptly away repeatedly during the blocking. It also fails when the blocking person walks past the reference objects shortly and comes back to block, causing the *"approach to block"* event (i.e. *"approach1"* and *"approach2"*) to be weakly recognized.

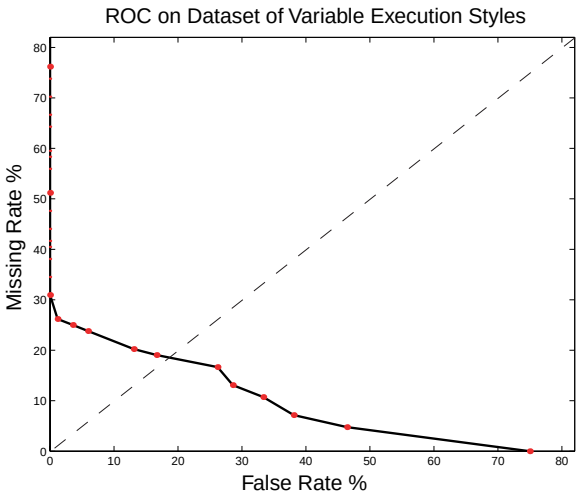


Fig. 22. *ROC curves for a data set with varying execution styles.*

9 Discussion

This paper presents an approach for both single- and multiple- actor activity recognition. We propose a transparent, generic (extendible) activity representation that provides flexibility and modularity in event modeling scheme and simplifies the task of parameter learning. Bayesian networks and probabilistic finite state automata are used to describe single actor activities. The interaction of multiple actors is modeled by an event graph, useful for plan and coordinated activity recognition. We have developed a complete system and shown, by real world examples, that possible events can be inferred and segmented from video data automatically by probabilistic analysis of the shape, motion and trajectory features of moving objects.

9.1 Computation Time

We have described in sections 6.2 and 7.1 that the complexity of the single- and multi-thread event detection algorithms are $O(NT)$ and $O(k^{R+1})$, respectively. The computation time to process a video, however, depends on other free parameters such as: the number of moving objects and the number of scene contexts (e.g. checkpoint zones, phones, pedestrian paths). Table 2 shows the computation time of two most complicated sequences together with the free parameters; *SE*, *CE*, *MT* and *Ctx* are short for the numbers of simple events, complex events, multi-thread events and contexts respectively.

Computation Time of Video Sequences					
Sequence	No. of Obj's	Frames	SE/CE/MT/Ctx	Time (sec)	fps.
ObjTransfer	3	640	83/11/3/1	453	0.71
Stealing	4	460	104/15/2/3	994	0.46

Table 2
Computation Time of Video Sequences.

We note that the computation time does not include motion detection and tracking processes. We have processed these sequences using a PII-333MHz machine with 128 MB of RAM. To convert the computation time to today's processing power (e.g. P4-2GHz with 256 MB RAM), we can approximately divide it by 8. The number of start times (t_i^k) to maintain for each sub-event of a single-thread event is set to 2 (see Section 6.2).

From Table 2, the average frame rate is approximately 9.36 fps by today's processing power. To detect *"ObjTransfer"*, all eleven complex events are defined with regard to other moving objects, which are unbound parameters. Events with an unbound parameter can significantly increase the computation time. For example, if there are three objects in the video, there will be six possible combinations of (actor, reference) pairs for each complex event to be analyzed. In the cases where the number of moving objects are high (a crowd of people), some pruning of the *(actor, reference)* pairs may be necessary.

9.2 Future Work

In the following, we provide some discussion on each component of our system and future work.

- We have demonstrated at many processing levels the use of context information for event recognition task. Knowledge of the ground plane has been used to filter moving regions and improve the tracking results. We believe

that ground plane assumption is valid in many cases and our method can easily be applied to another type of surface as long as it is known. However, this approach requires the robust detection of the feet of the objects, which may not be available in some applications. Tracking a crowd of people, necessary for many surveillance applications, is also still very difficult due to self-occlusion and the occlusion of the body parts with others. A robust filtering mechanism that integrates both the low-level (e.g. color distribution, texture) and the high-level (e.g. consistency of actions) information may be useful in such cases.

- Our activity representation is currently based on 2-D shape and trajectory features. However, it provides layers of abstraction which allows an integration of other complex features (e.g. human pose configurations, optical flow statistics) to describe more sophisticated events. An additional mechanism to derive the probability of an abstract event entity from these complex features may be required. Also, further development of the temporal and logical constraints may be required to represent a more sophisticated scenario model. For example, numerical constraints such as *"A occurs before B at least an hour ago"* or additional logical constraints such as *"no other events should happen between A and B"* may be allowed to enhance the characterization of a scenario.
- In section 8, we notice a decrease of the recognition performance on noisy sequences and on variable execution styles of activities. The recognition of the interaction of actors, in fact, relies on the accuracy of the detection and segmentation of complex events. Currently, several assumptions are made about the probability distributions in complex event modeling. Some of these assumptions (e.g. a uniform distribution of event durations) may be removed to improve the accuracy of the analysis.
- We have demonstrated a preliminary performance evaluation scheme that validates our system on some aspects of a real application. Analysis of ROC curves on synthetic data is useful when obtaining enough test data set is not possible. With an increasing number of other event recognition approaches, a more systematic performance evaluation procedure is required to make useful comparisons between algorithms.
- One concern about the scalability of our system is that the complexity of our inference method depends on the number of moving objects and the complexity of the scene (e.g. number of reference scene objects). Currently we analyze all scenarios of all possible combinations of moving objects and reference objects. The complexity can be decreased by the use task context to process only relevant scenarios and the use of heuristics (e.g. a threshold on the proximity of relevant object) to choose only the objects that may involve in a particular activity.

References

[1] H. Buxton, S. Gong, Visual surveillance in a dynamic and uncertain world, *Artificial Intelligence* **78** (1-2), 1995, 431–459.

[2] Y. Ivanov, A. Bobick, Recognition of visual activities and interactions by stochastic parsing, in *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22** (8), 2000, pp. 852–872.

[3] S. Intille, A. Bobick, Recognizing planned multiperson action, *Journal of Computer Vision and Image Understanding* **3**, 2001, 414–445.

[4] S. Hongeng, R. Nevatia, Multi-agent event recognition, in *IEEE Proceedings of the International Conference on Computer Vision, Vol. 2, Vancouver, Canada, 2001*, pp. 84–91.

[5] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, R. Nevatia, Event detection and analysis from video streams, in *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23** (8), 2001, pp. 873–889.

[6] S. Hongeng, F. Bremond, R. Nevatia, Representation and optimal recognition of human activities, in *IEEE Proceedings of Computer Vision and Pattern Recognition, Hilton Head Island, SC, 2000*, pp. 1818–1825.

[7] C. S. Regazzoni, G. Fabri, G. V. (Eds.), *Advanced Video-Based Surveillance Systems*, Kluwer Academic Publishers, 1999.

[8] R. T. Colins, A. J. Lipton, T. K. (Eds.), Special issue on video surveillance, in *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(8), 2000.

[9] J. K. Aggarwal, Q. Cai, Human motion analysis: A review, *Computer Vision and Image Understanding* **73**, 1999, 428–440.

[10] L. Davis, R. Chelappa, A. Rosenfeld, D. Harwood, I. Haritaoglu, R. Cutler, Visual surveillance and monitoring, in *DARPA Image Understanding Workshop, 1998*, pp. 73–76.

[11] J. Binder, D. Koller, S. Russell, K. Kanazawa, Adaptive probabilistic networks with hidden variables, *Machine Learning* **29**, 1997, 213–244.

[12] J. Yamato, J. Ohya, K. Ishii, Recognizing human action in time-sequential images using hidden markov model, in *IEEE Proceedings of Computer Vision and Pattern Recognition, Champaign, IL, 1992*, pp. 379–385.

[13] T. Starner, A. Pentland, Real-time american sign language recognition from video using hidden markov models, in *Proceedings of ISCV'95, 1995.*

[14] A. Wilson, A. Bobick, Recognition and interpretation of parametric gesture, in *IEEE Proceedings of the International Conference on Computer Vision, Bombay, India, 1998*, pp. 329–336.

[15] P. Remagnino, J. Orwell, G. A. Jones, Visual interpretation of people and vehicle behaviours using a society of agents, in *Italian Association for Artificial Intelligence*, *1999*, pp. 333–342.

[16] N. Oliver, B. Rosario, A. Pentland, A bayesian computer vision system for modeling human interactions, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(8), 2000, 831–843.

[17] R. Morris, D. Hogg, Statistical models of object interaction, in *proc. of the Int'l Conference on Computer Vision (ICCV) Workshop on Visual Surveillance, Bombay, India*, *1998*, pp. 209–215.

[18] T. Syeda-Mahmood, S. Srinivasan, Detecting topical events in digital video, in *Proceedings of the Eight ACM International Conference on Multimedia*, *2000*, pp. 85–94.

[19] M. R. Nephade, T. S. Huang, Detecting semantic concepts using context and audio/visual features, in *Proceedings of the IEEE workshop on Detection and Recognition of Events in Video*, *2001*, pp. 92–98.

[20] N. Oliver, E. Horvitz, A. Garg, Layered representations for recognizing office activity, in *Proceedings of the Fourth IEEE International Conference on Multimodal Interaction, Pittsburgh (PA)*, *2002*.

[21] M. Yamamoto, K. Yagishita, Scene constraints-aided tracking of human body, in *IEEE Proceedings of Computer Vision and Pattern Recognition*, *2000*, pp. 151–156.

[22] H. SidenBladh, M.J.Black, D. Fleet, Stochastic tracking of 3d human figures using 2d image motion, in *Proceedings of the European Conference on Computer Vision*, *2000*, pp. 702–718.

[23] R. von Mises, *Mathematical Theory of Probability and Statistics*, Academic Press, New York, 1964.

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

[25] L. R. Rabiner, B. H. Juang, An introduction to hidden markov models, *IEEE ASSP Magazine*, 1986, 4–16.