Semantic Video Content Analysis

Massimiliano Albanese, Pavan Turaga, Rama Chellappa,
Andrea Pugliese, and V.S. Subrahmanian

Abstract. In recent years, there has been significant interest in the area of automatically recognizing activities occurring in a camera's field of view and detecting abnormalities. The practical applications of such a system could include airport tarmac monitoring, or monitoring of activities in secure installations, to name a few. The difficulty of the problem is compounded by several factors: detection of primitive actions in spite of changes in illumination, occlusions and noise; complex multi-agent interaction; mapping of higher-level activities to lower-level primitive actions; variations in which the same semantic activity can be performed. In this chapter, we develop a theory of semantic activity analysis that addresses each of these issues in an integrated manner. Specifically, we discuss ontological representations of knowledge of a domain, integration of domain knowledge and statistical models for achieving semantic mappings, definition of logical languages to describe activities, and design of frameworks which integrate all the above aspects in a coherent way, thus laying the foundations of effective Semantic Video Content Analysis systems.

1 Introduction

Interaction amongst humans and with objects forms the basis of almost all human activities. Semantics refers to the meaning associated with a particular interaction. While the set of all possible interactions can be quite large, many activities are constrained in nature. These constraints are induced in part by the surrounding environment under consideration, such as an airport, and in part by the underlying motive of the interaction, such as a hand-shake. The visual analysis of activities performed

Massimiliano Albanese, Pavan Turaga, Rama Chellappa, V.S. Subrahmanian
Institute for Advanced Computer Studies,University of Maryland, College Park, MD 20742
e-mail: albanese,pturaga,rama,vs@umiacs.umd.edu

Andrea Pugliese
DEIS, University of Calabria, Rende, Italy
e-mail: apugliese@deis.unical.it

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 147–176.
springerlink.com

by a group of humans in such settings is of great importance in applications such as automated security and surveillance systems. In real life, one usually has some prior knowledge of the semantic structures of activities that occur in a given setting. Taking into account this knowledge, we need to understand how to represent the constraints induced both by the surrounding environment and by the motive of the particular activity. The representation and analysis of these constraints and the mapping to semantically meaningful concepts form the subject of this chapter.

There has been significant interest in the area of automatically recognizing activities occurring in a camera's field of view and detecting abnormalities. Several factors contribute to the complexity of the problem: a) unambiguous detection of low-level primitive actions in spite of changes in illumination, occlusions and noise; b) high-level representation of activities in settings which are usually characterized by complex multi-agent interaction; c) mapping higher-level concepts to lower-level primitive actions; d) understanding the variations in which the same semantic activity can be performed and achieving robustness to such variations during recognition.

In this chapter, we develop a theory of semantic activity analysis that addresses each of these issues in an integrated manner. Specifically, Section 2 summarizes the major challenges and issues in the field of semantic activity analysis, and proposes a general framework which integrates all these aspects in a coherent way. Section 3 discusses ontological representations of domain knowledge, while Section 4 proposes two different solutions – based on Finite State Automata and Petri Nets respectively – for integrating domain knowledge and statistical models, and achieving semantic mappings. Section 5 explores logical languages for describing activities. Finally, experimental evaluations of the proposed techniques are reported in Section 6 and concluding remarks are given in Section 7.

2 Human Interactions: Challenges and Issues

Designing a system for semantic analysis of human actions from video requires one to adopt a principled approach starting from defining an activity. This requires specifying the domain of interest and the types of activities of interest. Once we enumerate and define what we mean by activities, we then need to represent them by appropriate computational models. This forms the basis for any automatic approach to recognizing human activities. However, video data brings with it unique challenges that require specific attention to enable deploying semantic models for real applications. Many of the key challenges are a consequence of the nature of the imaging process. Cameras generate mappings from the three-dimensional world to a two-dimensional image plane. This necessarily introduces several ambiguities which make reasoning difficult and low-level processing prone to errors. Therefore, video analysis requires methods that are not only semantically rich, but also robust to errors in the low-level processing. Furthermore, achieving all of these goals in a real-time setting requires designing computationally efficient algorithms.

Another major issue in automatic activity detection is the semantic gap between the types of primitive actions that can be automatically recognized using state of

the art image processing algorithms (e.g., a person walking on a street, the presence of a package-like object in the scene) and the type of complex interactions that automatic surveillance systems are intended to detect. Therefore, expressive and flexible models are needed to describe complex interactions in terms of primitive actions. The challenge here is to build such models in a way that is general enough to be applied to different domains and robust to noise and variations from standard scenarios. Thus, video mining requires addressing the following core issues:

1. **Representation:** Defining the meaning of an activity.
2. **Model:** Defining computational models for an activity.
3. **Robustness:** Designing algorithms robust to ambiguities in the imaging process.
4. **Efficiency:** Designing efficient recognition algorithms.

In this chapter, we discuss each of these problems and provide the reader with a deeper understanding of the underlying challenges. We demonstrate the effectiveness of the proposed design philosophy using experiments on real video sequences.

2.1 Unified Framework for Activity Semantics

Although a huge amount of work has been done on many of the individual issues discussed in the previous section, considerably less effort has been put towards the definition of a framework where all the aspects of Semantic Video Content Analysis are integrated in a coherent and effective way. Broadly speaking, there are three main classes of problems – from a user's perspective – that such an integrated system should address:

- **Evidence.** Given a video v, a set of activity definitions $\mathscr{A}$, a time interval (t_s, t_e), and a probability threshold p_t, find all the minimal subsequences of v containing occurrences X of activities in $\mathscr{A}$ such that X occurs within the interval (t_s, t_e) with probability $p \geq p_t$.
- **Identification.** Given a video v, a set of activity definitions $\mathscr{A}$, a time interval (t_s, t_e), find the activity which occurs in v within the interval (t_s, t_e) with maximal probability among the activities in $\mathscr{A}$.
- **Online Identification.** Given a real-time video feed $v = \langle f_1, f_2, \ldots, f_t \rangle$, where f_t is the current frame, and a set of activity definitions $\mathscr{A}$, find the probability that each activity in $\mathscr{A}$ is unfolding at current time t.

In this chapter, we put particular emphasis on discussing how all the components of a Semantic Video Content Analysis system can be integrated and propose a framework and design methodology that takes into account these requirements and the issues outlined in the previous section. We show in real experiments the effectiveness of the proposed design philosophy.

Figure 1 shows the general architecture of the proposed framework. The front-end of the system consists of the usual physical elements, such as cameras and video storage units. At the core, there is an ontological repository of activity and domain definitions, possibly provided by an expert. The Ontology consists of the 'vocabulary' and the 'grammar' of human activities. To bridge the gap between the

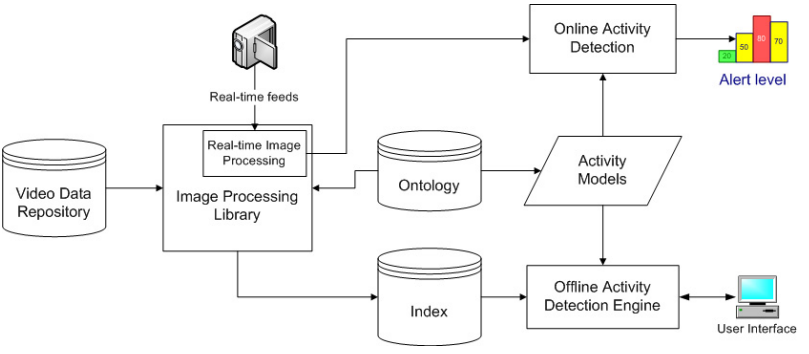


Fig. 1 General architecture of a Semantic Video Content Analysis system

low-level features at the front-end and the activity-definitions at the core, we require image and video analysis methods that can provide detection and recognition of the 'vocabulary'. Computational algorithms for activity recognition draw upon these, and identify activities of interest defined according to the 'grammar'.

3 Knowledge Representation

With the proliferation of visual surveillance systems in a wide variety of domains such as banks, airports, and convenience stores, it is necessary to create a general representation framework for modeling activities. Examples of such knowledge-bases, or more formally, ontologies have been in existence in other fields of AI such as the semantic web, image and video annotation and computational genomics. Designing ontologies for human activities has only recently gained interest in the computer vision community. The advantages of such a centralized representation are easily seen. It standardizes activity definitions, allows for easy portability to specific deployments, enables interoperability of different systems and allows easy replication and comparison of system performance. Since video-mining approaches rely on a domain expert to provide activity semantics, it is useful to create a standardized ontology from which to draw upon. Recent efforts have focused at creating ontologies for activities in specific scenarios. Examples include analysis of social interactions in nursing homes [6], classification of meeting videos [13], and activity detection in a bank monitoring setting [10]. As a result of the Video Event Ontology Challenge Workshop held in 2003 [12], ontologies have been defined for six video surveillance domains: 1) Perimeter and Internal Security, 2) Railroad Crossing Surveillance, 3) Visual Bank Monitoring, 4) Visual Metro Monitoring, 5) Store Security, 6) Airport-Tarmac Security. The workshop led to the development of two formal languages – Video Event Representation Language (VERL) [15], which provides an ontological representation of complex events in terms of simpler sub-events, and Video Event Markup Language (VEML), which is used to annotate VERL events in videos.

In most practical deployments, activity definitions are constructed in an empirical or ad-hoc manner. Though empirical constructs are fast to design and even work

```
PROCESS(cruise-parking-lot(vehicle v, parking-lot lot),
    Sequence(enter(v, lot), Repeat(AND(move-in-circuit(v), inside(v, lot))),
        exit(v, lot)))
```

Fig. 2 Cruise Parking Lot ontology

very well in most cases, they are limited in their utility to the specific deployment for which they have been designed. A principled approach must then be adopted. An ontology for human activities describes entities, environment, interaction among them and the sequence of events that is semantically identified with an activity. It specifies how an activity can be composed using lower-level primitive events and identifying the role played by each entity in the sequence of events. Building activity ontologies for a domain may be broadly classified into the following steps:

Entities: Define animate and inanimate entities relevant to an activity. The list could include humans, vehicles and environment (e.g. scene location, buildings).

Identity: Specify the properties – such as physical attributes and appearance – that uniquely identify each entity.

Spatio-temporal Attributes: Specify spatio-temporal attributes representing the state of an entity. Examples include definitions of 'near', 'far', 'before', 'after'.

Activity Primitives: Describe primitive events involving each entity individually and in relation to others.

Description of Activities: Define the relationship between entities and the spatio-temporal sequence of change and interaction among the entities.

Human activities are characterized by complex spatio-temporal interactions. Consider the activity of a vehicle cruising in a parking lot whose definition is given in Figure 2. This definition illustrates the importance of encoding spatio-temporal constraints. Without regard to such constraints, we might say that it is composed of the primitives: 'enter', 'move-in-circuit' and 'exit'. But, a normal car-parking activity can also be described using the same set of primitives. The difference that sets apart the two activities lies in the temporal span of the 'move-in-circuit' primitive.

3.1 Designing a Good Ontology

Thomas Gruber [11] made one of the earliest systematic attempts to propose guidelines for the design of ontologies intended for knowledge sharing and interoperability. Five important criteria for ontology design – clarity, coherence, extendibility, minimal encoding bias and minimal ontological commitment – were proposed.

Clarity: An ontology should convey the meaning of all conceptualizations unambiguously.

Coherence: An ontology should be coherent and allow for meaningful inferences to be drawn that are consistent with definitions and axioms.

Extendibility: The design of the ontology should allow future extensions without the need for revising definitions.

```
SINGLE-THREAD(suspicious-load(vehicle v, person p, ent obj, facility fac),
    AND(zone(loading-area), near(loading-area, facility), portable(obj),
        Sequence(approach(v, fac), AND(stop(v), near (v, fac), NOT(inside(v, loading-area))),
                AND(approach(p, v), carry(p, obj)), AND(stop(p), near (p, v)),
                cause(p, open(portal-of(v))), enter (obj, v),
                cause(p, close(portal-of(v))), leave(v, facility))))
```

Fig. 3 Suspicious Load in Perimeter Security

Minimal Encoding Bias: An ontology should have symbol-independent conceptualizations.

Minimal Ontological Commitment: An ontology should make as few assumptions as possible about the domain being modeled.

Additional issues that are relevant from a computer vision perspective are:

View Invariance: An ontology should not be tuned to one particular camera view, but should generalize across views.

Invariance to Rate of Activity Execution: The same activity performed by different agents may have different temporal spans. The ontology should not make any assumptions about the activity execution rate.

Invariance to Sensor Characteristics: An ontology should be insensitive to variations of sensor characteristics such as resolution, frame-rate, etc.

We now show how to resolve ambiguities according to the design principles outlined above. Let us consider the 'suspicious load' example from the Perimeter Security ontology given in Figure 3. Though this example maintains clarity, further inspection shows that minimal ontological commitment is not preserved. According to the definition the vehicle's portal has to be opened in order to load the object. This does not encompass other possible scenarios. For example, the suspicious load can be placed onto the trailer of a truck which is open from the top, hence not using any portal, or it can even be an explosive that is placed under the body of the vehicle. Moreover, it is not necessary for the vehicle to stop. For instance, somebody inside the vehicle could grab a bag from a suspicious pedestrian through the window. Hence, minimal ontology should only include the object being on the vehicle's exterior, and then being transferred to the vehicle's interior.

As this example illustrates, Gruber's criteria provide a principled method to design and refine ontologies to make them generalizable to new deployments. We refer the reader to [1] for several more detailed examples illustrating the use of Gruber's criteria in designing good ontologies for video surveillance.

4 Computational Models: Integrating Structure and Statistics

Defining semantic equivalence between two sets of interactions amounts to defining equivalence classes on the space of all interactions. The space of interactions can be quite large and doing this individually is not a feasible solution. In addition to a priori knowledge, one may also have access to a limited training set, which in most

cases is not exhaustive. Thus, we can leverage available domain knowledge to design activity models and define semantic equivalence using statistical approaches. The fusion of these disparate approaches – statistical, structural and knowledge based – has not yet been fully realized and has gained attention only in recent years. The two approaches described in this section fall in this category – we exploit domain knowledge to create rich and expressive models for activities and augment them with probabilistic extensions.

For this discussion, we present two computational models – Finite State Automata (FSA) and Petri Nets (PN). At this stage, the choice of the model is driven mostly by the complexity of the activities that need to be represented. If activities of interest are mostly sequential in nature, with only one or two agents involved in the activity, then a FSA may suffice. On the other hand, if activities of interest involve multiple agents performing different actions in parallel and occasionally interacting with each other, then Petri Nets may be the model of choice, since it allows modeling more complex behavior such as parallelism and synchrony. Once we choose a model for a particular domain, it is fundamental to equip this model with the ability to handle (a) 'noise' in the labeling of video data, (b) inaccuracies in vision algorithms and (c) variations from a hard-coded activity model. We illustrate how this can be achieved using FSA and PN as examples. However, it is important to note that the same design principles can be extended to other activity models as well.

In the following, we will often refer to the output of the image processing library with the term 'observations', or 'observation table', implicitly assuming that video data was processed offline and the output stored in a database, which might have been eventually indexed (see Section 4.1.2).

4.1 Probabilistic Automata

There is a large body of work in the AI community on plan and activity recognition, a large portion of which relies on Hidden Markov Models (HMMs) and their variants. Luhr at al. [16] use Hierarchical HMMs to learn the probabilistic nature of simple sequences of activities. Duong at al. [9] introduce the Switching Hidden Semi-Markov Model (S-HSMM), a two-layered extension of the Hidden Semi-Markov Model (HSMM). The bottom layer represents atomic activities using HSMMs, while the top layer represents a sequence of high-level activities, defined in terms of atomic activities. [17] uses non-stationary HSMMs to model the dependency of transition probabilities on the duration an agent has spent in a given state. Dynamic Bayesian networks have also been used for tracking and recognizing multi-agent activities [14]. The CFG-based representation of human activities and interactions [20] enables to formally define complex activities based on simple actions. The problem of recognizing multiple interleaved activities has been studied in [5], where the authors propose a symbolic plan recognition approach, relying on a hierarchical plan-based representation.

Fig. 4 Frames from the Bank dataset depicting (a) a staged robbery; (b) regular bank operations.

Example 1. Consider the two sequences of frames in Figures 4 depicting a staged robbery and regular bank operations respectively. Suppose that the following five activities may occur in a bank setting, and interleaving of activities is possible.

- (a_1) Regular customer-employee interaction, similar to the example in Figure 4.b.
- (a_2) Outsider enters the safe.
- (a_3) A bank robbery attempt – the suspected assailant does not make a getaway.
- (a_4) A successful bank robbery, similar to the example in Figure 4.a.
- (a_5) An employee accessing the safe on behalf of a customer.

We are interested in monitoring and detecting all of these activities concurrently.

In this section, we begin with the stochastic automaton activity model for video described in [3], and extend it in order to capture multiple activities in a single labeled stochastic automaton. We then define an index to index large numbers of observations from interleaved activities and efficiently retrieve completed instances of these activities. The stochastic activity model and the associated index presented in this section enable activity recognition not only from video data labeling, but also from any type of time-stamped data. In order to make this point clear, we will present examples where this approach has been applied to log data.

Definition 1 (Stochastic activity). A *stochastic activity* is a labeled graph (V, E, δ) where: V is a finite set of action symbols; E is a subset of $(V \times V)$; $\exists v \in V$ s.t. $\nexists v' \in V$ s.t. $(v', v) \in E$, i.e., there exists at least one *start node* in V; $\exists v \in V$ s.t. $\nexists v' \in V$ s.t. $(v, v') \in E$, i.e., there exists at least one *end node* in V; $\delta : E \rightarrow [0, 1]$ is a function that associates a probability distribution with the outgoing edges of each node, i.e., $\forall v \in V \sum_{\{v' \in V | (v,v') \in E\}} \delta(v, v') = 1$.

Example 2. Figure 5 shows the stochastic activity associated with ordering products from an online store. A user first accesses the product catalog (*catalog*) and either inspects the details of an item (*itemDetails*) or continues with a previously saved cart

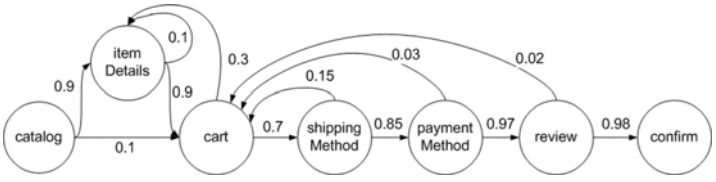


Fig. 5 Online purchase stochastic activity

Table 1 Example of a web log

id	ts	action	id	ts	action	id	ts	action	id	ts	action
1	1	catalog	6	5	itemDetails	11	8	itemDetails	16	11	paymentMethod
2	2	catalog	7	5	shippingMethod	12	9	cart	17	11	review
3	3	itemDetails	8	6	cart	13	9	review	18	12	confirm
4	3	cart	9	7	shippingMethod	14	10	confirm	19	13	paymentMethod
5	4	itemDetails	10	7	paymentMethod	15	11	shippingMethod	20	13	review

(*cart*). The checkout process requires the user to select a shipping method (*shipping-Method*), choose a payment method (*paymentMethod*), review the order (*review*) and confirm it (*confirm*). At each stage during checkout, the user can cancel and return to the cart and from there on to one of the items. The probabilities labeling the edges have the following intuitive meaning: once the *catalog* action has been taken, there is a .9 probability that the user will check details of an item (*itemDetails*) and a .1 probability that she will continue with a previously saved cart (*cart*).

Definition 2 (Activity instance). Let (V,E,δ) be a stochastic activity. An *instance* of (V,E,δ) is a sequence $\langle v_1,\ldots,v_n\rangle$ with $v_i \in V$ such that: (i) v_1 is a start node and v_n is an end node in (V,E,δ); (ii) $\forall i \in [1,n-1]$, $(v_i,v_{i+1}) \in E$. Thus, an activity instance is a path from a start node to an end node in (V,E,δ). The *probability* of the instance is $prob(\langle v_1,\ldots,v_n\rangle) = \Pi_{i\in[1,n-1]}\delta(v_i,v_{i+1})$.

Intuitively, an activity instance is a path from a start node to an end node – the probability of this activity instance is the product of the edge probabilities on the path. We assume that each node in an activity is an observable event. We also assume that the probability of taking an action at any time only depends on the previous event. Thus the probability labeling an edge (v_1,v_2) can be seen as the conditional probability of observing action v_2 given that action v_1 has been observed. These assumptions allow to easily learn the probabilities labeling the edges from training data and to compute the probability of an instance as a product of probabilities.

4.1.1 The Evidence and Identification Problems

We assume that all observations are stored in a single relational database table O. Each row (or tuple) o in the table denotes a single action, $o.action$, which is observed at a given time, $o.ts$.

Definition 3 (Activity occurrence). Let (V,E,δ) be a stochastic activity and let O be an observation table. An occurrence of (V,E,δ) in O with probability p is a

set $\{o_1,\ldots,o_n\} \subseteq O$ s.t. $o_1.ts \leq o_2.ts \ldots \leq o_n.ts$ and $\langle o_1.action,\ldots,o_n.action\rangle$ is an instance of (V,E,δ) with $prob(\langle o_1.action,\ldots,o_n.action\rangle) \geq p$. The *span* of the occurrence is the time interval $span(\{o_1,\ldots,o_n\}) = [o_1.ts, o_n.ts]$.

Intuitively, an activity occurrence is a sequence $\{o_1,\ldots,o_n\}$ of observations in O corresponding to the nodes of an activity instance and the probability of this occurrence is the probability of the instance.

Example 3. The activity in Figure 5 occurs in the server log of Table 1. In fact the sequence of observations with identifiers 1, 4, 7, 10, 13, and 14, corresponds to the instance $\{catalog, cart, shippingMethod, paymentMethod, review, confirm\}$.

Proposition 1. *Given an observation table O and a stochastic activity A, the problem of finding all occurrences of A in O takes exponential time, w.r.t.* $|O|$.

The reason for this behavior is interleaving of activities, which leads to an exponential number of identifiable occurrences of A in O. As an example, the set of observations: $\{catalog, cart, shippingMethod, paymentMethod, review, confirm, confirm\}$ leads to two occurrences of the activity in Figure 5, one for each of the *confirm* actions. Therefore, it is not feasible in practice to try to find all occurrences. Instead, we impose restrictions on what constitutes a valid occurrence in order to greatly reduce the number of possible occurrences. We propose two constraints applicable in most real-world scenarios. In addition, due to the size of the search space, it is important to have data structures that enable very fast searches for activity occurrences. We describe the Multi-Activity Graph Index Creation (MAGIC) [4] that allows to solve the *Evidence* and *Identification* problems efficiently.

Our first restriction requires that if the span of an occurrence O_2 is contained **within** the span of an occurrence O_1 we will discard O_1 from the set of results. This is called the *minimal span restriction* (MS for short).

Definition 4 (MS restriction). Let $O_1 = \{o_1,\ldots,o_k\} \subseteq O$ and $O_2 = \{o'_1,\ldots,o'_l\} \subseteq O$ be two occurrences of the same activity. We say that the span of O_1 is less than or equal to the span of O_2 – $span(O_1) \leq span(O_2)$ – iff $o_1.ts \geq o'_1.ts$ and $o_k.ts \leq o'_l.ts$. Under the *MS restriction*, we only consider occurrences that are minimal w.r.t. span.

The MS restriction may still allow exponentially many occurrences of an activity, since multiple occurrences may have the same span. The *earliest action* (EA for short) restriction requires that when looking for the next action in an activity occurrence, we always choose the first possible successor. For instance, consider the activity definition $v_1 \xrightarrow{1} v_2 \xrightarrow{1} v_3$ and the observation sequence $\{v_1^1, v_2^2, v_1^3, v_2^4, v_3^5\}$ where v_j^i denotes the fact that action v_j was observed at time i. There are two occurrences starting with v_1^1, namely $\{v_1^1, v_2^2, v_3^5\}$ and $\{v_1^1, v_2^4, v_3^5\}$. Under the EA restriction we only consider $\{v_1^1, v_2^2, v_3^5\}$, since v_2^2 is the first possible successor to v_1^1. This restriction makes the search space linear in the size of the observation sequence.

Definition 5 (EA restriction). An activity occurrence $\{o_1,\ldots,o_n\} \subseteq O$ is said to have the *earliest action* property if $\forall i \in [2,n], \nexists w_i \in O$ s.t. $w_i.ts < o_i.ts$ and $\{o_1,\ldots,o_{i-1}, w_i, o_{i+1},\ldots,o_n\}$ is an occurrence of the same activity.

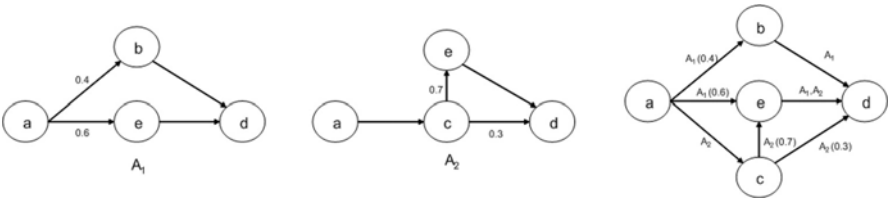


Fig. 6 Stochastic activities and multi-activity graph

It is now straightforward to generalize the *Evidence* and *Identification* problems stated in Section 2.1. In particular, in the Evidence problem we are now interested in recognizing all the *possibly restricted* occurrences of activities in $\mathscr{A}$, from any sequence of time-stamped observations, including video data as a particular case.

4.1.2 Multi-activity Graph Index Creation

In order to concurrently monitor multiple activities, we first merge all activity definitions from $\mathscr{A} = \{A_1, \ldots, A_k\}$ into a single graph.

Definition 6 (Multi-activity graph). Let $\mathscr{A}=\{A_1, \ldots, A_k\}$ be a set of stochastic activities, where $A_i = (V_i, E_i, \delta_i)$. A *Multi-Activity Graph* is a triple $G = (I_{\mathscr{A}}, V_G, \delta_G)$ where: (i) $I_{\mathscr{A}} = \{id(A_1), \ldots, id(A_k)\}$ is a set of identifiers for activities in $\mathscr{A}$; (ii) $V_G = \cup_{i\in[1,k]} V_i$ is a set of action symbols; (iii) $\delta_G : V_G \times V_G \times I_{\mathscr{A}} \rightarrow [0,1]$ is a function that associates a triple $(v, v', id(A_i))$ with $\delta_i((v, v'))$, if $(v, v') \in E_i$ and 0 otherwise.

A multi-activity graph can be graphically represented by labeling nodes with action symbols and edges with id's of activities containing them, along with the corresponding probabilities. The multi-activity graph for a set $\mathscr{A}$ of activities can be computed in time polynomial in the size of $\mathscr{A}$. Figure 6 shows two stochastic activities, A_1 and A_2, and the corresponding multi-activity graph. If $A = (V, E, \delta)$ is an activity and $v \in V$, we use $A.p_{max}(v)$ to denote the maximum product of probabilities on any path in A between v and an end node. The *multi-activity graph index* allows us to efficiently monitor activity occurrences new observations occur.

Definition 7 (Multi-activity graph index). Let $\mathscr{A} = \{A_1, \ldots, A_k\}$ be a set of stochastic activities, where $A_i = (V_i, E_i, \delta_i)$, and let $G = (I_{\mathscr{A}}, V_G, \delta_G)$ be the multi-activity graph built over $\mathscr{A}$. A *Multi-Activity Graph Index* is a 6-tuple $I_G = (G, start_G, end_G, max_G, tables_G, completed_G)$, where:

- $start_G : V_G \rightarrow 2^{I_{\mathscr{A}}}$ is a function that associates each node $v \in V_G$ with the set of activity-id's for which v is a start node;
- $end_G : V_G \rightarrow 2^{I_{\mathscr{A}}}$ is a function that associates each node $v \in V_G$ with the set of activity id's for which v is an end node;
- $\max_G : V_G \times I_{\mathscr{A}} \rightarrow [0,1]$ is a function that associates a pair $(v, id(A_i))$ with $A_i.p_{max}(v)$, if $v \in V_i$ and 0 otherwise;
- For each $v \in V_G$, $tables_G(v)$ is a set of tuples of the form $(current^{\uparrow}, actID, t_0$ is a timestamp, $probability$, $previous^{\uparrow}$, $next^{\uparrow})$, where $current^{\uparrow}$ is a pointer to an

Algorithm 1. *MAGIC-insert*(t_{new}, I_G, p_t)

Require: New observation to be inserted t_{new}, multi-activity graph index I_G, probability threshold p_t
Ensure: Updated multi-activity graph index I_G

```
1: for all tuple t ∈ tables_G(t_new.action) do {Look at start nodes}
2:     if t.activityID ∈ start_G(t_new.action) and t.next = ⊥ then
3:         t.current↑ ← t_new↑
4:     end if
5: end for
6: for all activity id for which no tuple was updated in the previous loop do
7:     add (t_new↑, id, t_new.ts, 1, ⊥, ⊥) to tables_G(t_new.action)
8: end for
9: for all action symbol v ∈ V_G s.t. ∃id ∈ I_A, δ_G(v, t_new.action, id) ≠ 0 do {Look at intermediate nodes}
10:    for all tuple t ∈ tables_G(v) with t.next = ⊥ and maximal t_0 w.r.t. tuples in tables_G(v) with the same actID do
11:        a ← t_new.action, id ← t.activityID
12:        if δ_G(v,a,id) ≠ 0 and t.probability · δ_G(v,a,id) · max_G(a,id) ≥ p_t then
13:            t' ← (t_new↑, id, t.t_0, t.probability · δ_G(v,a,id), t↑, ⊥)
14:            add t' to tables_G(a)
15:            t.next ← t'↑
16:            if id ∈ end_G(a) then {Look at end nodes}
17:                add t'↑ to completed_G(id)
18:            end if
19:        end if
20:    end for
21: end for
```

observation, $actID \in I_{\mathscr{A}}$, t_0, $probability \in \mathbb{R}^+$, $previous^{\uparrow}$ and $next^{\uparrow}$ are pointers to tuples in $tables_G$;

- $completed_G : I_{\mathscr{A}} \rightarrow 2^{\mathscr{P}}$, where $\mathscr{P}$ is the set of tuples in $tables_G$, is a function that associates an activity identifier $id(A)$ with a set of references to tuples in $tables_G$ corresponding to a completed instance of activity A.

Note that $G, start_G, end_G, max_G$ can be computed a-priori. All the tables that are part of the index will be initially empty. As new observations are added, the index tables will be updated accordingly. Therefore, MAGIC can track *partially-completed* activity occurrences.

4.1.3 MAGIC Insertion Algorithm

This section describes an algorithm to update the MAGIC index under the MS and EA restrictions when new observations occur (Algorithm 1). We refer the reader to [4] for a detailed example. Lines 1–8 handle the case when the observation contains an action that is the start node of an activity in $\mathscr{A}$. Tuples in the table associated with the action are updated to minimize span, unless they already have a successor. In this case a new tuple is added, indicating the start of a new concurrent occurrence. Lines 9–15 look at the tables associated with the predecessors of $t_{new}.action$ in the multi-activity graph and check whether the new observation can be linked to existing partial occurrences. For each activity in $\mathscr{A}$, the tuple with the most recent t_0 is linked, to minimize the span. Moreover, in order to impose the EA restriction, the algorithm requires that each tuple has at most one successor. We also enforce the probability threshold by detecting whether the partial occurrence can still have a probability above the threshold on completion. If all these conditions are met, a new tuple t' is added to the table associated with $t_{new}.action$. Finally, lines 16–18 check

whether $t_{new}.action$ is an end node for some activity; in this case, a pointer to t' is added to $completed_G$ signaling that an occurrence has been completed.

Proposition 2. *Algorithm* MAGIC-insert *runs in time* $\mathcal{O}(|\mathscr{A}| \cdot max_{(V,E,\delta)\in\mathscr{A}}(|V|) \cdot |O|)$*, where O is the set of observations indexed so far.*

The *MAGIC-evidence* algorithm finds all minimal sets of observations that validate the occurrence of activities in $\mathscr{A}$ with a probability exceeding a given threshold. The *MAGIC-id* algorithm, which solves the *Identification* problem, identifies the activities in $\mathscr{A}$ that have maximum probability of occurring in the required time span. For reasons of space, we omit a detailed description of the two algorithms and refer the reader to [4] for further details.

4.2 Probabilistic Petri Nets

Petri Nets (PNs) [21] were defined as a mathematical tool for describing relations between conditions and events, and are particularly useful to model and visualize behaviors such as sequencing, concurrency, synchronization and resource sharing. David *et al.* [8] and Murata [19] provide comprehensive surveys on Petri Nets. Several forms of PNs such as Colored PNs, Continuous PNs, Stochastic timed PNs, and Fuzzy PNs have been proposed. Colored PNs associate a *color* to each token, hence are useful to model complex systems where each token can potentially have a distinct identity. In Continuous PNs, the markings of places are *real numbers*, hence they can be used to model situations where the underlying physical processes are continuous in nature. Stochastic timed PNs [18] associate, to each transition, a probability distribution representing the delay between the enabling and firing of the transition. Fuzzy PNs [7] are used to represent fuzzy rules between propositions.

In real-life situations, vision systems have to deal with ambiguities and inaccuracies in the lower-level detection and tracking systems. Moreover, activities may not unfold exactly as described by the model that represents them. The aforementioned types of PNs are not well suited to deal with these situations. The probabilistic PN model [2] described in this section is better suited to express uncertainty in the state of a token or associate a probability to a particular unfolding of the Petri Net.

Definition 8 (Constrained Probabilistic Petri Net). A *Constrained Probabilistic Petri Net* PPN is a 5-tuple $\{P,T,\rightarrow,F,\delta\}$, where

- P and T are finite disjoint sets of places and transitions respectively, i.e., $P\cap T=\emptyset$.
- $\rightarrow$ is the flow relation between places and transitions, i.e., $\rightarrow\subseteq(P\times T)\cup(T\times P)$.
- The preset ${}^{\cdot}x$ of a node $x\in P\cup T$ is the set $\{y|y\rightarrow x\}$.
- The postset $x^{\cdot}$ of a node $x\in P\cup T$ is the set $\{y|x\rightarrow y\}$.
- F is a function defined over the set of places P, that associates each place x with a local probability distribution $f_x(t)$ over $\{t|x\rightarrow t\}$. For each place $x\in P$, we denote by $p^*(x)$ the maximum product of probabilities over all paths from x to a terminal node. We will often abuse notation and use $F(x,t)$ to denote $f_x(t)$.

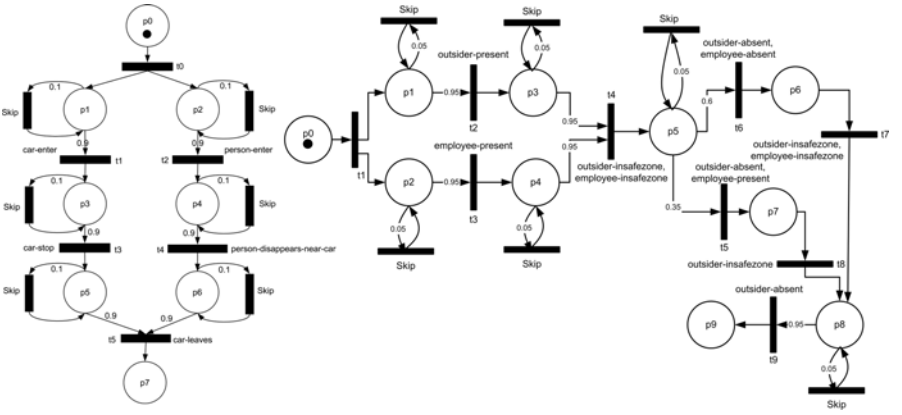
- $\delta : T \rightarrow 2^{\mathscr{A}}$ associates a set of action symbols with every transition in the Petri Net. Transitions will only fire when all the action symbols in the constraint are also encountered in the video sequence labeling.
- $\exists x \in P$ s.t. $x^{\cdot} = \emptyset$, i.e., there exists at least one terminal node in the Petri Net.
- $\exists x \in P$ s.t. ${}^{\cdot}x = \emptyset$, i.e., there exists at least one start node in the Petri Net.

4.2.1 Tokens and Firing of Transitions

Petri Net dynamics are represented via 'markings'. For a PPN $(P, T, \rightarrow, F, \delta)$, a *marking* is a function $\mu : P \rightarrow \mathbb{N}$ that assigns a number of *tokens* to each place in P. In a PPN modeling an activity, a marking μ represents the current state of completion of that activity. The execution of a PN is controlled by its current marking. A transition is *enabled* if and only if all its input places (the preset) have a token. When a transition is enabled, it may *fire*. When a transition fires, all enabling tokens are removed and a token is placed in each of the output places of the transition (the postset). We use μ_0 to denote the *initial marking* of a PPN. A *terminal marking* is reached when one of the terminal nodes contains at least one token. In the simplest case, all the enabled transitions may fire. However, to model more complex scenarios we can impose other conditions to be satisfied before an enabled transition can fire. This set of conditions is represented by δ.

Example 4. Consider the car pickup activity modeled by the PPN in Figure 7.a, with places labeled $p_1, \ldots, p_7$ and transitions $t_0, \ldots, t_5$, p_0 being the start node and p_7 the terminal node. In the initial marking depicted in the figure all places except p_0 have 0 tokens. Transition t_0 is unconstrained and it is always fired, adding a token in both p_1 and p_2. When a car enters the scene, t_1 fires and a token is placed in p_3 and we have a new marking μ_1 such that $\mu_1(p_3) = 1$ and $\mu_1(p) = 0$ for any $p \neq p_3$. The transition t_3 is enabled in this state, but it cannot fire until the condition associated with it is satisfied – i.e., when the car stops. When this occurs, and a person enters the parking lot and then disappears near the car, the Petri Net evolves to a state where there is a token in each of the enabling places of transition t_5. Once the car leaves, t_5 fires and both the tokens are removed and a token placed in the final place p_7. This example illustrates sequencing, concurrency and synchronization.

In the above discussion, we have not yet discussed the probabilities labeling the place-transition edges. Note that the postsets of $p_1, \ldots p_6$ contain multiple transitions. The *skip* transitions are used to *explain away* deviations from the base activity pattern – each such deviation is penalized by a low probability. The probabilities assigned to skip transitions control how tolerant the model is to deviations from the base activity pattern. All tokens are initially assigned a probability of 1. Probabilities are accumulated by multiplying the token and transition probabilities on the edges. When two or more tokens are removed from the net and replaced by a single token then the probability for the new token is set to be the product of the probabilities of the removed tokens. We will use the final probability of a token in a terminal node as the probability that the activity is satisfied.



(a) Car-pickup (b) Bank attack

Fig. 7 Examples of Probabilistic Petri Nets

4.2.2 Activity Recognition

We now define the concepts of *PPN trace* and *activity satisfaction* and provide a method for computing the probability with which the PPN is satisfied.

Definition 9 (PPN trace). A trace for a PPN $(P,T,\rightarrow,F,\delta)$ with initial marking μ_0 is a sequence of transitions $\langle t_1,\ldots,t_k\rangle \subseteq T$ such that:

(i) t_1 is enabled in marking μ_0.

(ii) Firing $t_1,\ldots,t_k$ in that order reaches a terminal marking.

For each $i \in [1,k]$, let $p_i = \Pi_{\{x\in P|x\rightarrow t_i\}} F(x,t_i)$. The trace $\langle t_1,\ldots,t_k\rangle$ has probability $p = \Pi_{i\in[1,k]} p_i$[1]. Let p_{max} be the maximum probability of any trace for $(P,T,\rightarrow,F,\delta)$. Then $\langle t_1,\ldots,t_k\rangle$ has *relative probability* $\frac{p}{p_{max}}$. Intuitively, the relative probability measures how close that trace is to the *ideal* execution of the PPN, i.e., the execution that leads to the maximum possible absolute probability.

Example 5. Consider the PPN in Figure 7.b. $\langle t_1,t_2,t_3,t_4,t_6,t_7,t_9\rangle$ is a trace with probability .464 and a relative probability 1.

Definition 10 (Activity satisfaction). Let $\mathscr{P} = (P,T,\rightarrow,F,\delta)$ be a PPN, let v be a video sequence and let ℓ be the labeling of v. We say ℓ satisfies $\mathscr{P}$ with *relative probability* $p \in [0,1]$ iff there exists a trace $\langle t_1,\ldots,t_k\rangle$ for $\mathscr{P}$ such that:

(i) There exist frames $f_1 \leq \ldots \leq f_k$ such that $\forall i \in [1,k]$, $\delta(t_i) \subseteq \ell(f_i)$ AND

(ii) The relative probability of $\langle t_1,\ldots,t_k\rangle$ is equal to p.

We will refer to a video sequence v satisfying an activity definition, as an equivalent way of saying the labeling of v satisfies the activity. If ℓ is the labeling of v and $v' \subseteq v$ is a subsequence of v, we will also use ℓ to refer to the restriction of ℓ to v'.

[1] We are assuming that the initial probability assigned to tokens in the start nodes is 1.

4.2.3 The Evidence and Identification Problems

In this section we present an algorithm for the Evidence Problem stated in Section 2.1, PPN-evidence (Algorithm 2), which simulates the constrained PPN forward. The algorithm uses an ad-hoc data structure to store tuples of the form $\langle \mu, p, f_s \rangle$, where μ is a valid marking, p is the probability of the partial trace that leads to μ and f_s is the frame when the first transition in that trace was fired.

The algorithm starts by adding $\langle \mu_0, 1, 0 \rangle$ to the store S (line 1). This means we start with the initial marking μ_0 and probability equal to 1. The value 0 for the start frame means the first transition has not yet fired. Whenever the first transition fires and the start frame is 0, the current frame is chosen as a start for this candidate subsequence (lines 8–9). The algorithm iterates through the video frames, and at each iteration analyzes the current candidates in S. Any transitions t enabled in the current marking that have the conditions $\delta(t)$ satisfied are fired. The algorithm generates a new marking, and with it a new candidate partial subsequence (line 7). If the new probability p' is still above the threshold (line 15) and can remain above the threshold on the best path to a terminal marking (line 16), the new state is added to the store S. This first pruning does away with any states that will result in low probabilities. Note that we have not yet removed the old state from S, since we also need to fire any enabled skip transition, in order to explore the space of solutions. This is done on line 26. At this point (line 27) we also prune any states in S that have no possibility of remaining above the threshold as we reach a terminal marking.

The following two theorems state correctness and complexity results for the PPN-evidence algorithm. We refer the reader to [2] for proofs of both theorems.

Theorem 1 (PPN-evidence correctness). *Let $\mathscr{P} = (P, T, \rightarrow, F, \delta)$ be a PPN with initial configuration μ_0, let v be a video sequence and ℓ its labeling, and let $p_t \in [0,1]$ be a probability threshold. Then for any subsequence $v' \subseteq v$ that satisfies $\mathscr{P}$ with probability greater than or equal to p_t, one of the following holds:*

(i) v' is returned by PPN-evidence OR

(ii) $\exists v'' \subseteq v$ that is returned by PPN-evidence such that $v'' \subseteq v'$.

Theorem 2 (PPN-evidence complexity). *Let $\mathscr{P} = (P, T, \rightarrow, F, \delta)$ be a PPN with initial configuration μ_0, such that the number of tokens in the network at any marking is bounded by a constant k. Let v be a video sequence and ℓ its labeling. Then PPN-evidence runs in time $\mathscr{O}(|v| \cdot |T| \cdot |P|^k)$.*

We now briefly describe an algorithm for the Identification Problem. Listing of the algorithm, detailed description and correctness theorem are omitted for reasons of space. We refer the reader to [4] for further details.

Let $\{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$ be a given set of activity definitions. If we assume that there is only one activity $\mathscr{P}_l$ which satisfies the video sequence v with maximal probability, a simple binary search iterative method – naivePPN-ident – can employ PPN-evidence to find the answer: naivePPN-ident runs PPN-evidence for all activities in $\{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$ and for different thresholds until a delimiter threshold is found (i.e., one for which only one activity $\mathscr{P}_l$ has a non-empty result from PPN-evidence).

Algorithm 2. PPN-evidence($\mathscr{P}$, μ_0, ℓ, p_t)

Require: $\mathscr{P} = (P, T, \rightarrow, F, \delta)$, initial configuration μ_0, video sequence labeling ℓ, probability threshold p_t
Ensure: Set of minimal subsequences that satisfy $\mathscr{P}$ with relative probability above p_t

$S \leftarrow \{\langle \mu_0, 1, 0\rangle\}$
$R \leftarrow \emptyset$
$max_p \leftarrow$ the maximum probability for any trace of $\mathscr{P}$
for all f frame in video sequence **do**
  for all $\langle \mu, p, f_s\rangle \in S$ **do**
    for all $t \in T$ enabled in μ s.t. $\delta(t) \subseteq \ell(f)$ **do**
      $\mu' \leftarrow$ fire transition t for marking μ
      if $f_s = 0$ and $\delta(t) \neq \emptyset$ **then**
        $f' \leftarrow f$
      else
        $f' \leftarrow f_s$
      end if
      $p^* \leftarrow \Pi_{\{x \in P | x \rightarrow t\}} F(x,t)$
      $p' \leftarrow p \cdot p^*$
      if $p' \geq p_t \cdot max_p$ **then**
        if $\exists x \in P$ s.t. $\mu(x) < \mu'(x) \wedge p' \cdot p^*(x) \geq p_t \cdot max_p$ **then**
          $S \leftarrow S \cup \{\langle \mu', p', f'\rangle\}$
        end if
      end if
      if μ' is a terminal configuration **then**
        $S \leftarrow S - \{\langle \mu', p', f'\rangle\}$
        $R \leftarrow R \cup \{[f_s, f]\}$
      end if
    end for
    for all skip transition $t \in T$ enabled **do**
      Fire skip transition if no other transitions fired and update $\langle \mu, p, f_s\rangle$
      Prune $\langle \mu, p, f_s\rangle \in S$ s.t. $\forall x \in P$ s.t. $\mu(x) > 0$, $p \cdot p^*(x) < p_t \cdot max_p$
    end for
  end for
end for
Eliminate non-minimal subsequences in R
return R

A more efficient solution is PPN-ident which uses a similar storage structure as PPN-evidence (tuples of the form $\langle \mu, p, \mathscr{P}\rangle$, where $\mathscr{P}$ is the activity definition to which marking μ applies). The algorithm maintains a global maximum relative probability max_p with which the video satisfies any of the activities in $\{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$ and a list R of the activity definitions that have been satisfied with probability max_p. max_p is updated any time a better relative probability is found.

Theorem 3 (PPN-ident complexity). *Let* $\{(P_i, T_i, \rightarrow_i, F_i, \delta_i)\}_{i \in [1,m]}$ *be a set of PPNs, let v be a video sequence. PPN-ident runs in time* $\mathscr{O}(m \cdot |v| \cdot |T| \cdot |P|^k)$.

5 Language Based Paradigm: PADL

The task of modeling complex activities using the models discussed in the previous section may be challenging when the models need to be created manually by users. Although a lot of work has been done on learning model parameters from training data, learning the whole model from training data is still an open issue. Therefore, we also explore the definition of logical languages to define complex activities in an expressive yet formal fashion. Thus, we introduce PADL (Probabilistic Activity Detection Language), an extensible logical language including a set of boolean and

probabilistic predicate symbols which map to primitive actions that can be recognized through image processing algorithms. Along with PADL, we propose a suite of offline and real-time algorithms that solve the Evidence and Online Identification problems stated in Section 2.1.

5.1 Syntax of PADL

PADL is a *logical language* which has a set of constant, function and variable symbols, a set of boolean predicate symbols, and, unlike ordinary logic, a set of probabilistic predicate symbols. PADL builds on top of a library of image processing algorithms. Each algorithm implements either a boolean or a probabilistic predicate. PADL can be extended by extending the underlying image processing library.

Definition 11 (atom). If p is a predicate symbol with k arguments, and $t_1,\ldots,t_k$ are either variables or constants of the right types, then $p(t_1,\ldots,t_k)$ is an *atom*. $p(t_1,\ldots,t_k)$ is an *uninstantiated atom* if at least one of the t_i's is a variable.

Note that the definition of an atom does not distinguish between boolean and probabilistic predicates. We will address the issue in Section 5.2 when discussing probabilistic activity satisfaction. We now define the set of well-formed formulas in PADL. The degree of a formula is the number of alternating $\forall$ and $\exists$ quantifiers.

Definition 12 (activity formula). An activity formula is defined as follows:

1. Every atom is an activity formula of degree 0.
2. If F is a formula of degree 0 and x is a variable, then $(\forall x)F$ and $(\exists x)F$ are activity formulas of degree 1.
3. If F, G are activity formulas of degree m,n respectively, then the conjunction $(F \wedge_{\otimes} G)$ and the disjunction $(F \vee G)$ are activity formulas of degree $\max(m,n)$. By $(F \wedge_{\otimes} G)$ we denote that, if F and/or G are probabilistic, the probability of their conjunction can be obtained using the t-norm $\otimes$[2]. We will assume that the default t-norm is independence ($x \otimes y = xy$), unless otherwise specified.
4. If F is an activity formula of degree k and x is a variable that is not within the scope of any quantifier in F, then:
   a. If F has the form $(\forall ...)G$ then $(\forall x)F$ is an activity formula of degree k.
   b. If F has the form $(\exists ...)G$ then $(\forall x)F$ is an activity formula of degree $k+1$.
   c. If F has the form $(\forall ...)G$ then $(\exists x)F$ is an activity formula of degree $k+1$.
   d. If F has the form $(\exists ...)G$ then $(\exists x)F$ is an activity formula of degree k.

A well known result in logic [22] states that, as the degree of an activity formula becomes higher, the problem of finding occurrences of the activity becomes more complex. Moreover, if two formulas have the same form, but a different leading quantifier, $(\exists)$ and $(\forall)$ respectively, then the former is easier to solve than the latter.

[2] We omit the notations on disjunctions, as they can be expressed by conjunction and negation.

5.2 Probabilistic Activity Satisfaction

As mentioned earlier, PADL assumes the existence of a set of image processing algorithms By applying these algorithms to each frame in a video, we obtain a *labeling* of the video.

Definition 13 (Frame labeling). Suppose $\mathscr{O}$ is a universe of distinct reference image objects (e.g. a database of mugshots or a set of car images). A *labeling* is a pair (ℓ, pl) where ℓ is a set of ground boolean atoms and pl is a function which assigns values in $[0,1]$ to each probabilistic ground atom, such that for all objects o of interest in a given frame $\sum_{o' \in O} \mathsf{pl}(\mathsf{eq}(o,o')) = 1$[3].

We will abuse notation and use (ℓ, pl) to denote the labeling of the entire video sequence or any subsequence of it. Once we obtain the video labeling (ℓ, pl), we need to compute the probability that it satisfies an activity formula. Intuitively, a formula is satisfied if we can substitute objects from the labeling for the variables in the formula. A *substitution* θ is any set $\theta = \{X_1 = v_1, \ldots, X_n = v_n\}$ where the X_i's are variables and the v_i's are constants. Given an activity formula F, we use $F\theta$ to denote the replacement of all occurrences of X_i in F by v_i. If θ_1, θ_2 are substitutions, $(\theta_1 \cup \theta_2)$ is solvable if and only if the substitutions are consistent.

Given a labeling (ℓ, pl) of a video and an activity formula F, we now define the concept of a substitution set $\delta(F)$ for F. Intuitively, $\delta(F)$ contains all pairs (θ, p) where θ is a substitution that binds existentially quantified variables in F to objects or frame numbers and p is the probability that F is satisfied by the labeling when applying substitution θ. These informal notions are formalized below.

Definition 14 (Substitution sets for activity formulas). Let F be an activity formula, v a video sequence, and (ℓ, pl) a labeling of v. The *substitution set* for F, denoted $\delta(F)$, is defined as follows:

(i) If F is a boolean atom, then $\delta(F) = \{(\theta, 1) \mid F\theta \in \ell(v)\}$.
(ii) If F is a probabilistic atom, then $\delta(F) = \{(\theta, \mathsf{pl}(F\theta)) \mid (F\theta, \mathsf{pl}(F\theta)) \in \mathsf{pl}(v)\}$.
(iii) If $F = (\exists\, x)G$, then $\delta(F) = \delta(G)$.
(iv) If $F = (\forall\, x)G$, $\delta(F) = \delta(\bigwedge_{o \in \mathscr{O}_x} G([x/o]))$, where $\mathscr{O}_x$ is the set of possible values for x.
(v) If $F = G \wedge_{\otimes} H$, then $\delta(F) = \{(\theta_1 \cup \theta_2, v_1 \otimes v_2) \mid (\theta_1, v_1) \in \delta(F) \wedge (\theta_2, v_2) \in \delta(G) \wedge (\theta_1 \cup \theta_2)$ is solvable $\}$.
(vi) If $F = \neg G$, let Θ_G contain all substitutions for variables in G. Then $\delta(F) = \{(\theta, p) \mid (\theta, 1-p) \in \delta(G)\} \cup \{(\theta, 1) \mid \nexists (\sigma, v) \in \delta(G)$ s.t. $\sigma \cup \theta$ is solvable $\}$
(vii) If $F = G \vee H$, then $\delta(F) = \delta(G) \cup \delta(H)$.

Example 6. Consider the frame sequence in Figure. 8.a. We assume that the set of reference objects consists of *John Doe*, *Joe Doe* – the two protagonists – and the object *Bag*. The boolean labeling ℓ and the probabilistic labeling pl might

[3] For each object o in the video, $\mathsf{pl}(\mathsf{eq}(o,o'))$ is the probability that o is the same as some $o' \in \mathscr{O}$.

Fig. 8 Frames depicting (a) a package transfer from the ITEA CANDELA dataset; (b) tarmac operations from the TSA dataset.

be: $\ell = \{\text{in}(p_1, f_1), \text{haspkg}(p_1, pkg_1, f_1), \text{in}(p_1, f_2), \text{in}(p_2, f_2), \text{haspkg}(p_1, pkg_1, f_2), \text{in}(p'_1, f_3), \ldots\}$ $\mathsf{pl} = \{(\mathsf{eq}(p_1, p'_1), 0.91), (\mathsf{eq}(p_2, p'_2), 0.95), (\mathsf{eq}(p_1, p_2), 0.07), (\mathsf{eq}(p_1, \textit{John Doe}), 0.94), \ldots\}$

Now consider the activity formula $F = \text{in}(P_1, F_1) \wedge \text{in}(P_2, F_2) \wedge F_1 < F_2 \wedge eq(P_1, P_2)$, which requires the same person appear in two different frames F_1 and F_2. The substitution set for F is $\delta(F) = \{(\{P_1 \rightarrow p_1, P_2 \rightarrow p_1, F_1 \rightarrow f_1, F_2 \rightarrow f_2\}, 1), (\{P_1 \rightarrow p_1, P_2 \rightarrow p'_1, F_1 \rightarrow f_2, F_2 \rightarrow f_3\}, 0.91), \ldots\}$.

We now formally define what constitutes an answer to the *Evidence* problem stated in Section 2.1. we then present the OffPad algorithm that solves the Evidence problem, and the OnPad algorithm that solves the Online Identification problem.

Definition 15 (answer). Suppose (ℓ, pl) is the labeling of video v, F is an activity formula, and p is a real number in the $[0,1]$ interval. A sub-sequence sv of v is said to *minimally satisfy F with probability p* iff

i There exists a pair $(\theta, q) \in \delta(F)$ such that $sv = [start(F), end(F)]$ and $q \geq p$[4].

ii There is no strict sub-sequence sv' of sv which satisfies the above condition.

5.3 The OffPad Algorithm

The OffPad algorithm (Algorithm 3) uses the *answer_af* method (Algorithm 4) to compute the substitution set $\delta(F)$ for the activity formula F, whereas the main body of the algorithm uses the substitutions returned by *answer_af* to compute the minimal contiguous sub-sequences of the video that satisfy the given activity definition.

[4] We assume that the user has explicitly denoted some variables in F as *frame* variables, and marked two frame variables as a start variable $start(F)$ and an end variable $end(F)$ respectively.

A brief outline of the algorithm reads as follows:

1. Compute the valid substitutions $\delta(F)$ by recursively decomposing F into parts.
2. Select those substitutions that have probability over the threshold.
3. For each substitution, find the sequence *sv* between start and end frame variables.
4. If a subsequence of *sv* is already in the result, continue to step 6.
5. Otherwise, add *sv* to the result.
6. Repeat steps 3–5.

The *answer_af* algorithm uses a recursive approach – based on Definition 14 – to compute the subset of $\delta(F)$ which contains only those elements with probabilities above the threshold p_t (we denote this subset by $\delta(F)|_{p_t}$). The algorithm optimizes the search for substitutions in three ways. The first is based on the observation that a conjunction of multiple formulas allows us to combine the intermediate results for each formula in any order we may choose; the *heuristic_order* algorithm (omitted for reasons of space) is used to give an ordering that is computationally optimal.

The second optimization uses the fact that we can eliminate those intermediate results that cannot possibly lead to solutions with probabilities higher than the threshold. The *pruning* method removes substitutions that would yield results under the given threshold. Consider the case of computing the substitution set for $F \wedge G$ with probability threshold p_t. A substitution $(\theta, p) \in \delta(F)$ can only be combined with substitutions in $\{(\theta', p') \in \delta(G) | p' \geq min_{\otimes}(p, p_t)\}$. If the cardinality of this set is small, the number of operations in computing $\delta(F \wedge G)$ is greatly reduced.

The third optimization separates the set of frame comparison predicates (e.g., $t_1 < t_2$) from the rest (lines 1, 13). The number of frames is much larger than the number of objects in a video, hence frame variables have a huge number of possible substitutions. We delay searching for substitutions for frame variables until the end of the method because pruning may make this costly operation unnecessary.

Algorithm 3. OffPad(F, v, p_t, V_s, V_e))

Require: Activity formula F, video v, probability threshold p_t, variables V_s, V_e denoting start and end frames of result.
Ensure: Set of video subsequences satisfying F with probability above p_t

```
1:  R ← answer_af(F, v, p_t, pruning)
2:  R' ← ∅
3:  for all (θ, ℘) ∈ R do
4:      should_add ← true
5:      ⟨f_s, f_e⟩ ← ⟨V_s θ, V_e θ⟩
6:      for all [f, f'] ∈ R' do
7:          if [f, f'] ⊇ [f_s, f_e] then
8:              R' ← R' − {[f, f']}
9:          else if [f, f'] ⊆ [f_s, f_e] then
10:             should_add ← false
11:         end if
12:     end for
13:     if should_add then
14:         R' ← R' ∪ {[f_s, f_e]}
15:     end if
16: end for
17: S ← ∅
18: for all [f_s, f_e] ∈ R' do
19:     S ← S ∪ {subvideo(v, f_s, f_e)}
20: end for
21: return S
```

Algorithm 4. *answer_af*$(F, v, p_t, pruning)$

Require: Activity formula F, video v, probability threshold p_t, pruning method *pruning*. If M is a substitution set and F_A a set of boolean atoms, we denote by $M|^{F_A} = \{(\theta, p) \in M | \forall\ A\ \in F_A, A\theta$ *is true*$\}$. *pruning_method*(M_1, M_2, p_t) returns a pair of substitution sets $\langle M_1', M_2' \rangle$ such that $M_1' \subseteq M_1$ and $M_2' \subseteq M_2$ and $(M_1' \wedge M_2')|_{p_t} = (M_1 \wedge M_2)|_{p_t}$.
Ensure: $\delta(F)|_{p_t}$.

```
1: F_C ← ∅;
2: F_A ← {a atom in F | a involves frame variable comparisons}
3: if F is an atom then
4:     return δ(F)|_{p_t};
5: else if F is of type (∃ x)G then
6:     return answer_af(G, v, p_t, pruning)
7: else if F is of type (∀ x)G then
8:     for all o ∈ 𝒪_x do {𝒪_x is the set of all possible values of x}
9:         F_C ← F_C ∪ {G([x/o])}
10:    end for
11:    F_C ← F_C − F_A
12: else if F = G_1 ∧ ··· ∧ G_n then
13:    F_C ← {G_1, ···, G_n} − F_A
14: else if F = ¬G then
15:    M ← answer_af(G, v, 0, pruning)
16:    M' ← {(θ, p) | (θ, 1 − p) ∈ M} ∪ {(θ, 1) | ∄(σ, v) ∈ M s.t. θ ∪ σ solvable }
17:    return M'|_{p_t}
18: else if F = G ∨ H then
19:    return answer_af(G, v, p_t, pruning) ∪ answer_af(H, v, p_t, pruning)
20: end if
21: p ← p_t
22: for all F_i ∈ F_C do
23:    M_i ← answer_af(F_i, v, p, pruning)
24:    p ← min_⊗(max({p' | (θ, p') ∈ M_i}), p)
25: end for
26: T ← heuristic_order({M_1, ···, M_i, ···}, p_t)
27: while |T| > 1 do
28:    for all nodes M_i, M_j with the same parent do
29:        ⟨M_i', M_j'⟩ ← pruning(M_i, M_j, p_t)
30:        parent(M_i, M_j) ← (M_i' ∧ M_j')|_{p_t}
31:        if parent(M_i, M_j) = ∅ then
32:            return ∅
33:        end if
34:        remove M_i, M_j from T
35:    end for
36: end while
37: return root(T)|^{F_A}_{p_t}
```

In the case of formulas of type $\neg G$ (lines 14–17), the algorithm computes the entire set of possible substitutions for the variables in G; this is because the answer to $\neg G$ contains – for boolean predicates – all substitutions that *are not* in $\delta(G)$ which is very expensive. The OffPad algorithm takes the substitution set returned by *answer_af*, along with the start and end frame variables and, for each substitution returned, determines the corresponding video sub-sequence. In order to ensure the minimality condition, such sub-sequences are retained if and only if they have no sub-sequence that satisfies the activity definition with a probability above the threshold. The following theorems state correctness and complexity results for *answer_af* and OffPad. Proofs are omitted for reasons of space.

Lemma 1. *Let F be an activity formula, v be a video sequence and $p_t \in [0,1]$ be a probability threshold. Method answer_af$(F, v, p_t, -, -)$ returns $\{(\theta, p) \mid p \geq p_t\}$ independently of the pruning and order methods.*

Theorem 4 (OffPad correctness). *Let F be an activity formula, v be a video sequence and $p_t \in [0,1]$ be a probability threshold. Let F_s, F_e be two frame variables that appear in F. Let S be the set of video sub-sequences returned by OffPad. Then $\forall\, sv \in S$, ($\nexists\, sv' \in S$ s.t. sv' is a subsequence of sv) $\wedge$ ($\exists\, (\theta, p) \in \delta(F)$ s.t. $p \geq p_t$, $F_s\theta = sframe(sv,v)$ and $F_e\theta = eframe(sv,v)$)*[5].

Theorem 5 (OffPad complexity). *Let F be an activity formula, let v be a video sequence and let $p_t \in [0,1]$ be a probability threshold. Let $|F|$ be the number of atoms in F and let O be the set of reference objects. Let (ℓ, pl) be the labeling for the video v; we denote by $|\mathsf{pl}|$ the size of the domain of pl. Then OffPad is running in time $\mathcal{O}(max(|O|,|\ell|,|\mathsf{pl}|)^{2\cdot|F|})$.*

5.4 The OnPad Algorithm

While OffPad is very effective for labeled videos, its recursive search for substitutions cannot be performed unless the full labeling is available when the algorithm is executed. However, in many security surveillance scenarios, activities must be detected as the video is being captured. We now present a real-time algorithm that solves the *Online Identification* problem stated in Section 2.1. OnPad (Algorithm 5) is based on representing formulas and their substitution sets as trees that are incrementally expanded and updated as new labeling information becomes available. We start by defining the tree representation for activity formulas.

Definition 16 (activity formula tree). An activity formula tree is a tuple $T = \langle N, l, v, var, \rightarrow \rangle$, where:

(i) N is the set of nodes.

(ii) $l : N \rightarrow \{A, \neg, \forall, \wedge, \vee\}$ is a function that assigns a label to each node.

(iii) $v : N \rightarrow \mathscr{M}$ is a function that assigns a substitution set to each node.

(iv) $var : \{n \in N | l(N) = \forall\} \rightarrow \mathscr{V}$ is a function that assigns a variable name to all nodes that are labeled with $\forall$.

(v) $\rightarrow$ is a binary relation on N such that $(N, \rightarrow)$ is a tree and $n \in N$ is a leaf node iff $l(n) = A$. The transitive closure of $\rightarrow$ will be denoted by $\rightarrow^*$.

For $n, n' \in N$ such that $n \rightarrow^* n'$, we use $Path(n,n')$ to denote the set of nodes situated on the path from n to n', excluding n, n'. Let $d(n,n') = |Path(n,n')|$. We can easily see that for any activity formula we can construct an activity formula tree in which every leaf node is an atom in the formula; conversely, for any activity formula tree we can easily compute the associated activity formula. We now define an ordering relationship between activity formula trees, which models how an activity formula tree can be expanded when new frames are available for processing.

Definition 17 (activity formula tree ordering). Let $T_1 = \langle N_1, l_1, v_1, var_1 \rightarrow_1 \rangle$, $T_2 = \langle N_2, l_2, v_2, var_2, \rightarrow_2 \rangle$ be two activity formula trees. We write $T_1 \sqsubseteq T_2$ iff there is an injective function $f : N_1 \rightarrow N_2$ such that:

[5] $sframe(sv,v)$ and $eframe(sv,v)$ denote the start and end frames of sv in v.

(i) Paths are preserved, i.e., $\forall\, n_1, n_1' \in N_1$ s.t. $n_1 \rightarrow_1 n_1'$, $f(n_1) \rightarrow_2^* f(n_1')$.

(ii) The substitution set for a leaf node in T_1 is a subset of the substitution set for the corresponding node in T_2, i.e., $\forall\, n_1 \in N_1$ s.t. $l_1(n_1) = A$, $v(n_1) \subseteq v(f(n_1))$.

Intuitively, $T_1 \sqsubseteq T_2$ if T_2 results from modifying T_1 when new labeling elements are added. This suggests that the OnPad algorithm will produce a order-preserving sequence of activity formula trees, one at each step of the algorithm.

Activity formula trees provide an ideal way of representing the current state of OnPad and are updated with new frames and labeling elements. However, the activity formula trees do not provide a direct way of estimating the probability that an activity is about to occur. To provide useful feedback to the user, OnPad returns a numeric *alert level* between 0 and 1 representing an indirect measure of the probability that an activity is about to occur.

Formally, an alert function takes an activity formula tree $T = \langle N, l, v, var, \rightarrow \rangle$ and, based on its structure and/or contents, returns a real number $\mathscr{L} \in [0,1]$, such that:

(i) $\mathscr{L} = 1 \Leftrightarrow v(root((T)) \neq \emptyset$;

(ii) $\forall n \in T, v(n) = \emptyset \Rightarrow \mathscr{L} = 0$.

Example 7. Let $T = \langle N, l, v, var, \rightarrow \rangle$ be an activity formula tree and $S = \{n \in N \mid v(n) \neq \emptyset\}$ be the set of non empty nodes in the tree. The following is an example of *alert level* function, but many others functions are possible. One sanity check is that an alert function returns 1 iff the activity has been fully detected.

$$f_p(T) = \begin{cases} 1, \text{ if } v(root(T)) \neq \emptyset \\ \frac{|S|}{|N|} \text{ otherwise} \end{cases} \tag{1}$$

OnPad takes as input the current state represented by an activity formula tree and the latest frame in the video sequence and returns a level of alert that measures the probability that the activity will occur in the near future. We assume the activity formula has been pre-parsed into an activity formula tree T, which is initialized so that for each node $n \in N$, $v(n) \leftarrow \emptyset$. The algorithm starts by computing the labeling for the frame currently being processed on line 1, and then recomputing the set of all possible substitutions by instantiating variables to the new objects in frame fr. In practice, we have noticed that these operations can often be avoided as frames generally resemble prior frames in a video sequence. Lines 5–12 may add new subtrees to $\forall$-labeled nodes; the subtrees correspond to instances introduced in fr of the same type as the universally-quantified variable. Finally, the algorithm computes the new substitution sets in a bottom-up fashion – the nodes furthest from the root are computed first and then changes are propagated upwards. When this process is completed, T represents the new state of the algorithm and is returned together with the alert level. The following theorems state correctness and complexity results for OnPad. Proofs are omitted for reasons of space.

Theorem 6 (OnPad correctness). *Let F be an event description, let v be a video sequence consisting of frames $[0, fr]$ and let $p_t \in [0,1]$ be a probability threshold. Let $\langle T, \delta \rangle$ be the result of applying OnPad to the frames of v until frame $fr - 1$. Then $root(T) = \delta(F)|_{p_t}$.*

Algorithm 5. OnPad(F, T_s, fr, ℓ, pl, *alert*))

Require: Activity formula F, activity formula tree $T_s = \langle N_s, l_s, v_s, var_s, \rightarrow_s \rangle$ for F representing the current state and the current frame fr. (ℓ, pl) represents the labeling of the video up to the current frame. *alert* is an instance of *alert_level*. We assume the existence of two methods: $toAFTree(F)$, which returns an activity formula tree from an activity formula F and $fromAFTree(T)$ that returns the activity formula corresponding to activity formula tree T.

Ensure: Pair $\langle T, \delta \rangle$, where $\delta \in [0,1]$ representing the alert level and T represents the new state of the incremental algorithm.

1: $(\ell', \mathsf{pl}') \leftarrow$ (**compute** labeling for frame fr)
2: $\mathscr{O}' \leftarrow$ the set of new objects in fr
3: $\Theta \leftarrow$ **recompute** the set of all possible substitutions from $\mathscr{O}'$
4: $T(\langle N, l, v, var, \rightarrow \rangle) \leftarrow T_s$
5: **for all** $n \in T$ s.t. $l(n) = \forall$ **do**
6: $T' \leftarrow$ the subtree of T rooted at n
7: $F \leftarrow fromAFTree(T')$
8: **for all** $o \in \mathscr{O}$ s.t. o has the same type with $var(n)$ **do**
9: $T' \leftarrow toAFTree(F[var(n)/o])$
10: **add** T' as a child of n
11: **end for**
12: **end for**
13: $S \leftarrow \{n \in N | l(n) = A\}$
14: $h \leftarrow max_{n \in S}(d(n, root(T))$
15: $w \leftarrow 0$
16: **while** $w \leq h$ **do**
17: **for all** $n \in N$ s.t. $d(n, root(T)) = h - w$ **do**
18: **if** $l(n) = A$ **then**
19: $\gamma \leftarrow$ **compute** new substitutions for fromAFTree(n) from (ℓ', pl')
20: $v(n) \leftarrow v(n) \cup \gamma$
21: **else if** $l(n) = \wedge$ or $l(n) = \forall$ **then**
22: $v(n) \leftarrow \bigwedge_{m \in \{n' \in N | n' \rightarrow n\}} v(m)$
23: **else if** $l(n) = \neg$ **then**
24: $n' \leftarrow n' \in N$ *s.t.* $n' \rightarrow n$
25: $v(n) \leftarrow \{(\theta, 1-p) | (\theta, p) \in v(n')\} \cup \{(\theta, 1) |$ $\nexists (\sigma, x) \in v(n')$ s.t. $\theta \cup \sigma$ solvable $\}$
26: **else if** $l(n) = \vee$ **then**
27: $v(n) \leftarrow \bigcup_{m \in \{n' \in N | n' \rightarrow n\}} v(m)$
28: **end if**
29: **end for**
30: $w \leftarrow w + 1$
31: **end while**
32: **return** $\langle T, alert(T) \rangle$

Theorem 7 (OnPad complexity). *Let F be an activity formula, let v be a video sequence $[0, fr-1]$, let fr be the current frame and let (ℓ, pl) be the labeling for $[0, fr]$. We denote by O the set of possible objects in the video. Then OnPad is running in time $\mathscr{O}(max(|O|, |\ell|, |\mathsf{pl}|)^{|F|})$.*

6 Experimental Evaluation

In this section, we report the most relevant results of the experimental evaluation of the algorithms proposed in this chapter and refer the reader to [2, 4] for more details.

We used two publicly available datasets – the ITEA CANDELA dataset, a bank surveillance dataset – and a third dataset containing TSA airport tarmac footage. In addition, we evaluated MAGIC on (i) a synthetic dataset of 5 million observations; (ii) a third party dataset consisting of travel information such as hotel reservations, passport and flight information, containing approximately 7.5 million observations.

The **ITEA CANDELA** dataset (http://www.multitel.be/∼va/) consists of 16 videos, about 1 minute in length, depicting package exchanges or people picking up and dropping off packages. We defined 10 activities of increasing complexity and designed the corresponding activity formulas. The **TSA** dataset consists of approximately 118 minutes of tarmac footage. We used a set of 23 activity definition including flight take-off and landing, baggage handling and other maintenance operations. The **Bank** dataset [23] consists of 7 videos, 15–30 seconds in length, depicting staged bank attacks and daily bank operations. Figures 4(a) and 4(b) contain frames from video sequences depicting a staged bank attack and regular bank operations respectively. For the Bank dataset, we used the 5 activities of Example 1 and designed the associated Stochastic Automata, PPNs and activity formulas.

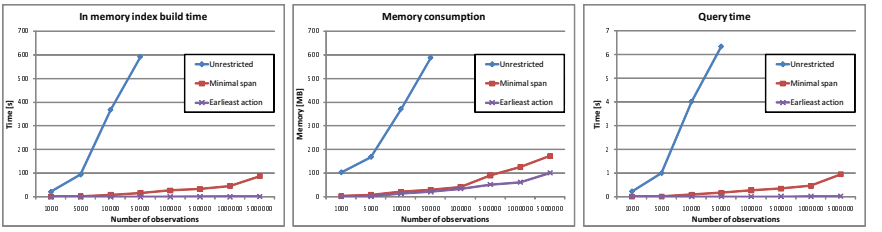
We compared the precision and recall of our algorithms against the ground truth provided by human reviewers as follows. Human reviewers received detailed explanations on the activity models, as well as sets of activity definitions and were asked to mark the starting and ending frame of each activity they encountered in the videos. An average over the reviewers was then used as the ground truth. In order to evaluate OnPad, for each activity definition, at uniformly sampled time points throughout the video, reviewers were asked to provide a number between 0 and 1 representing the likelihood that the activity was about to complete. We considered an average of the alert levels over the reviewers as the ground truth.

6.1 MAGIC

Figures 9.a and 9.b respectively show the time and memory taken to build the index for the synthetic dataset, w.r.t. to number of observations. As expected, the exponential nature of the unrestricted index makes the problem impractical for more than 50,000 observations. Instead, the proposed restrictions yield significantly better results. We also evaluated the average query time on the synthetic dataset. We generated multiple *evidence* and *identification* queries. Each query was run on 10 intervals generated uniformly at random encompassing between 1% and 75% of the data. Each *evidence* query was also run with different thresholds selected uniformly at random. The running times reported in Figure 9.c are an average over the entire set of queries. Query answering times are always below 2 seconds for any restrictions, showing that the MAGIC structure handles activity occurrences efficiently.

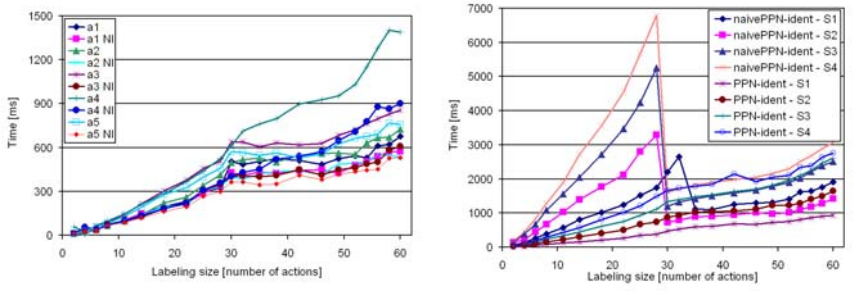
6.2 PPN-evidence, naivePPN-ident and PPN-ident

In the following, we briefly discuss the most significant results for PPN-evidence, naivePPN-ident and PPN-ident. We first measured the running times of the three algorithms for the five activity definitions described above while varying the the number of action symbols in the labeling (Figure 10). It is worth noting that naivePPN-ident exhibits a seemingly strange behavior – running time increases almost exponentially with labeling size, only to drop suddenly at a labeling size of 30. At this labeling size activity definitions first begin to be satisfied by the labeling.



(a) (b) (c)

Fig. 9 MAGIC (a) build time; (b) memory occupancy; and (c) query time

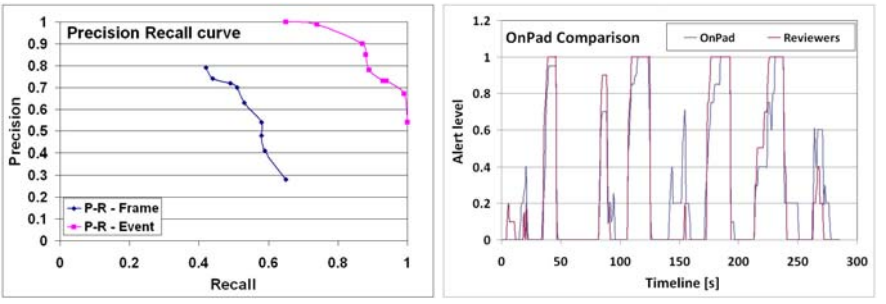


(a) (b)

Fig. 10 Typical execution times for (a) PPN-evidence, (b) naivePPN-ident and PPN-ident.

When no activity definition is satisfied, naivePPN-ident performs several iterations to decrease the threshold until it reaches 0, but it is comparable in terms of running time with PPN-ident when the activity is satisfied by at least one subsequence. Both PPN-evidence and PPN-ident perform linearly with the size of the input.

We then measured the precision and recall of PPN-evidence, naivePPN-ident and PPN-ident w.r.t. the human reviewers. We observed that the minimality condition used by the PPN-evidence algorithm poses an interesting problem – in almost all cases, humans will choose a sequence which is a superset of the minimal subsequence for an activity. In order to have a better understanding of true precision and recall, we compute two sets of measures: recall and precision at the *frame level* (R_f and P_f) and recall and precision at the event (or activity) level (R_e and P_e). At the event level we count as correct any subsequence returned by the algorithm that overlaps with a subsequence returned by a human reviewer. At the frame level we count as correct any frame returned by the algorithm that is also returned by a human reviewer. Details are omitted for reasons of space. However, it is worth mentioning that we observed a surprising behavior for frame recall of PPN-evidence, which appears to increase as the threshold increases. This can be explained considering that, for low thresholds, there is a relatively high number of small candidate subsequences, hence the minimality condition causes *fewer frames* to appear in the answer. This pattern disappears for higher threshold. We also investigated to what



(a) (b)

Fig. 11 (a) Precision/recall of OffPad; (b) Comparison between OnPad and human reviewers

extent the sequences returned by the reviewers and by PPN-evidence actually overlap, and found that overlap is above 70% in 80% of the cases.

6.3 OffPad and OnPad

We first measured the running time of OffPad for the three datasets and observed a high correlation (coefficient 0.95) between the degree of the formula and the time taken to find an answer. We also investigated the impact of the *heuristic_order* and *pruning* methods on the running time of OffPad. Without the *heuristic_order* method, the average running time was 1.35 times greater. Pruning had an even larger effect – running without the pruning method increased the running time 3.5 times, due primarily to the fact that substitutions for frame variables were analyzed even on search paths that could not lead to a viable result. Finally, we observed that running time is at most linear in the size of the labeling. We then evaluated precision and recall of OffPad w.r.t. the ground truth provided by human reviewers. The minimality condition used by OffPad poses the same problem discussed in Section 6.2, thus we computed recall and precision both at the *frame* and event level (Figure 11.a).

W.r.t. to OnPad, we first evaluated the average processing time per frame, and concluded that OnPad can process – in real-time – videos sampled at 4 frames per second. We then looked at how the alert levels returned by OnPad compare to those provided by human reviewers. In brief, the experiments show that OnPad approximates very well the behavior of the reviewers (Figure 11).b. However, we also observed that OnPad tends to be more "conservative" in its alert level while an activity is in its incipient stages, whereas humans tend to have a better intuition for what will happen, even just a few frames after the activity starts.

7 Conclusions and Future Directions

Significant progress has been made in recent years on several aspects of activity detection in videos. However, considerably less effort has been put towards the

definition of a framework where all the aspects of Semantic Video Content Analysis are integrated in a coherent and effective way.

In this chapter, we have presented a framework and a design methodology with the objective of bridging this gap. We have analyzed the typical requirements of a video analysis system from a user's perspective and identified three main classes of problems that such an integrated system should address. We have shown that, based on the nature and complexity of the activities being monitored, different formalisms may be used to model those activities, and different classes of algorithms can be designed to solve the above mentioned problems. Finally, we have shown in real experiments the effectiveness of the proposed design philosophy.

Although our work constitutes a first important step toward a unified framework for Semantic Video Content Analysis, there is still huge room for improvement. As part of the effort to reduce the gap between low-level primitives and semantic activities, we may explore the possibility of pre-processing the output of image processing algorithms and combine primitive actions into 'less than primitive' actions, in order to make the definition of high-level activities more intuitive.

References

1. Akdemir, U., Turaga, P., Chellappa, R.: An ontology based approach for activity recognition from video. In: Proc. of the 16th ACM Intl. Conf. on Multimedia (MM 2008), pp. 709–712 (2008)
2. Albanese, M., Chellappa, R., Moscato, V., Picariello, A., Subrahmanian, V.S., Turaga, P., Udrea, O.: A constrained probabilistic petri net framework for human activity detection in video. IEEE Transactions on Multimedia 10(8), 1429–1443 (2008)
3. Albanese, M., Moscato, V., Picariello, A., Subrahmanian, V.S., Udrea, O.: Detecting stochastically scheduled activities in video. In: Proc. of the 20th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2007), pp. 1802–1807 (2007)
4. Albanese, M., Pugliese, A., Subrahmanian, V.S., Udrea, O.: MAGIC: A multi-activity graph index for activity detection. In: Proc. of the IEEE Intl. Conf. on Information Reuse and Integration (IRI 2007), pp. 267–278 (2007)
5. Avrahami-Zilberbrand, D., Kaminka, G., Zarosim, H.: Fast and complete symbolic plan recognition: Allowing for duration, interleaved execution, and lossy observations. In: Proc. of the AAAI Workshop on Modeling Others from Observations, MOO 2005 (2005)
6. Chen, D., Yang, J., Wactlar, H.D.: Towards automatic analysis of social interaction patterns in a nursing home environment from video. In: Proc. of the 6th ACM SIGMM Intl. Workshop on Multimedia Information Retrieval (MIR 2004), pp. 283–290 (2004)
7. Chen, S.M., Ke, J.S., Chang, J.F.: Knowledge representation using fuzzy petri nets. IEEE Transactions on Knowledge and Data Engineering 2(3), 311–319 (1990)
8. David, R., Alla, H.: Petri nets for modeling of dynamic systems a survey. Automatica 30(2), 175–202 (1994)
9. Duong, T.V., Bui, H.H., Phung, D.Q., Venkatesh, S.: Activity recognition and abnormality detection with the switching hidden semi-markov model. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2005), vol. 1, pp. 838–845 (2005)
10. Georis, B., Maziere, M., Brémond, F., Thonnat, M.: A video interpretation platform applied to bank agency monitoring. In: IEE Intelligent Distributed Surveilliance Systems (IDSS-04), pp. 46–50 (2004)

11. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Intl. Journal of Human-Computer Studies 43(5-6), 907–928 (1995)
12. Guler, S., Burns, J.B., Hakeem, A., Sheikh, Y., Shah, M., Thonnat, M., Bremond, F., Maillot, N., Vu, T.V., Haritaoglu, I., Chellappa, R., Akdemir, U., Davis, L.: An ontology of video events in the physical security and surveillance domain (2004), http://www.ai.sri.com/~burns/EventOntology/PhysicalSecurity1-30-2004.doc
13. Hakeem, A., Shah, M.: Ontology and taxonomy collaborated framework for meeting classification. In: Proc. of the 17th Intl. Conf. on Pattern Recognition (ICPR 2004), vol. 4, pp. 219–222. IEEE Computer Society, Los Alamitos (2004)
14. Hamid, R., Huang, Y., Essa, I.: ARGMode – activity recognition using graphical models. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW 2003), vol. 4, pp. 38–43 (2003)
15. Hobbs, J., Nevatia, R., Bolles, B.: An ontology for video event representation. In: Proc. of the 2004 IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW 2004), p. 119 (2004)
16. Luhr, S., Bui, H.H., Venkatesh, S., West, G.A.W.: Recognition of human activity through hierarchical stochastic learning. In: Proc. of the First IEEE Intl. Conf. on Pervasive Computing and Communications (PCC 2003), pp. 416–422 (2003)
17. Marhasev, E., Hadad, M., Kaminka, G.A.: Non-stationary hidden semi-markov models in activity recognition. In: Proc. of the AAAI Workshop on Modeling Others from Observations, MOO 2006 (2006)
18. Marsan, M.A., Balbo, G., Chiola, G., Conte, G., Donatelli, S., Franceschinis, G.: An introduction to generalized stochastic petri nets. Microelectronics and Reliability 31(4), 699–725 (1991)
19. Murata, T.: Petri nets: Properties, analysis and applications. Proc. of the IEEE 77(4), 541–580 (1989) (1989)
20. Nevatia, R., Zhao, T., Hongeng, S.: Hierarchical language-based representation of events in video streams. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW 2003), vol. 4 (2003)
21. Petri, C.A.: Communication with automata. DTIC Research Report AD0630125 (1966)
22. Shoenfield, J.R.: Mathematical Logic. Addison Wesley, Reading (1967)
23. Vu, V.T., Brémond, F., Thonnat, M.: Automatic video interpretation: A novel algorithm for temporal scenario recognition. In: Proc. of the 18th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 1295–1302 (2003)