Keyhole Adversarial Plan Recognition for Recognition of Suspicious and Anomalous Behavior

Dorit Avrahami-Zilberbrand and Gal A. Kaminka
Computer Science Department and Gonda Brain Research Center
Bar Ilan University, Israel

Abstract

Adversarial plan recognition is the use of plan recognition in settings where the observed agent is an adversary, having plans or goals that oppose those of the observer. It is one of the key application areas of plan recognition techniques. There are two approaches to adversarial plan recognition. The first is *suspicious activity recognition*, i.e., directly recognizing plans, activities and behaviors that are known to be suspect (e.g., carrying a suitcase, then leaving it behind in a crowded area). The second is *anomalous activity recognition*, in which we *indirectly* recognize suspect behavior, by first ruling out normal, non-suspect, behaviors as explanations for the observations. Different challenges are raised in pursuing these two approaches. In this work we discuss a set of efficient plan recognition algorithms that are capable of handling the variety of challenges required of realistic adversarial plan recognition tasks. We describe an efficient hybrid adversarial plan recognition system that is composed of two processes: a plan recognizer capable of efficiently detecting anomalous behavior, and a utility-based plan recognizer incorporating the observer's own biases—in the form of a utility function—into the plan recognition process. This allows choosing recognition hypotheses based on their expected cost to the observer. These two components form a highly efficient adversarial plan recognizer capable of recognizing abnormal and potentially dangerous activities. We evaluate the system with extensive experiments, using real-world and simulated activity data, from a variety of sources.

1 Introduction

Adversarial plan recognition is the use of plan recognition in settings where the observed agent is considered a potential adversary, having plans or goals that oppose those of the observer. The objective of adversarial plan recognition is to identify behavior that is potentially harmful, differentiating it from non-adversarial behavior, in time for the observer to decide on a reaction. Applications of adversarial plan recognition include computer intrusion detection [22], virtual training environments [55], in-office visual surveillance [9], and detection of anomalous activities [16, 44, 58].

Adversarial plan recognition faces several inherent challenges:

- First, often there are limited data from which to construct a plan library of the adversary's behavior. Unlike most plan recognition settings, we cannot assume knowledge of the complete set of plans an adversary may pursue. Thus an adversarial plan recognition system has to recognize anomalous plans—plans that are characterized simply by the fact that they are not known to the observing agent.
- Second, most non-adversarial plan recognition systems ignore the expected impact of the explanations they generate, on the observer. They generate a list of recognition hypotheses (typically ranked by decreasing likelihood). It is up to the observer's decision-making component to examine the hypotheses and draw a conclusion leading to taking action. But often adversarial plan hypotheses have low likelihoods, because of their rarity. Thus they must either be ignored for being too unlikely, or they must be considered together with many other low-likelihood hypotheses (which may lead to many false positives).

  For instance, suppose we observe a rare sequence of Unix commands that can be explained by for some plan I or for a more common plan L. Most plan recognition systems will prefer the most likely hypothesis L, and ignore I. Yet, if the risk of I for the observer is high (e.g., if I is a plan to take down the computer system), then that hypothesis should be preferred *when trying to recognize suspicious behavior*. If I implies low risk (even if the observed sequence is malicious), or if the goal is not to recognize suspicious behavior, then L may be a better hypothesis.

In this chapter, we describe a comprehensive system for adversarial plan recognition. The system is composed of a hybrid plan recognizer: A symbolic recognition system, SBR (Symbolic Behavior Recognition) [3,5], is used to detect anomalous plans. Recognized non-anomalous plans are fed into an efficient utility-based plan recognition system [4], which reasons about the expected cost of hypotheses, recognizing suspicious plans even if their probability is low.

We evaluate the two components in extensive experiments, using real-world and simulated activity data, from a variety of sources. We show that the techniques are able to detect both anomalous and suspicious behavior, while providing high levels of precision and recall (i.e., small numbers of false positives and false negatives).

2 Background: Adversarial Plan Recognition

Plan recognition is the task of inferring the intentions, plans, and/or goals of an agent based on observations of its actions and observable features. Other parts of this book provide an extensive survey of past and current approaches to plan recognition. We focus here on adversarial settings, in which plan recognition is used to recognize plans, activities, or intentions of an adversary.

Applications of plan recognition to surveillance in particular, and adversarial plan recognition in general, began to appear shortly after the introduction of the basic probabilistic recognition methods (e.g., [25]), and continue to this day (e.g., [9, 14, 16, 22, 37, 39, 44, 58]). We survey recent efforts below.

We limit ourselves to *keyhole* recognition, where the assumption is that the adversary is either not aware that it is being observed, or does not care. Towards the end of the section, we discuss relaxing this assumption (i.e., allowing *intended* recognition, where the observed adversary possibly modifies its behavior because it knows it is being observed).

One key approach to adversarial plan recognition is *anomalous plan recognition*, based on anomaly detection, i.e., recognition of that which is not normal. Here, the plan library is used in an inverse fashion. The plan library is limited to accounting only for normal behavior. When a plan-recognizer is unable to match observations against the library (or generates hypotheses with very low likelihood), an anomaly is announced.

The second key approach to adversarial plan recognition is *suspicious plan recognition*, based on directly detecting threatening plans or behavior. It operates under the assumption that a plan library is available that accounts for adversarial behavior, and thus recognition of hypotheses implying such behavior is treated no differently from recognition of hypotheses implying normal activity.

In both approaches, there is an assumption that the plan library is *complete*, in that it accounts for the full repertoire of expected normal behavior (in anomalous plan recognition) or adversarial behavior (in suspicious plan recognition). Likewise, a full set of observable features can be used for either approach: not just the type of observed action, but also its duration and its effects; not just the identity of the agent being observed, but also observable features which mark it in the environment (its velocity and heading, its visual appearance, etc).

Superficially, therefore, a plan recognition system intended for one approach is just as capable of being used for the other, as long as an appropriate plan library is used. However, this is not true in practice, as different approaches raise different challenges:

Different Queries. Fundamentally, the two approaches answer different queries about the observed agent's behavior.

The main query that an anomalous behavior recognition system has to answer is a *set membership query*: Is the observed behavior explained by the plan library? If so, then no anomaly is declared. If not, then an anomaly is found. Thus the relative likelihood of the hypothesized recognized plan, with respect to other hypotheses, is unimportant[1].

In contrast, the main query that a suspicious behavior recognition system has to answer is the *current state query*: What is the preferred explanation to the observations at this point? We use the term preferred because there are different ways of ranking plan recognition hypotheses. The most

[1] Note that multiple levels of anomaly are built by using multiple plan libraries, where the membership query is applied to all, separately.

popular one relies on probability theory to rank hypotheses based on their likelihood. But as we discuss below, this is only one way to rank hypotheses.

Origins of the plan library. The construction of plan library for the two approaches, presumably by human experts or automated means, is based on knowledge sources which have different characteristics.

In many adversarial settings, knowledge of the adversarial behavior is clearly available. For instance, in applications of plan recognition in recognizing a fight between a small number of people, it is possible to imagine a specific set of recognition rules (e.g., accelerated limb motions, breaking of personal distance, etc.). If such knowledge exists, suspicious behavior recognition is more easily used.

However, in some settings, knowledge of specific adversarial plans is very lacking, and examples are rare. For instance, in surveillance for terrorist attacks in airport terminals, there are very few recorded examples, and certainly we cannot easily enumerate all possible ways in which the adversary might act prior to an attack. Here, the use of anomalous behavior recognition is more natural.

The outcome from the decision-making perspective. A challenge related to that of the different queries is the use of the output from the plan recognition module, as a basis for decision making. In suspicious plan recognition, the greater the match between the observations and the plan library (as measured by likelihood or other means), the greater confidence in categorizing the behavior as suspicious. This can be utilized by a decision-making module to respond appropriately to different levels of suspiciousness, e.g., by actively taking actions to differentiate hypotheses, or directly counter the implied threat.

But in the anomalous behavior recognition case, the opposite is true. The *weaker* the match between the observations and the plan library, the greater implied confidence in the anomaly. However, because in this case a sequence of observations is considered anomalous when it does not exist in the plan library, there is no universally accepted method for assessing confidence in the outcome. In other words, there is no universally correct method for measuring the degree to which the observations do *not* match the plan library.

We first discuss existing work utilizing anomalous plan recognition. This approach has connections to machine learning (e.g., one-class learning for security) and monitoring (e.g., fault detection and diagnosis). The literature on anomaly detection within these areas is vast. However, much of it is only superficially related, in the sense that the overall goals may be the same, but the application domains give rise to methods that are very data-oriented, yet require little or no treatment of hierarchical states, interleaved activities, or future intentions. For instance, fingerprinting users from their characteristic behavior (e.g., [36]) is a related task, but the focus is on supervised machine learning of mouse and keyboard events. Similarly, we will not address here related work on anomaly detection in data signals (e.g., [1, 15, 34]).

Within plan recognition literature, Xiang and Gong [59] adopted dynamic Bayesian networks to model normal patterns of activity captured on video. An activity is identified as abnormal if the likelihood of being generated by normal models is less then a threshold. Duong et al. [16] use Switching Semi-Markov Models to represent user activities that have known duration (so an anomaly is detected if actions' durations deviate from normal). Marhasev et al. [42] look at low-likelihood hypotheses based both on duration and state, in the context of a clinical diagnosis task. Yin et al. [61] present a two phase approach to detect anomalous behavior based on wireless sensors attached to the human body. This approach first uses Support Vector Machines (SVMs) which filter out the activities with a high probability of being normal, then derives from these an anomalous activity model. Here also the model is learned based on normal activities.

One key issue with all of these investigations is that they spend computational time on probabilistic reasoning which might not be necessary for a membership query. In contrast, we advocate an approach that utilizes symbolic behavior recognition, which can answer the membership query much more efficiently.

In particular, the SBR (Symbolic Behavior Recognition) algorithms [3,5], advocated in this chapter, provide extremely efficient membership query response, at the expense of recognition accuracy (since they cannot rank hypotheses based on likelihood). The SBR algorithms are most related to symbolic plan recognition algorithms used to identify failures in teamwork. RESL [33], and later YOYO [29], are symbolic recognition algorithms for detecting disagreements among team members. Both RESL and YOYO exploit knowledge of the social structures of the team (called team hierarchy) to efficiently recognize splits in teams, where an agent is executing a different plan than agreed upon with the rest of its team. Disagreements are therefore treated as recognition failures. Both algorithms ignore observation history in the current state hypotheses, in contrast to SBR. Moreover, these algorithms do not account for complex temporal reasoning (duration and interleaving), while SBR algorithms do.

We now turn to discussing work in suspicious behavior recognition. It operates under the assumption that the plan library directly covers the adversarial behavior to be recognized. Thus recognition of hypotheses implying such behavior is treated no differently from recognition of hypotheses implying normal activity. Examples include systems that encode possible attacks in an intrusion detection system, trying to predict whether an attack is performed [22]; systems that focus on recognizing specific suspicious behaviors in train stations [14], such as fighting; systems with an a priori set of attack templates [28] that are compared against observations to infer whether a particular terrorist attack matches one or more templates; and models with representation of human activities based on tracked trajectories e.g., [44].

Most suspicious behavior recognition systems utilize probabilistic plan recognition approaches. These focus on determining the most likely hypothesis, or set of hypotheses, as to the current and/or past states of the agent. Previous work in this area has focused on utilizing specialized structures and techniques that allow more efficient recognition, or more expressive forms of plan recognition.

Some of the early work applying plan recognition to visual surveillance applied belief (Bayesian) networks (e.g., [25,26]). However, given the known computational hardness of exact and approximate reasoning in general belief networks [13], and general dynamic belief networks [7, 35], more efficient models were needed. These typically trade expressiveness for run-time, e.g., by introducing a Markov assumption into the models.

Indeed, Hidden Markov Models (*HMMs*) and extensions have been explored extensively for plan recognition. These are special cases of dynamic belief networks, with much improved inference run-time (typically, polynomial), at the cost of introducing a Markovian assumption and restricting the the set of variables. HMMs are widely used in activity recognition of simple activities (e.g., [51, 60]). However, they do not allow for complex activities (e.g., interacting agents, hierarchical decomposition of the activities, or explicit temporal dependencies). *Coupled HMMs* [8] and *Factorial HMM* [24] are extensions of the HMM with multiple hidden interacting chains, e.g., for modeling interactions between observed agents. *AHMM* [10], *HHMM* [17] and *Layered HMM* [46] are capable of handling activities that have hierarchical structure. These models are used in activity recognition applications such as learning and detecting activities from movement trajectories [43] and inferring user's daily movements from raw GPS sensor data [37]. *Hidden Semi-Markov Models* (*HSMMs*) allow probabilistic constraints over the duration of plans [16]. Extensions to these can model dependencies on state durations [42]. More recent work has continued developing methods for greater expressiveness without sacrificing tractability. For example, Blaylock and Allen [6] provided a novel HMM-based model that allows efficient exact reasoning about hierarchical goals. Hu and Yang [27] model interacting and concurrent goals.

Recent work has moved beyond dynamic belief networks and Markov models to use *Conditional Random Fields* [56] which allow for greater expressiveness. Bui et al. [11] introduce a Hidden Permutation model that can learn the partial ordering constraints in location-based activity recognition. There are also approaches to plan recognition based on parsing, in particular utilizing *probabilistic grammars*, which allow for efficient exact inference (e.g., [18–20, 49, 50]).

The key challenge to the use of probabilistic recognizers in suspicious behavior recognition is that most often, adversarial behavior is not likely, typically because it is rare. Given a specific sequence of observations, if the adversarial hypothesis is the only explanation, then of course it would be selected. But in general in plan recognition, there would be multiple hypotheses that match the observations, and have to be ranked in some fashion. In this general case, preferring the most likely hypothesis often means preferring the more common explanation, rather than the rare explanation signifying adversarial intent.

In other words, the posterior likelihood of an hypothesis signifying adversarial behavior is very often low, given the observations. In the general case, it would be one hypothesis out of many. This leads to either of two cases: Either the observer has to always consider low-likelihood hypotheses (leading to many false alarms, i.e., false positives), or the observer sets a high threshold for likelihood, in which cases she risks disregarding a suspicious behavior.

To address this, we posit that a system for suspicious plan recognition needs to reason about the *expected cost* to the observer, given that a hypothesis is correct. In other words, we argue that the observer ranks hypotheses based on their expected cost to itself, rather than just based on their likelihood. This way, a low-likelihood hypothesis which, if true, would signify large costs, may be preferred over a more likely hypothesis, which implies no risk. Put more generally, our work differs significantly from probabilistic approaches, as we advocate reasoning about the expected utility of hypotheses, rather than simply their likelihoods.

Essentially every probabilistic plan recognition system can be used as the basis for expected cost computations. This can be done by taking the *entire* (ranked) set of hypotheses from the probabilistic recognizers, and then externally associating each with costs, to compute expected costs. Computationally, this can be very expensive, as in addition to the plan recognition computation, there would be additional external computation of the expected costs. Note that the entire set of hypotheses must be processed, as a low-likelihood hypothesis (ranked low by the probabilistic recognizer) may end up implying high expected utility.

The key to efficient computation of plan recognition hypotheses, ranked by their expected costs, is the folding of the expected cost computation into the plan recognizer itself. The hypotheses would then be ranked directly, without the need for external computation.

We call this approach *UPR* (Utility-based Plan Recognition), and describe it fully in [2, 4], and more briefly in this chapter. The origins of this approach come from a worst-case reasoning heuristic, applied in Tambe and Rosenbloom's *RESC* algorithm [55]. RESC is a reactive plan recognition system, applied in simulated air-combat domains. Here, the observing agent may face ambiguous observations, where some hypotheses imply extreme danger (a missile being fired towards the observer), and other hypotheses imply gains (the opponent running away). RESC takes a heuristic approach that prefers hypotheses that imply significant costs to the observer (e.g., potential destruction). However, the relative likelihood of such hypotheses was ignored, as RESC was a symbolic plan recognizer. While we are inspired by this work, we take a principled, decision-theoretic approach. In the algorithms we present, the likelihood of hypotheses is combined with their utilities, to calculate the expected impact on the observer. We show later that this subsumes the earlier, heuristic work.

The UPR techniques described in this chapter utilize a technique for gaining the efficiency of symbolic plan recognition for recognizing anomalous plans, with a Markovian utility-based plan recognizer [4]. The Markovian assumption, which allows for efficient reasoning, comes at a cost of expressiveness. A more general UPR method, based on influence diagrams, is described in [2]. It provides one way of extending the expressiveness of UPR, but comes at a significant computational cost.

Recently, UPR has been extended in [52] to allow dynamic changes to the cost function, a specific extension which maintains the computational tractability, while increasing the expressiveness of the original Markovian technique described in [4] and in this chapter. This allows, for instance, expressing non-linearly increasing costs for observing repeated occurrences of a suspicious event (e.g., repeatedly trying to avoid a cop whenever one passes by).

There have been related investigations of the use of utility and cost in plan recognition. However, these focused on modeling the expected utility of the observed agent, not of the observer. They thus do not address the same challenge we do in recognizing suspicious plans. Indeed, it is arguable that modeling the utility of different plans to the observed agent is redundant, since a rational observed agent would be pursuing the plan with highest expected utility, and this would be reflected in the likelihood of the plan, as observed. Nevertheless, there are investigations of explicit modeling utilities of the observed agent. These include Mao and Gratch [40,41], Suzic [54], Pynadath and Marsella [47,48], and Sukthankar and Sycara [53].

The techniques described in this chapter utilize a technique for gaining the efficiency of symbolic plan recognition for recognizing anomalous plans, with a Markovian utility-based plan recognizer [4]. Here, an underlying symbolic recognizer (SBR, described in [3, 5]) is used to efficiently filter inconsistent hypotheses, passing them to a probabilistic inference engine, which focuses on ranking recognition hypotheses using UPR principles.

There have been a number of other plan recognizers which use a hybrid approach, though of a different nature. Geib and Harp [23] developed PHATT, a hybrid symbolic/probabilistic recognizer, where a symbolic algorithm filters inconsistent hypotheses before they are considered probabilistically. PHATT assumes instantaneous, atomic actions. It takes the following approach: With each observation, the symbolic algorithm generates a *pending set* of possible expected observations, which are matched against the next observation to maintain correct state history hypotheses. The size of the pending set may grow exponentially [21]. In contrast, SBR decouples the current state and state history queries, and incrementally maintains hypotheses implicitly, without predicting pending observations. The hypotheses are thus computed only when needed (when hopefully, many of them have been ruled out). PHATT does allow recognition of interleaved plans as well as some partial observability. However, the expected costs of plans are ignored.

YOYO* [32] is a hybrid symbolic/probabilistic plan recognition algorithm for multi-agent *overhearing* (recognition based on observed communication acts). The plan library used by YOYO* included information about the average duration of plan steps, which is used to calculate the likelihood of an agent terminating one step and selecting another without being observed to do so. In this, YOYO* addressed missing observations (though their likelihood of becoming lost is assumed to be provided a priori). However, in contrast to our work, YOYO* did not address matching multi-feature observations (where some features may be intermittently lost), nor did it allow for interleaved plans.

Quite recently, Lisy et al. have examined the problem of *intended* adversarial plan recognition [38], where the observed agent is actively attempting to prevent its plans from being recognized. They define the problem using game-theory, where the observer and observed play a game of selecting plans, and actions for recognizing these plans. While the game tree grows very large for even moderate-size problems, Lisy et al. show that its specialized structure lends itself to highly-scalable algorithms for determining the Nash equilibrium solution. However, in contrast to the work described in this chapter and elsewhere in this book, the algorithms avoid any details of matching observations to plans, which is, from a computational standpoint, a key challenge in plan recognition.

3 An Efficient Hybrid System for Adversarial Plan Recognition

We present here a hybrid anomalous and suspicious adversarial plan recognition system that uses efficient plan recognition algorithms for detecting anomalous and suspicious behaviors. The system is composed from two modules: SBR (Symbolic Behavior Recognition), initially presented in [3,5], and UPR (Utility Based Plan Recognition), presented in [2,4].

The system uses a plan library, which encodes normal behavior, including plans, that may or may not indicate a threat (i.e., may be suspicious). The SBR module extracts coherent hypotheses from the observation sequence. If the set of hypotheses is empty, it declares the observation sequence as anomalous. If the set is not empty, then the hypotheses are passed to the UPR module that computes the highest expected cost hypotheses. When the expected cost of the top-ranked hypothesis reaches a given threshold, the system declares that the observation sequence is suspicious. Figure 1 presents the hybrid system. The input to the system is an observation sequence and the output is whether anomalous or suspicious behavior is detected.

We begin by considering recognition of anomalous behavior using the symbolic plan recognition algorithm (Section 3.1). Here, the plan library represents normal behavior; any activity which does not match the plan library is considered abnormal. This approach can be effective in applications where we have few or no examples of suspicious behavior (which the system is to detect), but many examples for normal behavior (which the system should ignore). This is the case, for instance, in many vision-based surveillance systems, in public places.

A symbolic plan recognition system is useful in recognition of abnormal patterns, such as walking in the wrong direction, taking more than usual amount of time to get to the security check, etc. The symbolic recognizer can efficiently match activities to the plan library and rule out hypotheses that do not match. When the resulting set of matching hypotheses is empty, the sequence of observations is flagged as anomalous. The symbolic algorithm is very fast, since it rejects or passes hypotheses without ranking them.

However, detection of abnormal behavior is not sufficient. There are cases

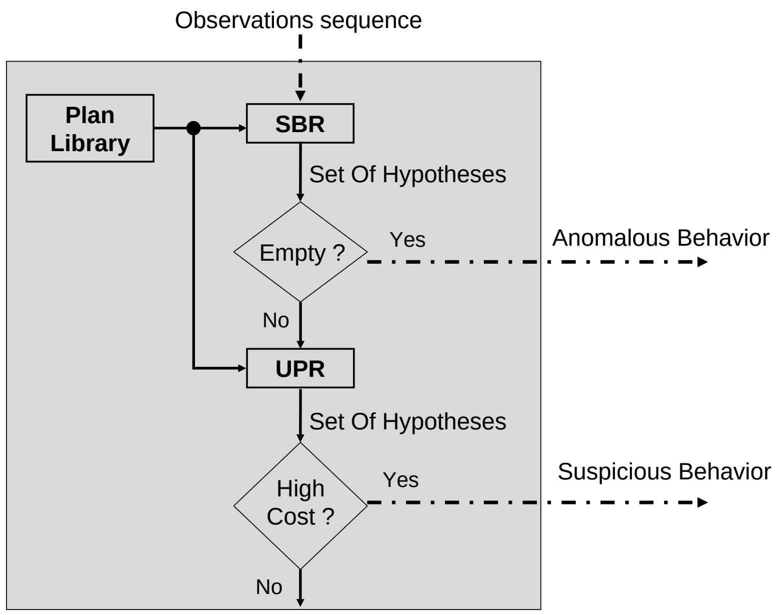

Fig. 1: A hybrid adversarial plan recognition system. Inputs and outputs for the system are in dashed arrows.

where a normal behavior should be treated as suspicious. In these cases, we cannot remove the behavior from the plan library (so as to make its detection possible using the anomalous behavior detection scheme outlined above), and yet we expect the system to detect it and flag it.

In Section 4.2, we present the use of UPR for recognizing suspicious behavior. Here the plan library explicitly encodes behavior to be recognized, alongside any costs associated with the recognition of this behavior. This allows the UPR system to rank hypotheses based on their expected cost to the observing agent. As we shall see, this leads to being able to recognize potentially dangerous situations, despite their low likelihood.

3.1 Efficient Symbolic Plan Recognition

We propose an anomalous behavior recognition system, the input of the system is an observation sequence. The system is composed of an SBR (Symbolic Behavior Recognition) module, which extracts coherent hypotheses from the observation sequence. If the set of hypotheses is empty, it declares the observation sequence as anomalous.

The symbolic plan recognition algorithm, briefly described below, is useful in recognition of abnormal behavior, since it is very fast (no need to rank hypotheses), and can handle key capabilities required by modern surveillance applications. The reader is referred to [3, 5] for details.

SBR's plan library is a single-root directed graph, where vertices denote *plan steps*, and edges can be of two types: Decomposition edges decompose plan steps into sub-steps, and sequential edges specify the temporal order of execution. The graph is acyclic along decomposition transitions.

Each plan has an associated set of conditions on observable features of the agent and its actions. When these conditions hold, the observations are said to match the plan. At any given time, the observed agent is assumed to be executing a *plan decomposition path*, root-to-leaf through decomposition edges. An observed agent is assumed to change its internal state in two ways. First, it may follow a sequential edge to the next plan step. Second, it may reactively interrupt plan execution at any time, and select a new (first) plan.

The recognizer operates as follow: First, it matches observations to specific plan steps in the library according to the plan step's conditions. Then, after matching plan steps are found, they are tagged by the time-stamp of the observation. These tags are then propagated up the plan library, so that complete plan-paths (root to leaf) are tagged to indicate they constitute hypotheses as to the internal state of the observed agent when the observations were made. The propagation process tags paths along decomposition edges. However, the propagation process is not a simple matter of following from child to parent. A plan may match the current observation, yet be *temporally inconsistent*, when a history of observations is considered. SBR is able to quickly determine the temporal consistency of a hypothesized recognized plan [3].

At the end of the SBR process we are left with a set of *current-state hypotheses*, i.e., a set of paths through the hierarchy, that the observed agent may have executed at the time of the last observation.

The overall worst-case run-time complexity of this process is $O(LD)$ [3]. Here, L is the number of plan-steps that directly match the observations; D is the maximum depth of the plan library. L is typically very small, the number of specific actions that can match a particular observation.

We presented above the basics of the symbolic plan recognition model. Extensions to this model address several challenges: (a) reducing space complexity of matching complex multi-featured observations to the plan library; (b) dealing with plan execution duration constraints; (c) handling lossy observations (where an observation is intermittently lost); and (d) handling interleaved plans (where an agent interrupts a plan for another, only to return to the first later). These are discussed in [2, 5].

3.2 UPR: Efficient Utility-based Plan Recognition

This section presents an efficient hybrid technique that is used for recognizing suspicious behavior. UPR (Utility-based Plan Recognition) allows the observer to incorporate a utility function into the plan recognition process. With every potential plan recognition hypothesis, we associate a utility to the observer, should the hypothesis be correct. In adversarial UPR, this utility is the cost incurred by the observer. This allows choosing recognition hypotheses based on their expected cost to the observer, even if their likelihood is low.

The highly efficient symbolic plan recognizer that was introduced in 3.1 is used to filter hypotheses, maintaining only those that are consistent with the observations (but not ranking the hypotheses in any way). We then add an expected utility aggregation layer, which is run on top of the symbolic recognizer.

We briefly describe the basics of UPR recognizer in this section. Avrahami-Zilberbrand [2] presents a general UPR framework, using influence diagrams as the basis for reasoning. Unfortunately, such reasoning is computationally expensive. Relying on a Markovian assumption, a less expressive (but much more efficient) UPR technique is used here. For its details, consult [2, 4].

After getting all *current state hypotheses* from the symbolic recognizer, the next step is to compute the expected utility of each hypothesis. This is done by multiplying the posterior probability of a hypothesis, by its utility to the observer.

We follow in the footsteps of—and then extend—*Hierarchical Hidden Markov Model* (HHMM) [17] in representing probabilistic information in the plan library. We denote plan-steps in the plan library by q_i^d, where i is the plan-step index and d is its hierarchy depth, $1 \leq d \leq D$. For each plan step, there are three probabilities:

Sequential transition. For each internal state q_i^d, there is a state transition probability matrix denoted by $A^{q^d} = (a_{i,j}^{q^d})$, where $a_{i,j}^{q^d} = P(q_j^d | q_i^d)$ is the probability of making a sequential transition from the i^{th} plan-step to the j^{th} plan-step. Note that self-cycle transitions are also included in A^{q^d}.

Interruption. We denote by $a_{i,end}^{q^d}$ a transition to a special plan step end^d

which signifies an interruption of the sequence of current plan step q_i^d, and immediate return of control to its parent, q^{d-1}.

Decomposition transition. When the observed agent first selects a decomposable plan step q_i^d, it must select between its (first) children for execution. The decomposition transition probability is denoted $\Pi^{q^d} = \pi^{q^d}(q^{d+1}) = P(q_k^{d+1}|q_i^d)$, the probability that plan-step q_i^d will initially activate the plan-step q_k^{d+1}.

Observation Probabilities. Each leaf has an output emission probability vector $B^{q^d} = (b^{q^d}(o))$. This is the probability of observing o when the observed agent is in plan-step q^d. For presentation clarity, we treat observations as children of leaves, and use the decomposition transition Π^{q^d} for the leaves as B^{q^d}.

In addition to transition and interruption probabilities, we add utility information on the edges in the plan library. The utilities on the edges represent the cost or gains to the observer, given that the observed agent selects the edge. For the remainder of the work, we use the term cost to refer to a positive value associated with an edge or node. As in the probabilistic reasoning process, for each node we have three kinds of utilities: (a) E^{q^d} is the sequential transition utility (cost) to the observer, conditioned on the observed agent transitioning to the next plan-step, paralleling A^{q^d}; (b) $e_{i,end}^{q^d}$ is the interruption utility; and (c) Ψ^{q^d} is the decomposition utility to the observer, paralleling Π^{q^d}.

Figure 2 shows a portion of the plan library of an agent walking with or without a suitcase in the airport, occasionally putting it down and picking it up again, an example discussed below. Note the *end* plan step at each level, and the transition from each plan-step to this end plan step. This edge represents the probability to interrupt. The utilities are shown in diamonds (we omitted zero utilities, for clarity). The transitions allowing an agent to leave a suitcase without picking it up are associated with large positive costs, since they signify danger to the observer.

We use these probabilities and utilities to rank the hypotheses selected by the SBR. First, we determine all paths from each hypothesized leaf in time-stamp $t-1$, to the leaf of each of the current state hypotheses in time stamp t. Then, we traverse these paths multiplying the transition probabilities on edges by the transition utilities, and accumulating the utilities along the paths. If there is more than one way to get from the leaf of the previous hypothesis to the leaf of the current hypothesis, then it should be accounted for in the accumulation. Finally, we can determine the *most costly* current plan-step (the current-state hypothesis with maximum expected cost). Identically, we can also find the *most likely* current plan-step, for comparison.

Formally, let us denote hypotheses at time $t-1$ (each a path from root to leaf) as $W = \{W_1, W_2, ..., W_r\}$, and the hypotheses at time t as $X = \{X_1, X_2, ..., X_l\}$. To calculate the maximum expected-utility (most costly) hypothesis, we need to calculate for each current hypothesis X_i its expected cost to the observer, $U(X_i|O)$, where O is the sequence of observations thus far. Due to the use of SBR to filter hypotheses, we know that the $t-1$ observations in O have resulted

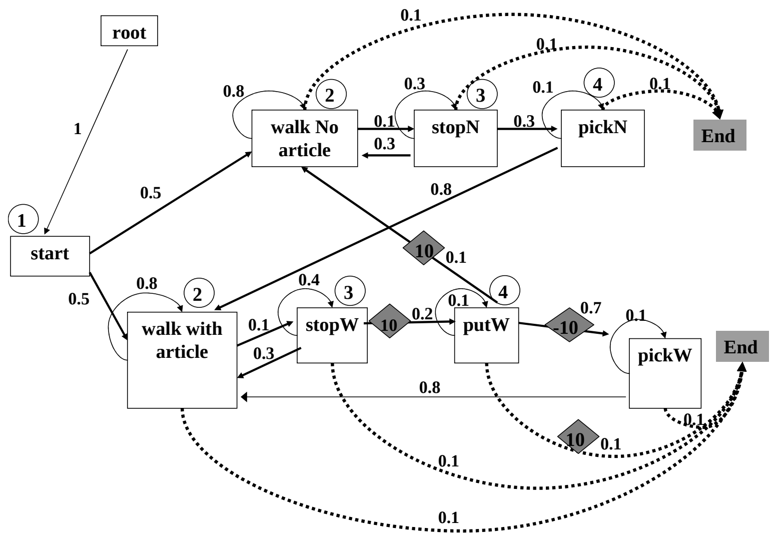

Fig. 2: An example plan library. Recognition time-stamps (example in text) appear in circles. Costs appear in diamonds.

in hypotheses W, and that observation t results in new hypotheses X. Therefore, under assumption of Markovian plan-step selection, $U(X_i|O) = U(X_i|W)$.

The most costly hypothesis is computed in Equation 1. We use $P(W_k)$, calculated in the previous time-stamp, and multiply it by the probability and the cost to the observer of taking this step from W_k to X_i. This is done for all i, k.

$$\hat{X_i} = \operatorname*{argmax}_{X_i} \sum_{W_k \in W} P(W_k) \cdot P(X_i|W_k) \cdot U(X_i|W_k) \qquad (1)$$

To calculate the expected utility $E(X_i|W_k) = P(X_i|W_k) \cdot U(X_i|W_k)$, let X_i be composed of plan steps $\{x_i^1, ..., x_i^m\}$ and W_k be composed of $\{w_k^1, ..., w_k^n\}$ (the upper index denotes depth). There are two ways in which the observed agent could have gone from executing the leaf $w^n \in W_k$ to executing the leaf $x^m \in X_i$: First, there may exist $w \in W_k$, $x \in X_i$ such that x and w have a common parent, and x is a direct decomposition of this common parent. Then, the expected utility is accumulated by climbing up vertices in W_k (by taking *interrupt* edges) until we hit the common parent, and then climbing down (by taking first child decomposition edges) to x^m. Or, in the second case, x^m is reached by following a sequential edge from a vertex w to a vertex x.

In both cases, the probability of climbing up from a leaf w^n at depth n, to a parent w^j (where $j < n$) is given by

$$\alpha^j_{w^n} = \prod_{d=n}^{j} a^d_{w,\text{end}} \tag{2}$$

and the utility is given by

$$\gamma^j_{w^n} = \sum_{d=n}^{j} e^d_{w,\text{end}}. \tag{3}$$

The probability of climbing down from a parent x^j to a leaf x^m is given by

$$\beta^j_{x^m} = \prod_{d=j}^{m} \pi^{x^d}(x^{d+1}) \tag{4}$$

and the utility is given by

$$\delta^j_{x^m} = \sum_{d=j}^{m} \psi^{x^d}(x^{d+1}). \tag{5}$$

Note that we omitted the plan-step index, and left only the depth index, for presentation clarity.

Using α^j_w, β^j_x, γ^j_w and δ^j_x, and summing over all possible j's, we can calculate the expected utility (Equation 6) for the two cases in which a move from w_n to x_m is possible .

$$\begin{aligned} E(X_i|W_k) &= P(X_i|W_k) \times U(X_i|W_k) \\ &= \sum_{j=n-1}^{1} [(\alpha^j_w \cdot \beta^j_x) \times (\gamma^j_w + \delta^j_x) \times \text{Eq}(x^j, w^j)] \\ &+ \sum_{j=n-1}^{1} [\alpha^j_w \cdot a^j_{w,x} \cdot \beta^j_x] \times (\gamma^j_w + e^j_{w,x} + \delta^j_x) \end{aligned} \tag{6}$$

The first term covers the first case (transition via interruption to a common parent). Let $\text{Eq}(x^j, w^j)$ return 1 if $x^j = w^j$, and 0 otherwise. The summation over j accumulates the probability multiplying the utility of all ways of interrupting a plan w^n, climbing up from w^n to the common parent $x^j = w^j$, and following decompositions down to x^m.

The second term covers the second case, where a sequential transition is taken. $a^j_{w,x}$ is the probability of taking a sequential edge from w^j to x^j, given that such an edge exists ($a^j_{w,x} > 0$), and that the observed agent is done in w_j. To calculate the expected utility, we first multiply the probability of climbing up to a plan-step that has a sequential transition to a parent of x^m, then we multiply in the probability of taking the transition, and then we multiply in the probability of climbing down again to x^m. Then, we multiply in the utility summation along this path.

A naive algorithm for computing the expected costs of hypotheses at time t can be expensive to run. It would go over all leaves of the paths in $t-1$ and for each of these, traverse the plan library until getting to all leaves of paths we got in time-stamp t. The worst-case complexity of this process is $O(N^2T)$, where N is the plan library size, and T is the number of observations.

In [2] we presented a set of efficient algorithms that calculates the expected utilities of hypotheses (Equation 1) in worst-case runtime complexity $O(NDT)$, where D is the depth of the plan library (N,T are as above). The reader is referred to [2] for details.

4 Detecting Anomalous and Suspicious Behavior: Experiments

In this section we evaluate the system that was described in Section 3 in realistic problems of recognizing both types of adversarial plans: anomalous and suspicious behaviors. We first discuss recognition of anomalous behavior (Section 4.1). In Section 4.2, we turn to the use of UPR for recognizing suspicious behavior.

4.1 Detecting Anomalous Behavior

We evaluate the use of SBR for anomalous behavior recognition, using real-world data from machine vision trackers, which track movements of people, and report on their coordinate positions. We conduct our experiments in the context of a vision-based surveillance application. The plan library consists of discretized trajectories which correspond to trajectories that are known to be valid. We use a specialized learning algorithm, described briefly in Section 4.1.1, to construct this plan library.

Section 4.1.2 describes in detail the experiments we conducted, along with their results on video clips and data from the CAVIAR project [12]. Section 4.1.3 presents results from data sets gathered as part of our participation in a commercial R&D Consortium (AVNET), which developed technologies for detection of criminal or otherwise suspicious objects and suspects.

To evaluate the results of the SBR algorithms we distinguish between two classes of observed trajectories: anomalous and non-anomalous. The true positives are the anomalous trajectories that were classified correctly. True negatives are, similarly, non-anomalous trajectories that were classified correctly. The false positives are the non-anomalous trajectories that were mistakenly classified as anomalous. The false negatives are the number of anomalous trajectories that were not classified as anomalous.

We use the *precision* and *recall* measures that are widely used in statistical classification. A perfect precision score of 1 means that every anomalous trajectory that was labeled as such was indeed anomalous (but this says nothing about

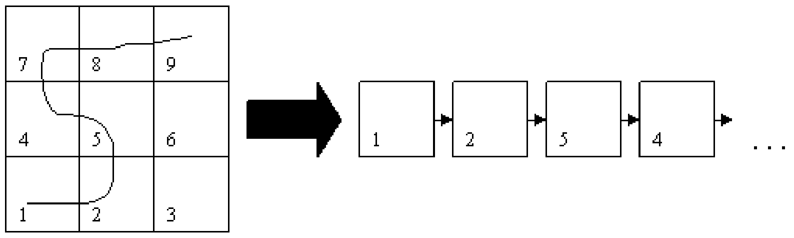

Fig. 3: Result of running naive learning algorithm on one trajectory

anomalous trajectories that were classified as non-anomalous). A perfect recall score of 1 means that all anomalous trajectories were found (but says nothing about how many non-anomalous trajectories were also classified as anomalous). The reader is referred to [57] for a detailed description.

4.1.1 The Learning Algorithm

We use a simple learning algorithm that was developed for the purpose of building plan recognition libraries based on examples of positive (valid) trajectories only. The learning algorithm is fully described in [2, 30, 31]. We provide a description below of its inputs and outputs, as those are of interest here.

The learning algorithm L receives a *training set* that contains observation sequences, S. Each observation sequence $s \in S$ is one trajectory of an individual target that is composed of all observations (samples) o_i, where i is an ordering index within s. Each observation o_i is a tuple $\langle x_i, y_i, t_i \rangle$, where x_i, y_i are Cartesian coordinates of points within W, and t is a time index.

The learning algorithm divides the work area W using a regular square-based grid. Each observation $\langle x_i, y_i, t_i \rangle$ is assigned a square that contains this point. For each trajectory of an individual target (s) in the *training set*, it creates a sequence of squares that represents that trajectory.

The output of the algorithm is a set K of discretized trajectories, each a sequence of grid cells, that are used together as the plan library for the recognition algorithm. Figure 3 shows a work area that is divided into nine squares, with one trajectory. The output of the algorithm is the sequence of squares that defines that trajectory (on the right of the figure).

The grid's square cell size is an input for the learning algorithm. By adjusting the size of the square we can influence the relaxation of the model. By decreasing the size of the square the learned library is more strict (there is less generalization); and in contrast, too large a value would cause over-generalization. Small square size may result in over-fitting; a trajectory that is very similar to an already seen trajectory, but differs slightly will not fit the model. By increasing the size of the square, the model is more general and it is less sensitive to noise, but trajectories that are not similar might fit.

People are rarely twice in the exact same position. As a result, the training

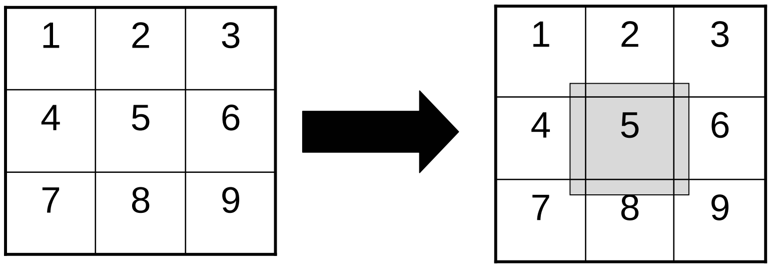

Fig. 4: Demonstrating position overlap. Added position overlap for square number 5.

set may contain many sample trajectories that differ by very small distance. Part of the challenge in addressing this lies in adjusting the square size, as described above. However, often a part of a trajectory would fall just outside the cell that contains the other examples of the same trajectories.

To solve this, the learning algorithm has another input parameter, called *Position Overlap Size*. The position overlap prevents over-fitting to the training data, by expanding each square such that it overlaps with those around it (see Figure 4, where the position overlap is shown for square number 5). Any point in a trajectory that lies in an overlapping area is defined to match both the overlapping square, as well as the square within which it falls. Thus, for instance, a point within cell 3 in Figure 4, at its bottom left corner (within the darkened overlapping area of cell 5) would match cell 3 and 5, as well as 6 and 2 (since these cells also have their own overlapping areas). Essentially, this is analogous to having non-zero observation emitting probabilities for the same observation from different states in a Hidden Markov Model.

4.1.2 Evaluation using CAVIAR Data

We utilize two sets of real-world data to evaluate SBR's performance as an anomalous behavior recognizer. Experiments with the first set are described in this section. The second set is described in Section 4.1.3.

The first set of experiments were conducted on video clips and data from the CAVIAR Project[2] [12]. The CAVIAR project contains a number of video

[2] EC Funded CAVIAR project/IST 2001 37540, found at URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

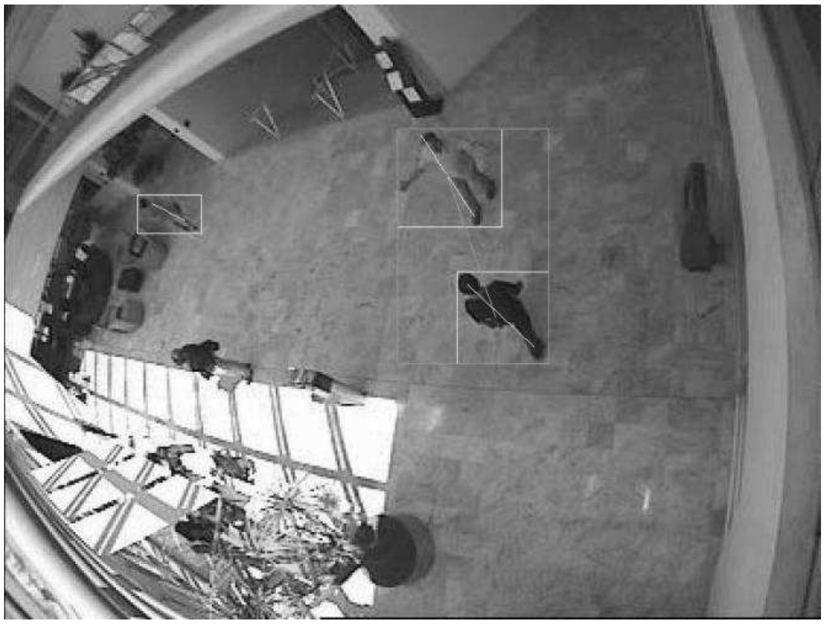

Fig. 5: A typical frame of image sequence in CAVIAR Project

clips with different scenarios of interest: People walking alone, standing in one place and browsing, etc. The videos are 384×288 pixels, 25 frames per second. Figure 5 shows a typical frame. Originally, the CAVIAR data was gathered to allow comparative evaluations of machine vision tracking systems. To do this, the maintainers of the CAVIAR data set determined the ground truth positions of subjects, in pixel coordinates, by hand-labeling the images.

We use the ground-truth data to simulate the output of realistic trackers, at different levels of accuracy. To do this, each ground-truth position of a subject, in pixel coordinates, is converted by homography (a geometric transformation commonly used in machine vision) to a position, in the 2D plane on which the subjects move (in centimeters). Then we add noise with normal distribution to simulate tracking errors. Higher variance simulates less accurate trackers, and low variance simulates more accurate trackers. In the experiments reported on below, we use a standard deviation of 11cm diagonal (8cm vertical and horizontal). The choice of noise model and parameters is based on information about state-of-the-art trackers (e.g., [45]).

To create a large set of data for the experiments (representing different tracking instances, i.e., the tracking results from many different video clips), we simulated multiple trajectories of different scenarios, and trained the learning system on them, to construct a plan library. This plan library was then used with different trajectories to test the ability of the algorithms to detect abnormal behavior.

In the first experiment we tested simple abnormal behavior. We simulated three kinds of trajectories:

1. *Curved path A*. Taken from the first set in CAVIAR, titled *One person walking straight line and return*. In this video, a subject is shown walking along a path, and then turning back. We took the first part, up to the turn back, as the basis for normal behavior in this trajectory.

2. *U-Turn*. As above, but then including also the movement back.

3. *Curved path B*. Taken from the CAVIAR set titled *One person walking straight*, which is similar to the above, but curves differently towards the end. We use this to evaluate the system's ability to detect abnormal behavior (e.g., in comparison to *Curved path A* above).

Figure 6 shows the three kind of trajectories. The arrow shows the starting position of the trajectories. The end-points lie at the other end (movement right to left).

We created 100 simulated trajectories of each type, for a total of 300 trajectories. We trained a model on 100 noisy trajectories from *Curved path A*, using the learning system described in Section 4.1.1. In this experiment we fixed the grid cell size to 55cm, and we vary the plan library relaxation parameter (called *Position Overlap Size* in Section 4.1.1). The cell size was chosen such that it covered the largest distance between two consecutive points in the training set trajectories.

Figure 7 shows the true positives versus false positives. The x axis is the plan library relaxation, and the y axis is the number of trajectories (subjects). We can see that the system starts stabilizing in plan library relaxation of 11 (the number of false positives is zero), and after a plan library relaxation of 15 the system is too relaxed; the number of abnormal trajectories that are *not* detected slowly increases.

Figure 8 shows the precision and recall of the system. The x axis is the plan library relaxation, and the y axis is the number of trajectories (subjects). The precision increases till perfect score of 1 from relaxation 11. The recall starts with perfect score of 1 and decreases slowly from relaxation 16. As in figure 7, we can see that for values of plan library relaxation of 11–15, we get perfect precision and perfect recall of 1.

Despite the optimistic picture that the results above portray, it is important to remember that these results ignore the time for detection. However, in practice the time for detection matters.

Figure 9 shows the time for detecting the abnormal behavior with standard deviation bars. The X axis measures the plan library relaxation parameter range, and the Y axis measures the time (in seconds) passed until the detection of abnormal behavior. In this figure we can see that until a relaxation of 12cm, the time for detection is negative, since we detect too early—before the abnormal

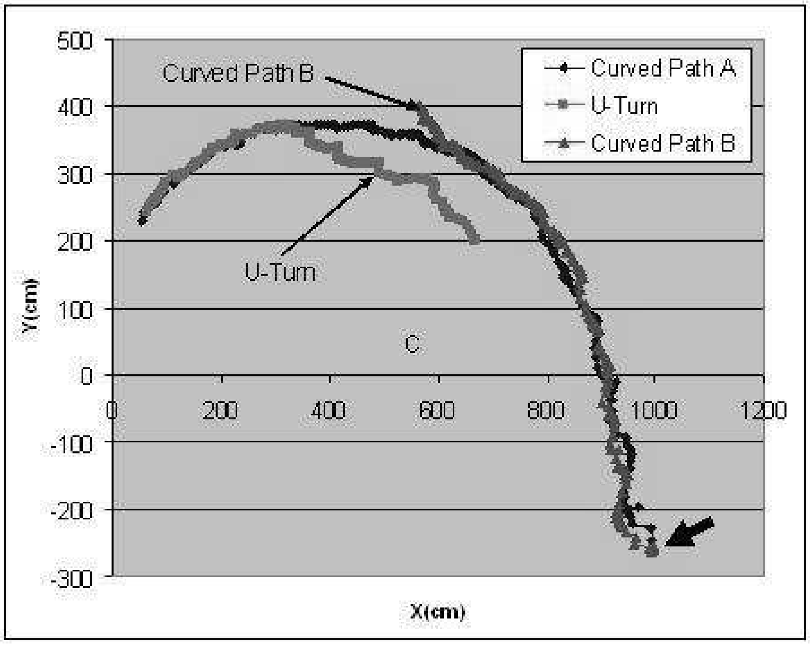

Fig. 6: Three Trajectories: Legal path (Curved Path A), suspicious path (Curved Path B), and return path (U-Turn) from CAVIAR data. The thick arrow points at the starting point.

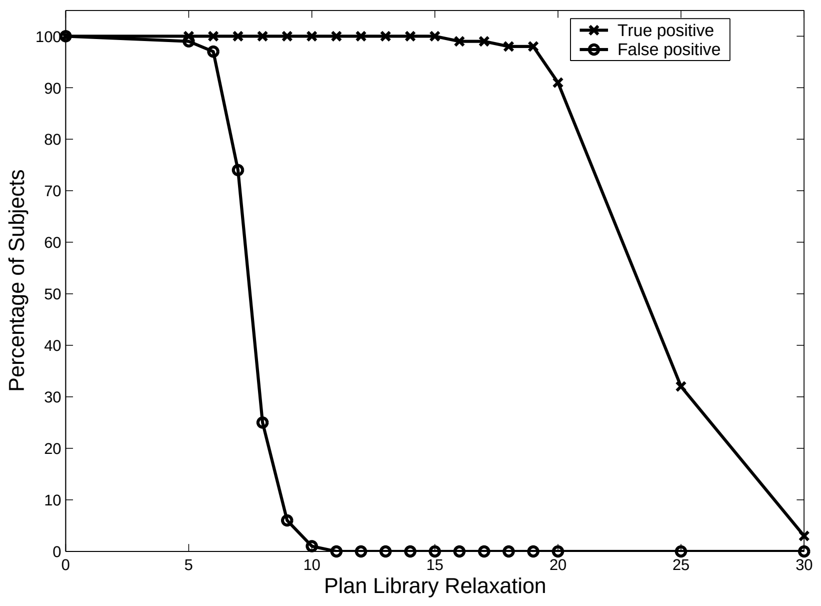

Fig. 7: True positive vs. false positives on CAVIAR data.

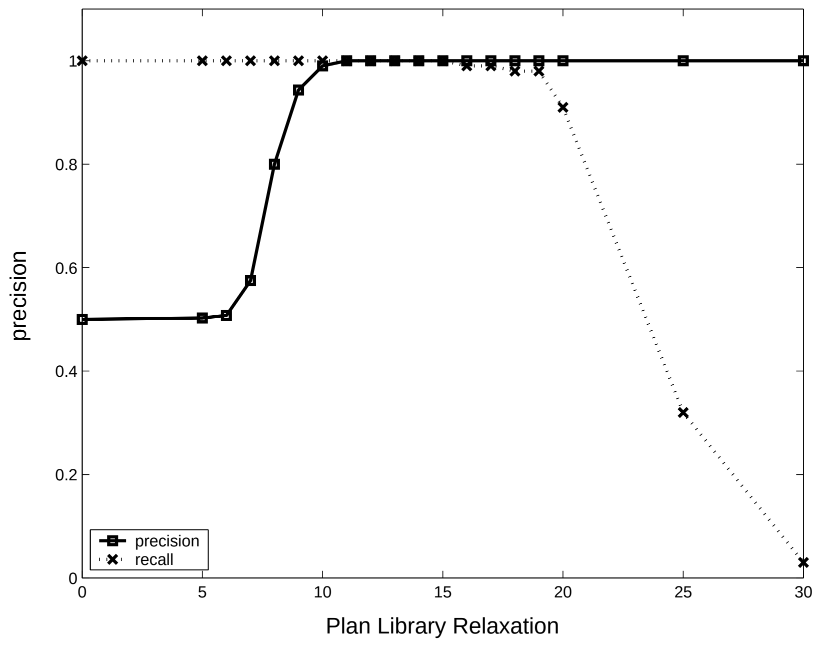

Fig. 8: Precision and recall on CAVIAR data.

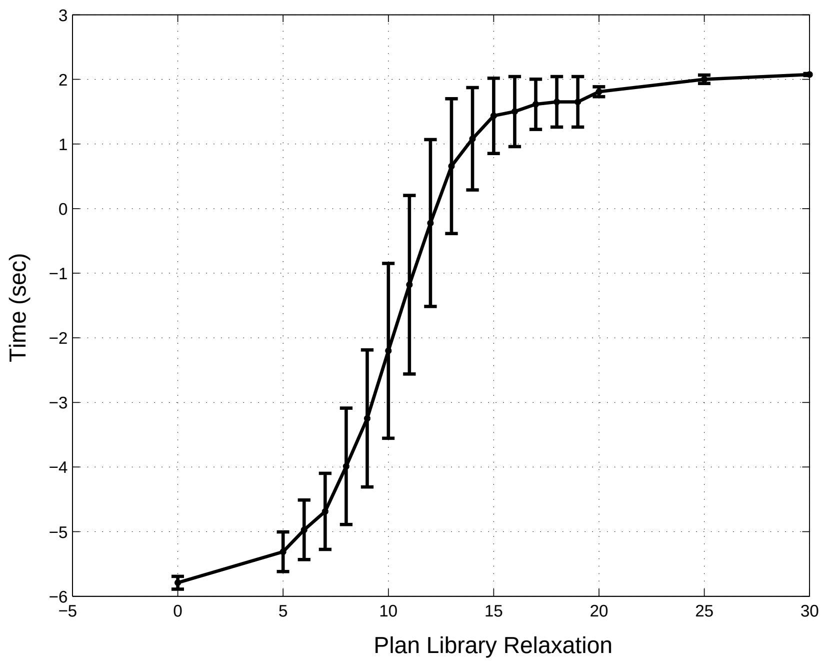

Fig. 9: Time for detection suspicious path on CAVIAR data.

behavior is taking place. Two seconds is the maximum of the graph, since this is the time that the scene was over, therefore detecting at 2 seconds is too late.

Figure 10 shows the trade-off between detecting too late and detecting too early as a function of the plan library relaxation (where too early is before the split and too late is the end of the abnormal path). The x axis is the plan library relaxation, and the y axis is the percentage of subjects that were detected too early or too late. We can see that relaxation of 16 gives the best results of 1% too early and 1% too late. After relaxation of 16, the percentage of trajectories that we detect too late is slowly increases.

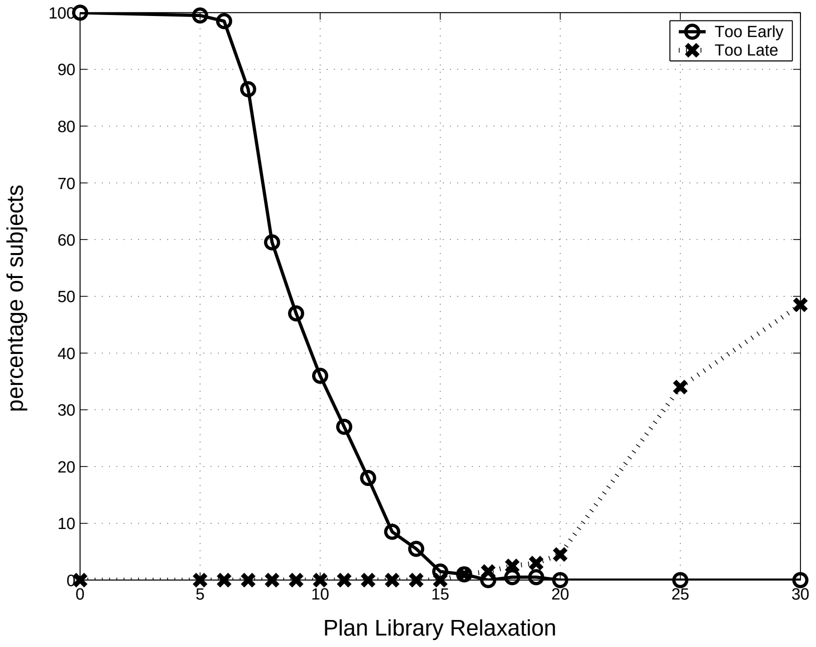

Fig. 10: Too early detection and too late detection on CAVIAR data.

The recognition algorithm also capable of detecting anomalous behavior in the direction, and not only in the position. The following experiment demonstrates this capability; here, we evaluate the use of the system in identifying the u-turn trajectories in the data set. We sampled 100 instances of the *U-Turn* trajectory with Gaussian noise and checked the time for detection. We trained a model on the same 100 noisy trajectories from *Curved path A* that we used in the first experiment.

We first examine the time to detect a u-turn as an abnormal behavior. Figure 11 shows the time for detecting abnormal behavior versus the plan library relaxation. The x axis is the plan library relaxation, and the y axis is the time (sec) passed until detecting the u-turn. We can see that until plan library relaxation of about 10, the time for detection is negative, since we detect too early before the u-turn behavior is taking place, and the standard deviation is high. The maximum detection time is 2.5 seconds after the turn is taking place. Figure 12 demonstrates the position on the trajectory 1 second after the turn, 2 second after the turn and 10 seconds after the turn.

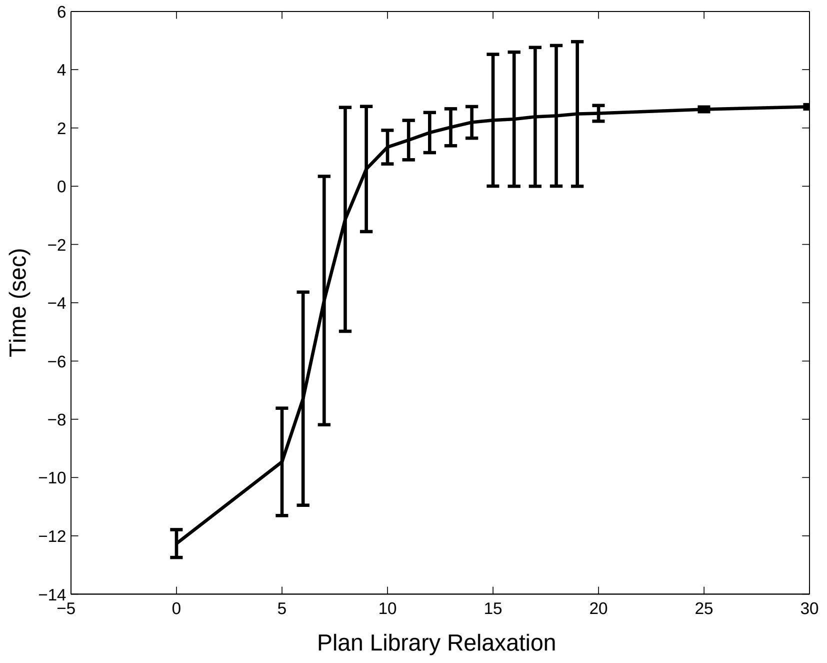

Fig. 11: Time for detecting U Turn on CAVIAR data.

Figure 13 shows the precision and recall for the u-turn experiment as function of the time. The plan library relaxation was set to 15, which is the best relaxation according to the first experiment. The precision has the perfect score of 1, for plan relaxation of 15 (every suspect that was labeled as suspect was indeed a suspect). The recall starts from score zero and gradually increases, and after about 2.5 seconds it gets the perfect score of 1 (all suspects were found 2.5 seconds after the turn).

4.1.3 AVNET Consortium Data

Parts of our work were funded through the AVNET consortium, a government-funded project including multiple industrial and academic partners, for development of suspicious activity detection capabilities. As part of this project, we were given the tracking results from a commercial vision-based tracker, developed by consortium partners.

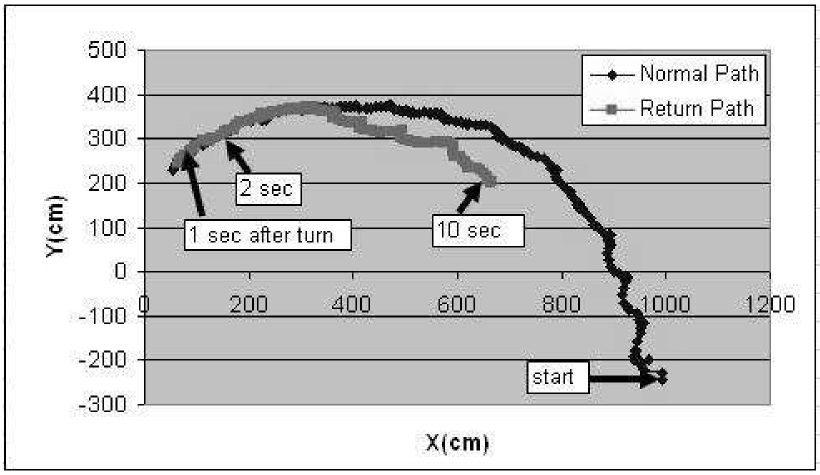

Fig. 12: U Turn on CAVIAR data.

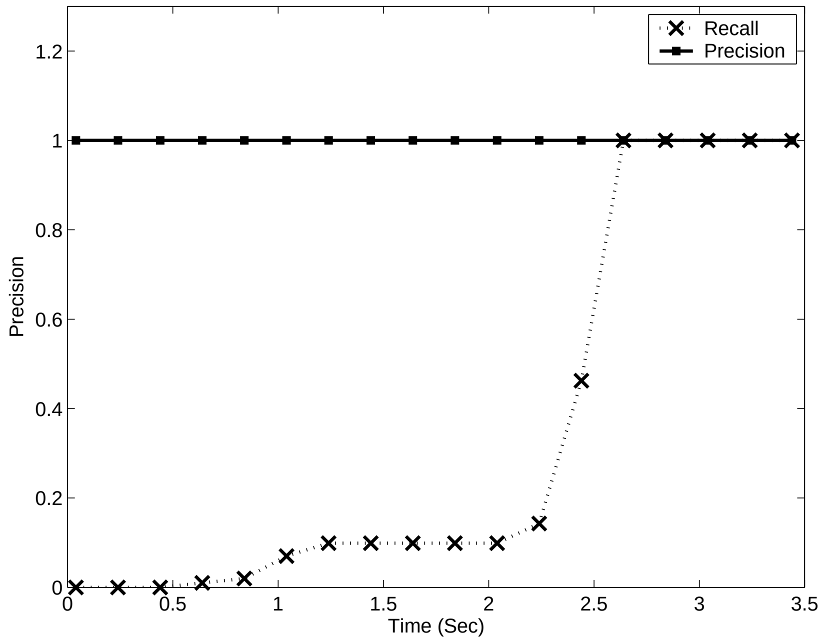

Fig. 13: Precision and recall for U Turn versus time on CAVIAR data.

We used the consortium data sets in order to evaluate our algorithm. In the first experiment, we got 164 trajectories, all with normal behavior. We ran a 10-fold cross validation test on the data set in order to test the performance. We divided the whole set to 10 data sets, each contained 148 trajectories for training, and 16 trajectories for test (except the last test that contained 20 trajectories for test and 144 for training). We learned a model with square size fixed to be the size of the maximum step in the data (31), and the *position overlap* to be 10.

We checked the number of false positives (the number of non-suspects that were mistakenly classified as suspects). Table 4.1.3 shows the results. We can see that the maximum number of false positives is 1 out of 16 (6.25%). On average (across the trials) the percentage of the false positives is 2.375%.

Test Number	Percentage of False Positives
1	0
2	0
3	0
4	0
5	6.25%
6	6.25%
7	0
8	0
9	6.25%
10	5%

Tab. 1: Percentage of false positives in AVNET data.

The next two experiments evaluate the recognition of anomalous behavior using the AVNET data set. In the first experiment, we were given a data set that consisted of 18 trajectories (432 single points). We learned from this data a model, with grid size of 31 and position overlap of 10 (as in the first experiment). We tested it against a single trajectory with a u-turn pattern. Figure 14 shows all of the 18 trajectories, the turn pattern which was found suspicious marked in bold by the recognition system. The arrows point to the start position and the turn position.

In the second experiment of evaluating anomalous behavior, we show that detecting abnormal behavior based on spatial motion is not enough. There is a need also to recognize abnormal behavior in time. For instance, we would like to recognize as abnormal someone that stays in one place an excessive amount of time, or who moves too slowly or too quickly.

We were given a data set consisting of 151 trajectories (a total of 4908 single points). In this experiment, we learned on this data a model, with grid size of 31 and position overlap of 10 (as in the first experiment). We tested it against a single trajectory of standing in place for a long duration. Figure 15 shows all of the 151 trajectories. The trajectory that was detected as anomalous by the

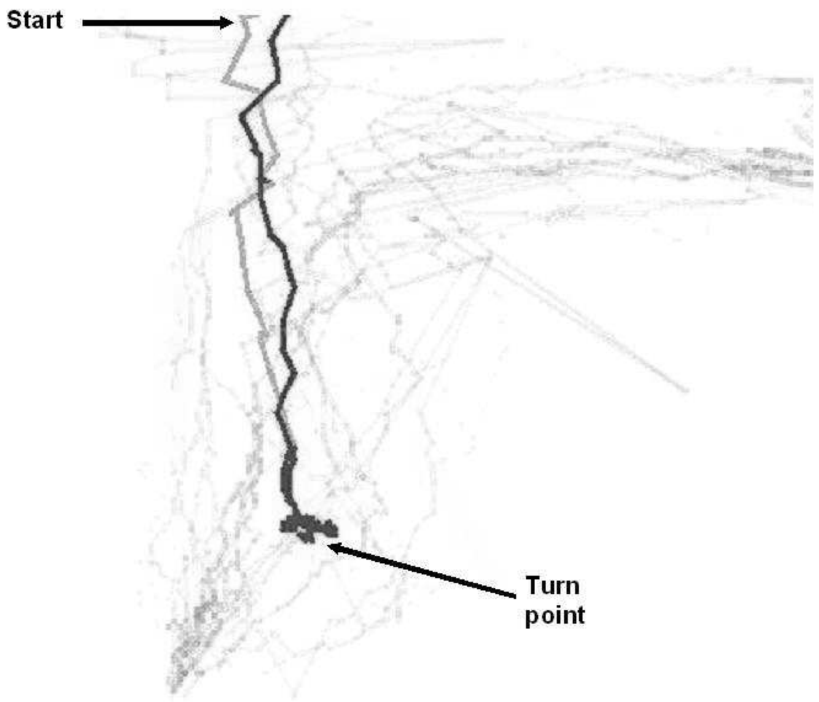

Fig. 14: Detecting U-Turn on AVNET data.

recognition system marked in bold. The arrows point at the start position and the standing position.

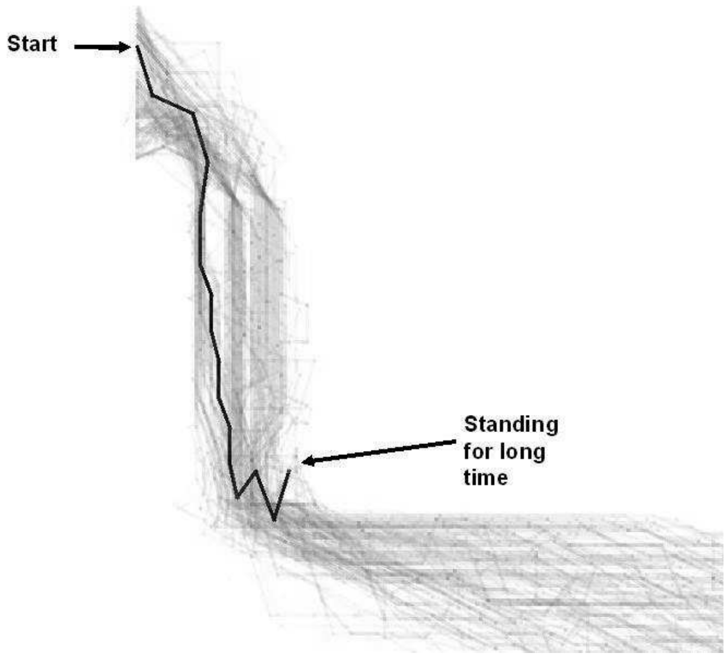

Fig. 15: Detecting standing for long time on AVNET data

4.2 Detecting Suspicious Behavior

To demonstrate the capabilities of UPR, and its efficient implementation, as described in Section 3, we tested the capabilities of our system in three different recognition tasks. The domain for the first task consisted of recognizing passengers that leave articles unattended, as in the example above. In the second task we will show how our algorithms can catch a dangerous driver that cuts between two lanes repeatedly. The last experiment intends to show how previous work, which has used costs heuristically [55], can now be recast in a principled manner. All of these examples show that we should not ignore the observer biases, since the most probable hypothesis sometimes masks hypotheses that are important for the observer.

4.2.1 Leaving Unattended Articles

It is important to track a person that leaves her articles unattended in the airport. It is difficult, if not impossible, to catch this behavior using only probabilistically-ranked hypotheses. We examine the instantaneous recognition of costly hypotheses.

We demonstrate the process using the plan library in Figure 2. This plan library is used to track simulated passengers in an airport that walk about carrying articles, which they may put down and pick up again. The recognizer's task is to recognize passengers that put something down, and then continue to walk without it. Note that the task is difficult because the plan-steps are hidden (e.g., we see a passenger bending, but cannot decide whether she picks something up, puts something down, or neither; we cannot decide whether a person has an article when they walk).

For the purposes of a short example, suppose that in time $t = 2$ (Figure 2), the SBR had returned that the two plan-steps marked *walk* match the observations (*walkN* means walking with no article, *walkW* signifies walking with an article); in time $t = 3$ the two *stop* plan steps match (*stopN* and *stopW*), and in time $t = 4$ the plan step *pickN* and plan step *putW*, match (e.g., we saw that the observed agent was bending).

The probability in $t = 4$ will be $P(putW|stopW) = 0.5 \times 0.2 = 0.1$ (the probability of *stopW* in previous time-stamp is 0.5, then following sequential link to *putW*), and in the same way $P(pickN|stopN) = 0.5 \times 0.3 = 0.15$. Normalizing the probabilities for the current time $t = 4$, $P(putW|stopW) = 0.4$ and $P(pickN|stopN) = 0.6$. The expected utility in time $t = 4$ is $U(putW|stopW) = P(putW|stopW) \times E(putW|stopW) = 0.4 \times 10 = 4$. The expected utility of *pickN* is zero. The expected costs, rather than likelihoods, raise suspicions of a passenger putting down an article (perhaps not picking it up).

Let us examine a more detailed example. We generated the following observations based on the plan library shown in Figure 2: Suppose that in time stamps $t = \{1-5\}$ the passenger walks in an airport, but we cannot tell whether she has a dangerous article in her possession. In time-stamps $t = \{6 - 7\}$ she stops, then at time $t = \{8\}$ we see her bending but can not tell whether to put down or to pick up something. In time-stamps $t = \{10 - 12\}$, she walks again.

Figure 16 shows the results from the recognition process for these observations. The X-axis measures the sequence of observations in time. The probability of different leaves (corresponding to hypotheses) is shown on the Y-axis in the upper graph. The expected costs are shown in the lower graph. In both, the top-ranking hypothesis (after each observation), is the one whose value on the Y-axis is maximal for the observation.

In the probabilistic version (upper graph), we can see that the probabilities, in time $t = \{1 - 5\}$, are 0.5 since we have two possible hypotheses of walking. with or without an article (*walkW* and *walkN*). Later when the person stops there are again two hypotheses *stopW* and *stopN*. Then, in $t = \{7\}$ two plan steps match the observations: *pickW* and *putN*, where the prior probability of *pickN* is greater than *putN* (after all, most passengers do not leave items unattended). As a result, the most likely hypothesis for the remainder of the sequence is that the passenger is currently walking with her article in hand *walkW*.

In the lower graph we can see a plot of the hypotheses, ranked by expected

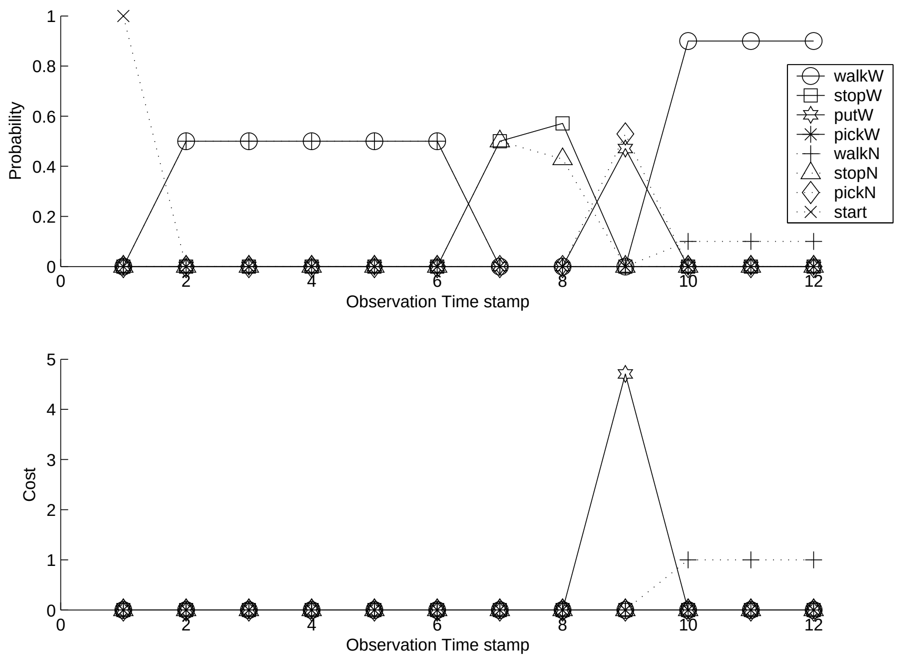

Fig. 16: Leaving unattended articles: Probabilities and Costs

cost. At time $t = 8$ when the agent picks or puts something, the cost is high (equal to 5), then in time stamp $t = \{9-12\}$ the top-ranking hypothesis is *walkN*, signifying that the passenger might have left an article unattended. Note that the prior probabilities on the behavior of the passenger have not changed. What is different here is the importance (cost) we attribute to observed actions.

4.2.2 Catching a Dangerous Driver

Some behavior becomes increasingly costly, or increasingly gainful, if repeated. For example, a driver switching a lane once or twice is not necessarily acting suspiciously. But a driver zigzagging across two lanes is dangerous. We demonstrate here the ability to accumulate costs of the most costly hypotheses, in order to capture behavior whose expected costs are prohibitive *over time*.

Figure 4.2.2 shows two lanes left and right in a continuous area, divided by a grid. There are 2 straight trajectories and one zigzag trajectory from left to right lane. From each position, the driver can begin moving to the next cell in the row (straight), or to one of the diagonal cells. We emphasize that the area and movements are continuous—the grid is only used to create a discrete state space for the plan library. Moreover, the state space is hidden: A car in the left lane may be mistakenly observed (with small probability) to be in the right lane, and vice versa.

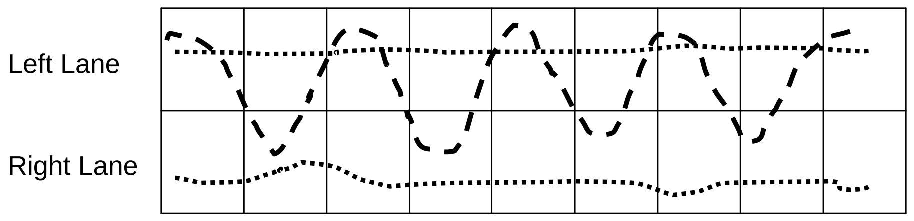

Fig. 17: Simulated trajectories for drivers.

Each grid-cell is a plan-step in the plan library. The associated probabilities and utilities are as follows: The probability for remaining in a plan-step (for all nodes) is 0.4. The probability of continuing in the same lane is 0.4. The probability of moving to either diagonal is 0.2. All costs are zero, except when moving diagonally, where the cost is 10. Observations are uncertain; with 0.1 probability, an observation would incorrectly report on the driver being in a given lane.

We generated 100 observation sequences (each of 20 observations) of a zigzagging driver, and 100 sequences of a safe driver. The observations were sampled (with noise) from the trajectories (i.e., with observation uncertainty). For each sequence of observations we accumulated the cost of the most costly hypothesis, along the 20 observations. We now have 100 samples of the accumulated costs for a dangerous driver, and 100 samples of the costs for a safe driver. Depending on a chosen threshold value, a safe driver may be declared dangerous (if its accumulated cost is greater than the threshold), and a dangerous driver might be declared safe (if its accumulated cost is smaller than the threshold).

Figure 18 shows the confusion error rate as a function of the threshold. The error rate measures the percentage of cases (out of 100) incorrectly identified. The figure shows that a trade-off exists in setting the threshold, in order to improve accuracy. Choosing a cost threshold at 50 will result in high accuracy, in this particular case: All dangerous drivers will be identified as dangerous, and yet 99 percent of safe drivers will be correctly identified as safe.

4.2.3 Air-Combat Environment

Tambe and Rosenbloom [55] used an example of agents in a simulated air-combat environment to demonstrate the RESC plan recognition algorithm. RESC heuristically prefers a single worst-case hypothesis, since an opponent is likely to engage in the most harmful maneuver in an hostile environment. Tambe and Rosenbloom [55] showed this heuristic in action in a simulated air-combat, where the turning actions of the opponent could be interpreted as either leading to it running away, or to its shooting a missile. RESC prefers the hypothesis that the opponent is shooting. However, unlike UPR, RESC will *always* prefer this hypothesis, regardless of its likelihood, and this has proven problematic [55].

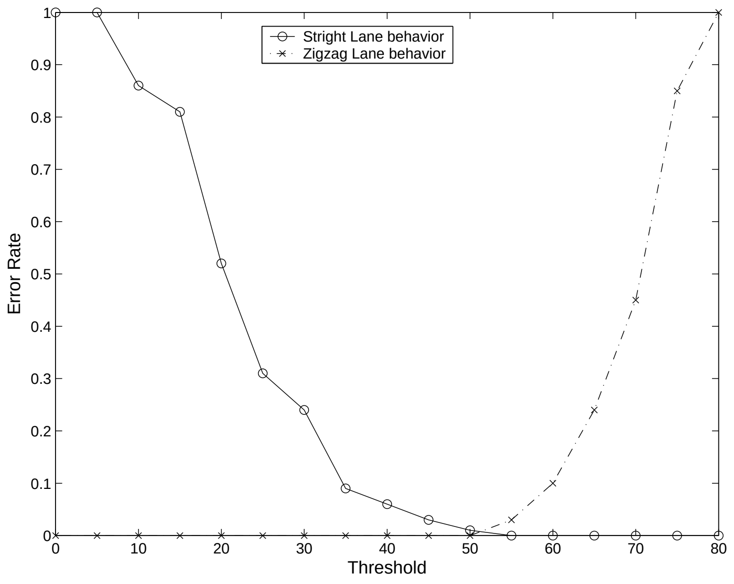

Fig. 18: Confusion error rates for different thresholds for dangerous and safe drivers.

Moreover, given several worst-case hypotheses, RESC will arbitrarily choose a single hypothesis to commit to, again regardless of its likelihood. Additional heuristics were therefore devised to control RESC's worst-case strategy [55].

We generalize this example to show UPR subsumes RESC's heuristic in a principled manner. Suppose instead of shooting a missile (which has infinite cost) vs. running away, we consider hypotheses of *invading air-space* vs. *run-away*, where invading the observer's air-space is costly for it, but not fatal. Figure 19 shows models of two types of opponents: An aggressive opponent (left sub-figure) that is more likely to shoot (0.8 a-priori probability) than to run away (0.2), and a cowardly opponent (right sub-figure) that is more likely to run away. Note that these models are structurally the same; the assigned probabilities reflect the a-priori preferences of the different opponent types. Thus an observation matching both hypotheses will simply lead to both of them being possible, with different likelihoods. The maximum posterior hypothesis in the aggressive case will be that the opponent is trying to invade our airspace. In the cowardly case, it would be that the opponent is running away. RESC's heuristic would lead it to always selecting the aggressive case, regardless of the likelihood.

In contrast, UPR incorporates the biases of an *observing* pilot much more cleanly. Because it takes the likelihood of hypotheses into account in computing the expected cost, it can ignore sufficiently improbable (but still possible) worst-case hypotheses, in a principled manner. Moreover, UPR also allows modeling optimistic observers, who prefer best-case hypotheses.

Table 2 presents three cost models. In the first case, the *runaway* plan-step will get zero cost, and *invade* a high cost (10). This is an observer who is

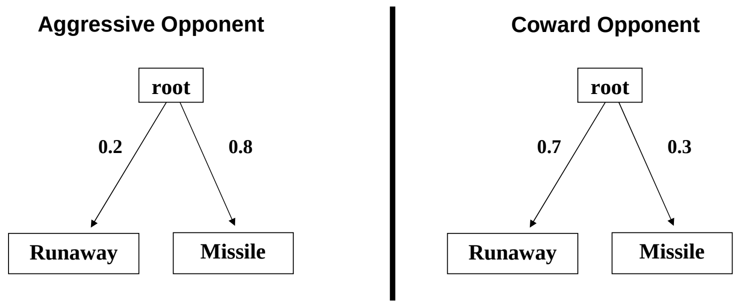

Fig. 19: Air-Combat Environment. Two types of opponents.

	Runaway	Missile
Case A	0	10
Case B	-10	10
Case C	10	10

Tab. 2: Three cases of utilities for Figure 19.

worried that its airspace being invaded, but not gaining anything from scaring the opponent away. In the second case the *runaway* plan-step will get negative cost (i.e., a gain for the observer). In the third case there are the same costs. Tables 3 and 4 show the recognition results. The first row shows the results of following only the probabilistic reasoning in each model. The next three rows show the hypothesis costs for each hypothesis, in each of the three cases in Table 2.

In the cases of the aggressive opponent, both the most costly or the most probable hypothesis is the *invade* hypothesis. However, in the cowardly opponent case, the answer depends on the utility model. In cases A and B, where we gave high cost for missile, the most probable hypothesis stays runaway but the costly hypothesis is missile. In the third case, C, since we gave neutral costs (same for the two plan-steps), we got a result as in the probability model, meaning *runaway*. The conclusion is that the probabilistic model is not enough in case we want to incorporate biases of the observer, in this case that the missile plan-step is harmful for the observer.

This generalization of the original example in [55] demonstrates that the heuristic worst-case preference of RESC is subsumed by the principled use of decision-theoretic reasoning in our algorithms. And the complexity analysis in earlier sections shows that such reasoning does not necessarily entail significant computational costs. RESC's run-time complexity is linear in the size of plan library. UPR's is polynomial.

	Runaway	Missile
Probabilistic	0.2	**0.8**
Cost A	0	**8**
Cost B	-2	**8**
Cost C	2	**8**

Tab. 3: Aggressive opponent: the result utilities for Figure 19.

	Runaway	Missile
Probabilistic	0.3	**0.7**
Cost A	0	**3**
Cost B	-7	**3**
Cost C	7	**3**

Tab. 4: Cowardly opponent: the result utilities for figure 19.

5 Future Directions and Final Remarks

In this chapter we concentrated on efficient hybrid adversarial plan recognition algorithms (Section 3) and their contribution to the domain of detecting anomalous and suspicious behavior (section 4). The two main contributions of this chapter are as follows:

- First, we presented an anomalous behavior recognition model using the Symbolic Behavior Recognizer (SBR) that is capable of answering the plan library membership query highly efficiently [3, 5]. Here, the plan library represents normal behavior; any activity which does not match the plan library is flagged as anomalous. We demonstrated its use in two sets of experiments with trajectories created by machine vision systems.

- Second, we present a suspicious behavior recognition model using Utility based Plan Recognition (UPR) [2, 4]. This is a general model of plan-recognition that allows the observer to incorporate her own biases and preferences—in the form of a utility function—into the plan recognition process. Here, the plan library explicitly represents suspicious behavior; any activity that matches the model will be assumed to be suspicious. We demonstrate the capabilities of the suspicious behavior recognition model in three different domains, and also its principled generalization of previous (heuristic) adversarial plan recognition work [55].

The hybrid system presented is an instantiation of a more general approach to keyhole adversarial plan recognition, combining two approaches: one detecting anomalous behavior, and the other detecting suspicious behavior. This approach leaves open many challenges, including:

- Formally define multi-agent plan recognition queries in adversarial settings, and explore methods for anomalous and suspicious plan recognition in multi-agent settings.
- Following up on [52], continue to explore methods for UPR for suspicious plan recognition, which maintain computational tractability, but with greater expressiveness.
- Most plan recognition literature—and this chapter is no different—ignore *intended recognition* (where the adversary knows it is being observed, and acts accordingly; see [38] for a recent exception). In adversarial settings, assuming the adversary knows that it is being observed can cause a computational blow up, because even when an another agent appears to be not suspicious, the system must necessarily assume it is only acting as such, because it is being observed. Thus pruning hypotheses is necessarily difficult.

Acknowledgments. We thank the anonymous reviewers and this collection's editors for their very valuable and detailed comments on early drafts of this work. This research was carried out with partial support by the Israeli Ministry of Commerce AVNET Consortium, and the Israel Science Foundation Grant #1511/12. As always, thanks to K. Ushi.

References

[1] J. A. Adams. *Human Management of a Hierarchical System for the Control of Multiple Mobile Robots.* PhD thesis, University of Pennsylvania, 1995.

[2] D. Avrahami-Zilberbrand. *Efficient Hybrid Algorithms for Plan Recognition and Detection of Suspicious and Anomalous Behavior.* PhD thesis, Bar Ilan University, 2009.

[3] D. Avrahami-Zilberbrand and G. A. Kaminka. Fast and complete symbolic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 653–658, 2005.

[4] D. Avrahami-Zilberbrand and G. A. Kaminka. Incorporating observer biases in keyhole plan recognition (efficiently!). In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*, pages 944–949, 2007.

[5] D. Avrahami-Zilberbrand, G. A. Kaminka, and H. Zarosim. Fast and complete plan recognition: Allowing for duration, interleaved execution, and lossy observations. In *Proceedings of the IJCAI Workshop on Modeling Others from Observations (MOO-05)*, 2005.

[6] N. Blaylock and J. Allen. Fast hierarchical goal schema recognition. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, pages 796–801, 2006.

[7] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proc. Fourteenth Annual Conference on Uncertainty in AI (UAI)*, pages 33–42, 1998.

[8] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 994, Washington, DC, USA, 1997. IEEE Computer Society.

[9] H. Bui. A general model for online probabilistic plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, 2003.

[10] H. Bui, S. Venkatesh, and G. West. Policy recognition in the abstract hidden markov models. *Journal of Artificial Intelligence Research*, 17:451–499, 2002.

[11] H. H. Bui, D. Phung, S. Venkatesh, and H. Phan. The hidden permutation model and location-based activity recognition. In *Proceedings of Twenty-Third National Conference on Artificial Intelligence (AAAI-08)*, 2008.

[12] EC funded CAVIAR project/IST 2001 37540. Found at URL: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.

[13] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks (research note). *Artificial Intelligence*, 42(2-3):393–405, 1990.

[14] F. Cupillard, Alberto.Avanzi, F. Bremond, and M. Thonnat. Video understanding for Metro surveillance. In *The IEEE conference on networking, sensing, and control (ICNSC), special session on Intelligent Transportation Systems*, pages 186–191, 2004.

[15] R. J. Doyle. Determining the loci of anomalies using minimal causal models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1821–1827, Montreal, Quebec, Canada, 1995.

[16] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-Markov models. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR-2005)*, pages 838–845, San Diego, CA, June 2005.

[17] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[18] C. Geib. Delaying commitment in plan recognition using combinatory categorial grammars. In *The International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1702–1707, 2009.

[19] C. Geib and R. Goldman. Recognizing plans with loops represented in a lexicalized grammar. In *The twenty fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 958–963, 2011.

[20] C. Geib and M. Steedman. On natural language processing and plan recognition. In *The International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1612–1617, 2007.

[21] C. W. Geib. Assessing the complexity of plan recognition. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 507–512, 2004.

[22] C. W. Geib and R. P. Goldman. Plan recognition in intrusion detection systems. In *Proceedings of the second DARPA Information Survivability Conference and Exposition (DISCEX II)*, pages 329–342, June 2001.

[23] C. W. Geib and S. A. Harp. Empirical analysis of a probabilistic task tracking algorithm. In *AAMAS workshop on Modeling Other agents from Observations (MOO-04)*, 2004.

[24] Z. Ghahramani and M. I. Jordan. Factorial hidden Markov models. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8, pages 472–478. MIT Press, 1995.

[25] B. Hilary and G. Shaogang. Advanced visual surveillance using bayesian networks. In *International Conference on Computer Vision*, pages 111–123, June 1995.

[26] S. Hongeng, F. Brémond, and R. Nevatia. Bayesian framework for video surveillance application. In *15th International Conference on Pattern Recognition (ICPR'00)*, volume 1, pages 164–170, 2000.

[27] D. H. Hu and Q. Yang. CIGAR: Concurrent and interleaving goal and activity recognition. In *Proceedings of Twenty-Third National Conference on Artificial Intelligence (AAAI-08)*, pages 1363–1368, 2008.

[28] P. Jarvis, T. Lunt, and K. Myers. Identifying terrorist activity with ai plan recognition technology. In *The Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI 04)*, pages 858–863, 2004.

[29] G. A. Kaminka. Detecting disagreements in large-scale multi-agent teams. *Journal of Autonomous Agents and Multi-Agent Systems*, 18(3):501–525, 2009.

[30] G. A. Kaminka, E. Merdler, and D. Avrahami. Advanced unsupervised spatial learning algorithm for the avnet37 consortium: Final report (in hebrew). Technical Report MAVERICK 2006/01, Bar Ilan University, Computer Science Department, MAVERICK Group, 2006.

[31] G. A. Kaminka, E. Merdler, and D. Avrahami. Advanced unsupervised spatial learning algorithm for the avnet37 consortium: Interim report (in hebrew). Technical Report MAVERICK 2006/01, Bar Ilan University, Computer Science Department, MAVERICK Group, 2006.

[32] G. A. Kaminka, D. V. Pynadath, and M. Tambe. Monitoring teams by overhearing: A multi-agent plan recognition approach. *Journal of Artificial Intelligence Research*, 17:83–135, 2002.

[33] G. A. Kaminka and M. Tambe. Robust multi-agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research*, 12:105–147, 2000.

[34] E. Khalastchi, M. Kalech, G. A. Kaminka, and R. Lin. Online anomaly detection in unmanned vehicles. In *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-11)*, pages 115–122, 2011.

[35] U. Kjærulff. A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-1992)*, pages 121–129, San Mateo, CA, 1992. Morgan Kaufmann.

[36] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3):295–331, Aug 1999.

[37] L. Liao, D. Fox, and H. A. Kautz. Learning and inferring transportation routines. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 348–353, 2004.

[38] V. Lisy, R. Pibil, J. Stiborek, B. Bosansky, and M. Pechoucek. Game-theoretic approach to adversarial plan recognition. In L. D. Raedt, C. Bessiere, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. Lucas, editors, *20th European Conference on Artificial Intelligence (ECAI)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 546–551. IOS Press, 2012.

[39] D. Mahajan, N. Kwatra, S. Jain, P. Kalra, and S. Banerjee. A framework for activity recognition and detection of unusual activities. In *ICVGIP*, pages 15–21, 2004.

[40] W. Mao and J. Gratch. Decision-theoretic approaches to plan recognition. In *USC/ICT Technical Report*, 2004.

[41] W. Mao and J. Gratch. A utility-based approach to intention recognition. In *Proceedings of the AAMAS Workshop on Modeling Other Agents from Observations (MOO-04)*, NY City, NY, USA, July 2004.

[42] E. Marhasev, M. Hadad, G. A. Kaminka, and U. Feintuch. The use of hidden semi-markov models in clinical diagnosis maze tasks. *Intelligent Data Analysis*, 13(6):943–967, 2009.

[43] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 955–960, 2005.

[44] W. Niu, J. Long, D. Han, and Y.-F. Wang. Human activity detection and recognition for video surveillance. In *Proceedings of the IEEE Multimedia and Expo Conference*, pages 719–722, 2004.

[45] P. E. of a Vision Based Lane Tracker Designed for Driver Assistance Systems. Ajoel c. mccall and mohan m. trivedi. *IEEE Intellient Vehicles Symposium*, pages 153–158, 2005.

[46] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. *In Fourth IEEE International Conference on Multimodal Interfaces*, 8:831–843, 2002.

[47] D. V. Pynadath and S. Marsella. Psychsim: Modeling theory of mind with decision-theoretic agents. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1181–1186, 2005.

[48] D. V. Pynadath and S. Marsella. Minimal mental models. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*, pages 1038–1044, 2007.

[49] D. V. Pynadath and M. P. Wellman. Generalized queries on probabilistic context-free grammars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):65–77, 1998.

[50] D. V. Pynadath and M. P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence*, pages 507–514, 2000.

[51] T. Starner and A. Pentland. Real-time American Sign Language recognition from video using hidden Markov models. In *In Proceedings of the International Symposium on Computer Vision (SCV-95)*, pages 265–270, 1995.

[52] B. stjan Kalu ˇ za, G. A. Kaminka, and M. Tambe. Detection of suspicious behavior from a sparse set of multiagent interactions. In *Proceedings of the Eleventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-12)*, 2012.

[53] G. Sukthankar and K. Sycara. A cost minimization approach to human behavior recognition. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-05)*, pages 1067–1074, 2005.

[54] R. Suzic. A generic model of tactical plan recognition for threat assesment. In B. V. Dasarathy, editor, *Proceedings of SPIE Multisensor*, volume 5813, pages 105–116, March 2005.

[55] M. Tambe and P. S. Rosenbloom. RESC: An approach to agent tracking in a real-time, dynamic environment. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 103–111, August 1995.

[56] D. V. Vail, M. M. Veloso, and J. D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-07)*, pages 1331–1338, 2007.

[57] Wikipedia entry: Precision and recall, 2009. Found at http://en.wikipedia.org/w/index.php?title=Precision_and_recall [Online; accessed 23-March-2009].

[58] G. Wu, Y. Wu, L. Jiao, Y.-F. Wang, and E. Y. Chang. Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 528–538, New York, NY, USA, 2003. ACM Press.

[59] T. Xiang and S. Gong. Video behavior profiling for anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):893–908, 2008.

[60] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *In Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR-92)*, pages 379–385, 1992.

[61] J. Yin, Q. Yang, and J. J. Pan. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1082–1090, 2008.