A Constrained Probabilistic Petri Net Framework for Human Activity Detection in Video

Massimiliano Albanese, Rama Chellappa, *Fellow, IEEE*
Vincenzo Moscato, Antonio Picariello, V.S. Subrahmanian, Pavan Turaga *****, *Student Member*
and Octavian Udrea, *Student Member*

***Abstract*— EDICS code: 4-KEEP, 2-ALAR. Recognition of human activities in restricted settings such as airports, parking lots and banks is of significant interest in security and automated surveillance systems. In such settings, data is usually in the form of surveillance videos with wide variation in quality and granularity. Interpretation and identification of human activities requires an activity model that is a) rich enough to handle complex multi-agent interactions, b) is robust to uncertainty in low-level processing and c) can handle ambiguities in the unfolding of activities. Efficient inference algorithms based on the activity model then provide means to accomplish the tasks of activity recognition and anomaly detection. Toward this end, we present a computational framework for human activity representation based on Petri nets. We propose an extension – Probabilistic Petri Nets (PPN)- and show how this model is well suited to address each of the above requirements in a wide variety of settings. Then, we focus on answering two types of questions: (i) what are the minimal sub-videos in which a given activity is identified with a probability above a certain threshold and (ii) for a given video, which activity from a given set occurred with the highest probability ? We provide the PPN-MPS algorithm for the first problem, as well as two different algorithms (naivePPN-MPA and PPN-MPA) to solve the second. Our experimental results on a dataset consisting of bank surveillance videos and an unconstrained TSA tarmac surveillance dataset show that our algorithms are both fast and provide high quality results.**

I. Introduction

The task of designing algorithms that can analyze human activities in video sequences has been an active field of research during the past ten years. Nevertheless, we are still far from a systematic solution to the problem. The analysis of activities performed by humans in restricted settings is of great importance in applications such as automated security and surveillance systems. There has been significant interest in this area where the challenge is to automatically recognize the activities occurring in a camera's field of view and detect abnormalities. The practical applications of such a system could include airport tarmac monitoring, monitoring of activities in secure installations, surveillance in parking lots etc. The difficulty of the problem is compounded by several factors - a) at the lower-level, unambiguous detection of primitive actions can be challenging due to changes in illumination, occlusions, noise etc. and b) at the higher-level, activities in such settings are usually characterized by complex multi-agent interaction, and c) the same semantic activity can be performed in several variations which may not conform to strict statistical or syntactic constraints.

In real life scenarios, one usually has some prior knowledge of the type of activities that occur in a given domain and their corresponding semantic structure. In addition, one may also have access to a limited training set, which in most cases is not exhaustive. Thus, there is a need for approaches that can leverage available domain knowledge to design semantic activity models and yet retain the robustness of the statistical approaches. The fusion of these disparate approaches – statistical, structural and knowledge based – has not yet been fully realized and has only been gaining attention in recent years. Our approach falls in this category that combines these approaches – where we exploit domain knowledge to create rich and expressive models for activities based on the Petri net formalism, and augment the standard Petri net model with a probabilistic extension to handle (a) 'noise' in the labeling of video data, (b) inaccuracies in vision algorithms and (c) allow for departures from a hard-coded activity model.

Based on this activity model, given a segment s of a video, we can then define a probability that the segment s contains an instance of the activity in question. We then provide fast algorithms to answer two kinds of queries. The first type of query tries to find *minimal* (i.e. the shortest possible) clips of video that contain a given event with a probability exceeding a specified threshold — we call these *Threshold Activity Queries*. The second type of query takes a portion of a video (the part seen thus far) and a set of activities, and tries to find the activity that most likely occurred in the video — we call these *Activity Recognition Queries.*

A. Prior Related Work

The study and analysis of human activities has a wide and rich literature. On the one hand there has been significant effort to build algorithms that can learn patterns from training data - which can be broadly classified as statistical techniques - and on the other hand, there are the model-based or knowledge-based approaches represent activities at a higher level of abstraction. A recent survey of computational approaches for activity modeling and recognition may be found in [1].

Statistical approaches: A large body of work in the activity recognition area builds on statistical pattern recognition

Authors listed in alphabetical order. V. Moscato and A. Picariello are with the Università di Napoli, Federico II, Napoli, Italy, Email: $\{vmoscato, picus\}@unina.it$. Massimiliano Albanese, Rama Chellappa, V.S. Subrahmanian, Pavan Turaga and Octavian Udrea are with the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Email: $\{albanese, rama, vs, pturaga, udrea\}@umiacs.umd.edu$.

methods. Hidden Markov Models (HMMs) [2], [3] and its extensions such as Coupled Hidden Markov Models (CHMMs) [4] have been used for activity recognition. Bobick and Davis [5] developed a method for characterizing human actions based on the concept of "temporal templates". Kendall's shape theory has been used to model the interactions of a group of people and objects by Vaswani et al [6]. Statistical approaches such as these require a training phase in which models are learnt from training exemplars. This inherently assumes the existence of a rich training set which encompasses all possible variations in which an activity may be performed. But, in real life situations, one is not provided with an exhaustive training set. Moreover, it is not possible to anticipate the various ways an activity may be performed. Statistical techniques do not model the semantics of activities and may fail to recognize the same activity performed in a different manner. They are also not well suited to model the complex behaviors such as concurrence, synchronization and multiple instances of activities.

Structural approaches: Unlike the statistical methods which assume the availability of a training set, these methods usually rely on hand-crafted models from experts or analysts. Since the models are hand-crafted, they are often intuitive and can be related to the semantic structure of the activity. A method for generating high-level descriptions of traffic scenes was implemented by Huang et al. using a dynamic Bayes networks (DBN) [7], [8]. Hongeng and Nevatia [9] proposed a method for recognizing events involving multiple objects using temporal logic networks. Context free grammars have been used by [10] to recognize sequences of discrete events. Ivanov and Bobick [11] used Stochastic Context Free Grammars (SCFG) to represent human activities in surveillance videos. In an approach related to that presented in this paper, Albanese et al. [12] use stochastic automata to model activities and provide algorithms to find subvideos matching the activities. Shet, Harwood and Davis [13] have proposed using a Prolog based reasoning engine to model and recognize activities in video. Castel, Chaudron and Tessier [14] developed a system for high-level interpretation of image sequences based on deterministic Petri nets. Deterministic Petri nets have also been used by Ghanem et al [15] as a tool for querying of surveillance videos. Though structural methods have distinct advantages over statistical techniques in terms of representative and expressive power - CFG's, DBN's, logic-based methods etc do not easily model concurrence, synchronization and multiple instances of activities. To address these issues, we use the Petri net formalism to represent activities. Further, using a probabilistic extension to the standard Petri net model, we show how we can integrate high-level reasoning approaches with the inaccuracies inherent in the low-level processing.

Hybrid approaches: Techniques that can leverage the richness of structural approaches, incorporate domain knowledge from experts or knowledge bases and also retain the robustness of statistical approaches have been gaining more attention in recent years. Ivanov and Bobick [11] use stochastic context free grammars (SCFG) that is better able to handle inaccuracies in lower level modules and variations in activities. SCFGs have also been used by Moore and Essa [16] to recognize activities involving multiple people, objects, and actions. Attribute grammars have also been used to combine syntactic and statistical models for visual surveillance applications in [17]. Though grammar based approaches present a sound theoretical tool for behavior modeling, in practice, the specification of the grammar itself becomes difficult for complex activities. Further, it is difficult to model behaviors such as concurrency, synchronization, resource sharing etc using grammars. Petri nets on the other hand are well suited for these tasks and have been traditionally used in the field of compiler design and hybrid systems. A probabilistic version of Petri nets based on particle filters was presented by Lesire and Tessier [18] for flight monitoring. Possibilistic Petri nets which combine possibility logic with Petri nets have been used for modeling of manufacturing systems by [19]. In computer vision, activity recognition using Petri nets has been investigated by Ghanem et al [15] and Castel et al [14]. However, they do not deal with probabilistic modeling and detection, instead relying on perfect accuracy in the lower levels.

It is important to note that the aforementioned papers do not explicitly address the following problems: (i) developing efficient algorithms to find all minimal segments of a video that contain a given activity with a probability exceeding a given threshold and (ii) developing efficient algorithms to find the activity that is most likely (from a given set of activities) in a given video segment.

Contributions: First, we introduce a probabilistic Petri Net representation of complex activities based on atomic actions which are recognizable by image understanding algorithms. The next major contribution is the PPN-MPS[1] algorithm to answer threshold queries. Our third major contribution is the naivePPN-MPA and PPN-MPA[2] algorithms to solve activity recognition queries. We evaluate these algorithms theoretically as well as experimentally (in Section VI) on two third party datasets consisting of staged bank robbery videos [20] and unconstrained airport tarmac surveillance videos and provide evidence that the algorithms are both efficient and yield high-quality results.

Organization of paper: First, in Section II we define probabilistic Petri nets (PPN) and discuss representation of complex activities using the PPN framework. Then, we present efficient algorithms to solve the Most Probable Subsequence (MPS) problem in Section III, and the Most Probable Activity (MPA) problem in Section IV. In Section V, we present details of the system implementation. In Section VI, we present experimental results on two third party datasets consisting of staged bank robbery videos and unconstrained airport tarmac surveillance videos. Finally in Section VII, we present concluding remarks.

II. Modeling Activities as Probabilistic Petri Nets

Petri Nets were defined by Carl Adam Petri as a mathematical tool for describing relations between conditions and events. Petri Nets are particularly useful to model and visualize behaviors such as sequencing, concurrency, synchronization

[1]Probabilistic Petri Net – Most Probable Subsequence.
[2]Probabilistic Petri Net – Most Probable Activity.

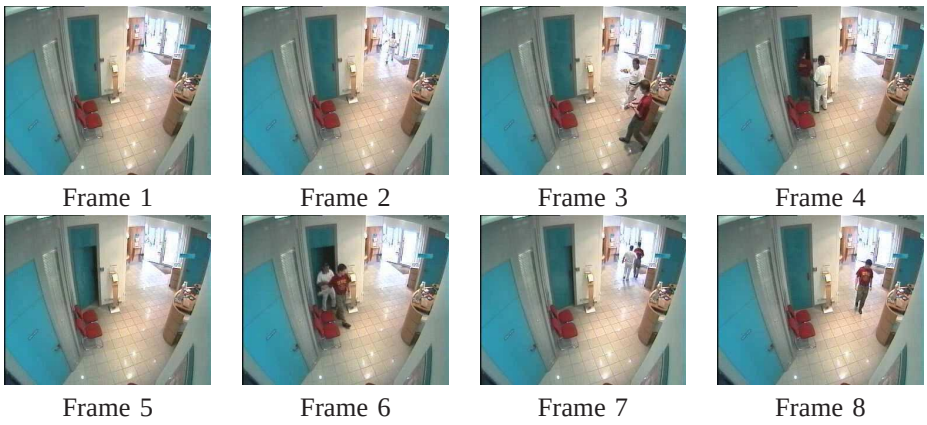


Fig. 1. Example video sequence of a simulated bank attack.

and resource sharing. We refer the reader to David et al [21] and Murata [22] for a comprehensive survey on Petri nets. Several forms of Petri nets (PN's) such as Colored PN's, Continuous PN's, Stochastic timed PN's, Fuzzy PN's etc have been proposed. Colored PN's associate a *color* to each token, hence are useful to model complex systems where each token can potentially have a distinct identity. In Continuous PN's, the markings of places are real numbers and not just integers. Hence, they can be used to model situations where the underlying physical processes are continuous in nature. Stochastic timed Petri nets [23] associate, to each transition, a probability distribution which represents the delay between the enabling and firing of the transition. Fuzzy Petri nets [24] are used to represent fuzzy rules between propositions.

In real-life situations, vision-based systems have to deal with ambiguities and inaccuracies in the lower-level detection and tracking systems. Moreover, activities may not unfold exactly in the sequence as laid out by the model that represents it. The aforementioned versions of Petri nets (continuous, stochastic, fuzzy) are not well suited to deal with these situations. The proposed probabilistic Petri net model is well-suited to express uncertainty in the state of a token or associate a probability to a particular unfolding of the PN.

A. Labeling of Video Sequences

There has been significant effort in computer vision to address the problem of detecting various types of elementary events or actions from video data, such as a person entering or leaving a room or unloading a package. Corresponding to the set of atomic or primitive actions, we assume the existence of a finite set $\mathcal{A}$ of *action symbols*. Image understanding algorithms[3] are used to detected these atomic actions. In general, the action symbols can be arbitrary logical predicates $a : \mathcal{O}^k \rightarrow \{true, false\}$, where $\mathcal{O}$ is the set of objects in the video and k is the arity of the action symbol a. To provide a few examples used for the TSA dataset, $pickup(P : Person, O : Object)$ returns the value true if person P has picked up object O in the current frame and false otherwise; similarly, $occludes(O_1 : Object, O_2 : Object)$ returns true if and only if O_1 occludes O_2 in the current frame[4].

For the remainder of the discussion, without loss of generality [5], we will assume that action symbols are propositional in nature. The image processing library associates a subset of $\mathcal{A}$ with every frame (or block of frames) in the video sequence. Formally, a labeling is a function $\ell : v \rightarrow 2^{\mathcal{A}}$, where v is a video sequence and $2^{\mathcal{A}}$ is the power set of $\mathcal{A}$. For example, consider the labeling process on the video sequence in Figure 1.

Example 1: Let $\mathcal{A} = \{outsider - present, employee - present, outsider - absent, employee - absent, outsider - insafezone, employee - insafezone\}$. The labeling of the video sequence in Figure 1 w.r.t. $\mathcal{A}$ is given by:
$\ell(frame1) = \{outsider - absent, employee - absent\}$
$\ell(frame2) = \{outsider - present, employee - absent\}$
$\ell(frame3) = \{outsider - present, employee - present\}$
$\ell(frame4) = \{outsider - insafezone, employee - insafezone\}$
$\ell(frame5) = \{outsider - absent, employee - absent\}$
$\ell(frame6) = \{outsider - insafezone, employee - insafezone\}$
$\ell(frame7) = \{outsider - present, employee - present\}$
$\ell(frame8) = \{outsider - absent, employee - present\}$
In this example, the presence of the $outsider - present$ label for frame 2 means that an outsider (a person who is not an employee) is present in that frame. Similarly, $employee - present$ in frame 3 means a person whose face is recognized as that of an employee is present in frame 3, whereas $employee - insafezone$ means an employee is near the bank safe (on the left-hand side of the image).

B. Constrained Probabilistic Petri Nets

We define a constrained probabilistic Petri net (PPN) as the tuple $PPN = \{P, T, \rightarrow, F, \delta\}$ where

1) P and T are finite disjoint sets of places and transitions respectively i.e. $P \cap T = \emptyset$.
2) $\rightarrow$ is the flow relation between places and transitions, i.e., $\rightarrow \subseteq (P \times T) \cup (T \times P)$.

[3]Although some of the image understanding methods cited in this paper return a numeric value in the unit interval $[0, 1]$, this can be readily transformed into a method that returns either "yes" or "no" based on a fixed threshold.

[4]According to the occlusion detection algorithm.

[5]For a finite object domain, we can "flatten" predicates of any arity to the propositional case.

3) The preset of a node $x \in P \cup T$ is the set $\{y | y \rightarrow x\}$. This set is denoted by the symbol ${}^{\cdot}x$.
4) The postset of a node $x \in P \cup T$ is the set $\{y | x \rightarrow y\}$. This set is denoted by the symbol $x^{\cdot}$.
5) F is a function defined over the set of places P, that associates to each place x a local probability distribution over $\{t | x \rightarrow t\}$. $F : x \longmapsto f_x(t)$, where $x \in P$ and $f_x(t)$ is a probability distribution defined over the postset of x. For each place $x \in P$ in a given PPN, 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 $f_x(t)$.
6) $\delta : T \rightarrow 2^{\mathcal{A}}$ associates a set of action symbols with every transition in the Petri Net. Intuitively, this imposes constraints on the transitions in the Petri Net – transitions will only fire when all the action symbols in the constraint are also encountered in the video sequence labeling.
7) $\exists x \in P$ s.t. $x^{\cdot} = \emptyset$. This means that there exists at least one terminal node in the Petri net.
8) $\exists x \in P$ s.t. ${}^{\cdot}x = \emptyset$. This means that there exists at least one start node in the Petri net.

C. Tokens and Firing of transitions

The dynamics of a Petri net are represented by means of 'markings'. For a PPN $(P, T, \rightarrow, F, \delta)$, a *marking* is a function $\mu : P \rightarrow \mathcal{N}$ that assigns a number of *tokens* to each place in the PPN. For example, $\mu(p_1) = 2$ means that according to marking μ, there are two tokens in place p_1. For a PPN representing an activity, a marking μ represents the current state of completion of that activity. The token is simply an abstract representation of a partial state.

The execution of a Petri net is controlled by its current marking. A transition is said to be *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 will denote by μ_0 the *initial marking* of a PPN. As transitions fire, the marking changes for the places in the preset and postset of the transition being fired. A *terminal marking* is reached whenever 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 2: Consider an example of a car pickup activity represented by a probabilistic Petri Net as shown in figure 2. In this figure, the places are labeled $p_1, \ldots, p_5$ and transitions $t_1, \ldots, t_6$. We denote by μ_0 the initial marking depicted in the figure in which all places have 0 tokens. In this PN, p_1 and p_3 are the start nodes and p_5 is the terminal node. When a car enters the scene, a token is placed in place p_1 and we have a new marking μ_1 such that $\mu_1(p_1) = 1$ and $\mu_1(p) = 0$ for any $p \neq p_1$. The transition t_1 is enabled in this state, but it cannot fire until the condition associated with it is satisfied – i.e., when the car stops near a parking slot. When this occurs,

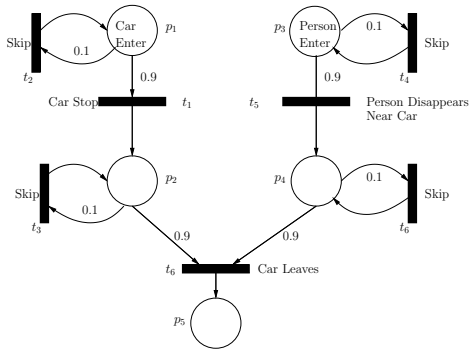


Fig. 2. Probabilistic Petri net for Car-pickup

the token is removed from p_1 and placed in p_2. This leads to a new marking μ_2 such that $\mu_2(p_2) = 1$ and $\mu_2(p) = 0$ for all $p \neq p_2$. Similarly, when a person enters the parking lot, a token is placed in p_3 and transition t_5 fires after the person disappears near the parked car. The token is then removed from p_3 and placed in p_4. Now, with a token in each of the enabling places of transition t_6, it is ready to fire when the associated condition i.e. car leaving the parking lot is satisfied. Once the car leaves, t_6 fires and both the tokens are removed and a token placed in the final place p_5. This example illustrates sequencing, concurrency and synchronization.

In the above discussion, we have not yet discussed the probabilities labeling some of the place-transition edges. Note that the postsets of p_1, p_2, p_3, p_4 contain multiple transitions. These 'skip' transitions are used to *explain away* deviations from the base activity pattern – each such deviation is penalized by a low probability. For example, after the car enters the scene, suppose that instead of stopping in a lot, the car goes around in circles. In this case, the skip transition is fired, meaning a token is taken from p_1 and immediately placed back. However, the *token probability* is decreased by the probability labeling the edge to the skip transition.

All tokens are initially assigned a probability score 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 – for instance when t_6 fires – then the net 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.

Example 3: Figure 3 depicts another constrained PPN modeling two possible ways in which a successful bank robbery may happen. Uncertainty appears after both the employee and outsider are inside the zone of the bank safe (at place p_5). According to the model, in 60% of the cases both the attacker and the employee will enter the safe (visually, this means that they will disappear from view for a period of time) – this is modeled starting with p_7, whereas in 35% of the cases, the attacker enters the safe alone (p_6). In either case, a successful robbery is completed once the outsider leaves the scene (p_{12}). Note the use of skip transitions to model possible "noise" in the labeling (i.e. labels that do not correspond to any of the currently enabled transitions) as explained previously.

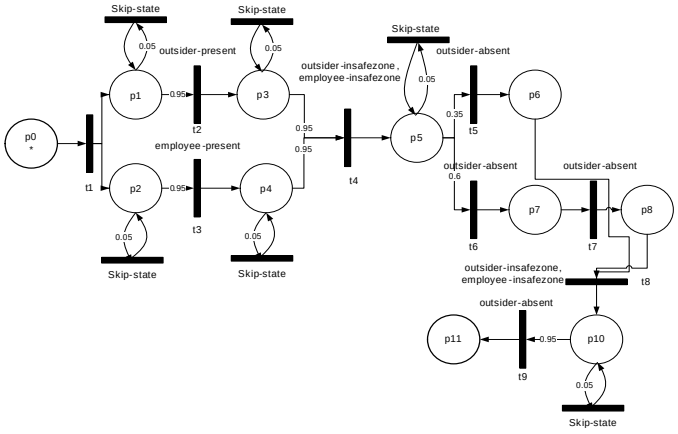


Fig. 3. Constrained PPN depicting a bank attack

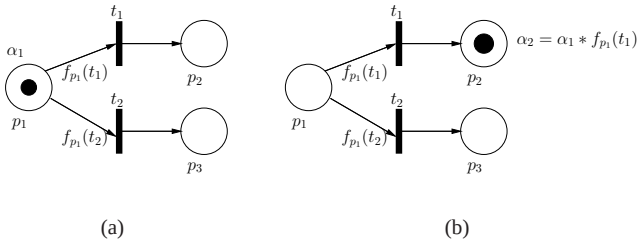


Fig. 4. (a) Before Firing (b) After firing of transition t_1

D. Rules for token probability computation

We denote by α_k the value or probability associated with a token at place p_k. When a token is placed in one of the start nodes, it is associated with a value of 1 (or some other prior probability for instance, as reported by the low-level vision module). Figure 4 shows how the value associated with a token gets updated as a transition fires. In this figure, transitions t_1 and t_2 are both enabled. f_{p_1} is the local probability distribution associated with the place p_1. If transition t_1 fires, then the value of the token in place p_2 becomes $\alpha_2 = \alpha_1 * f_{p_1}(t_1)$ as shown in figure 4. The probability computation rules for synchronization and concurrency are illustrated in figures 5 and 6 respectively.

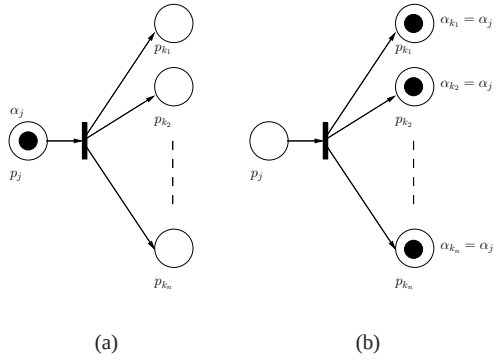


Fig. 5. Concurrency (a) Before Firing (b) After firing

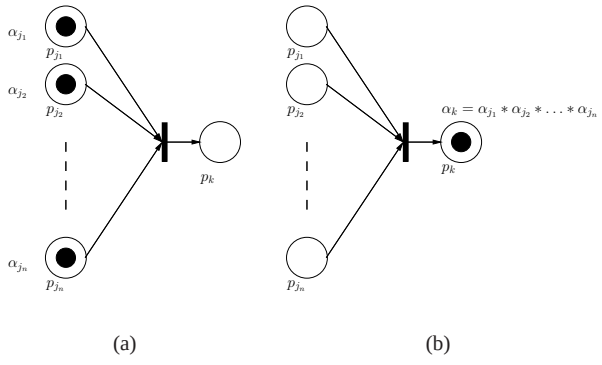


Fig. 6. Synchronization (a) Before Firing (b) After firing

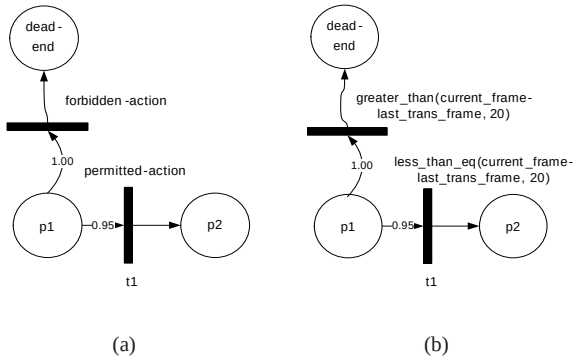


Fig. 7. (a) Forbidden actions (b) Temporal constraints

E. Enhancing PPN expressiveness: Forbidden actions and Temporal durations

In this section, we will describe how PPNs can be used to model "forbidden actions" – actions which trigger the recognition of a partially completed activity to be nullified – and constraints on the temporal duration of the activity.

Forbidden Actions: In several applications, it is important to be alert to suspicious behavior that should immediately trigger an alarm. We show how forbidden actions can be modeled in Figure 7(a). This is possible when the set of forbidden actions is mutually exclusive to the set of permitted actions. Essentially, for each forbidden action f_a that may occur in a given state, we add a transition with probability 1 conditioned on f_a to a place that is a dead-end (i.e., no terminal marking can be reached from it).

Temporal Duration: Oftentimes activities are constrained not just by a simple sequencing of action primitives, but also by the duration of each action primitive in the sequence. In general, we may associate a probability distribution over the 'dwell' time to each place in the Petri net. We have used a simplified version where we constrain the dwell time to be upper and lower bounded. We have built a few macros that handle frame computation. For instance, $current\_frame$ is the number of the current frame, whereas $last\_trans\_frame$ is the number of the last frame at which a transition was fired for the current activity. Intuitively, Figure 7(b) imposes the condition that the transition t_1 fire within twenty frames of the last transition that was fired. If this does not occur, then we again transition to a dead-end, as in the case of forbidden actions.

F. Activity recognition

We now define a 'PPN trace' and what it means for a video sequence to satisfy a given PPN and provide a method for computing the probability with which an activity is satisfied.

Definition 1 (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 $p_i = \Pi_{\{x \in P | x \rightarrow t\}} F(x, t)$. We say the trace $\langle t_1, \ldots, t_k \rangle$ has probability $p = \Pi_{i \in [1,k]} p_i$. 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* $\frac{p}{p_{max}}$.

Example 4: Consider the PPN in Figure 3. $\langle t_1, t_3, t_2, t_4, t_5, t_8, t_9 \rangle$ is a trace with probability $.2708$ and relative probability $.5833$. $\langle t_1, t_2, t_3, t_4, t_6, t_7, t_8, t_9 \rangle$ is a trace with probability $.4642$ and a relative probability of 1.

Definition 2 (Activity satisfaction): Let $\mathcal{P} = (P, T, \rightarrow, F, \delta)$ be a PPN, let v be a video sequence and let ℓ be the labeling of v. We say ℓ satisfies $\mathcal{P}$ with *relative probability* $p \in [0, 1]$ iff there exists a trace $\langle t_1, \ldots, t_k \rangle$ for $\mathcal{P}$ 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. For reasons of simplicity, 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'.

Example 5: Consider the video sequence in Figure 1 with the labeling in Example 1, and the activity PPN in Figure 3. The labeling in Example 1 satisfies the PPN with relative probability 1 since the conditions for the trace $\langle t_1, t_2, t_3, t_4, t_6, t_7, t_8, t_9 \rangle$ are subsets of the labeling for frames 1 – 8.

We now define the two types of queries we are interested in answering.

The Threshold Activity Query Problem (PPN-MPS) can be formulated as following: given a constrained PPN, a video sequence v and a probability threshold p_t, find the minimal subsequences of v that satisfy the PPN with probability at least p_t. Intuitively, such queries are of interest for identifying specific portions of the video sequence in which a given activity occurs.

The Activity Recognition Query Problem (PPN-MPA) is formulated as follows: Given a set of activities $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ and a video sequence v, determine which activity (or set of activities) is satisfied by the labeling of v with maximum probability. Intuitively, such queries are useful in scenarios in which video sequences are processed semi-automatically and we are interested in finding which activity occurs in a given sequence.

III. Algorithms for Threshold Activity Queries

We start this section by presenting an efficient algorithm for the PPN-MPS problem. We then show how an iterative variant of PPN-MPS can be used to naively solve the PPN-MPA problem in a naive fashion – we call this the naivePPN-MPA. Finally, we provide a greatly improved version of the PPN-MPA algorithm.

The PPN-MPS algorithm (Algorithm 1) is based on the process of simulating the constrained PPN forward. The algorithm uses a separate 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 the marking μ and f_s is the frame when the first transition in that trace was fired. The PPN-MPS algorithm takes advantage of the fact that we only need to return minimal subsequences and not the entire PPN traces that lead to them, hence using the last marking as a state variable is sufficient for our purposes.

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), with the 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[6], and at each iteration analyzes the current candidates stored in S. Any transitions t enabled in the current marking from S 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 value p' (lines 13 – 14) 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 the store – nor should we at this time, since we also need to fire any skip transitions that are available. This is done for any enabled skip transition on line 26. At this point we also prune away any states in S that have no possibility of remaining above the threshold as we reach a terminal marking (line 27).

Example 6: We illustrate the working of PPN-MPS on the PPN in Figure 3, the labeling in Example 1 and $p_t = 0.5$. Initially, $S = \langle \mu_0, 1, 0 \rangle$. We will denote by μ_i the marking obtained after firing t_i in Figure 3. At frame 1, since transition t_1 has no precondition, it will fire and we will have one new element in S (along with the existing initial state) – $\langle \mu_1, 1, 0 \rangle$. Note that the starting frame is still set to 0 – this is done so that the start of the subsequence is marked only when a labeling condition is satisfied. At frame 2, transition t_2 fires and we add $\langle \mu_2, .95, 2 \rangle$ to S. At this point, transition t_3 is enabled but cannot fire because its condition $employee - present$ is not satisfied. Hence a skip transition fires, but that generates a state with probability $.05$, which is under the threshold and hence thrown away immediately. We will ignore the firing of skip transitions for the rest of the example, as they are immediately pruned away due to the low probabilities generated. Transition t_3 does fire at frame 3 and adds $\langle \mu_3, .9025, 1 \rangle$ to S. Transition t_4 is now enabled and its conditions satisfied at frame 4, hence it is fired and a new state $\langle \mu_4, .8145, 1 \rangle$ is generated. At frame

[6]In our implementation, we group frames with identical labeling into blocks and iterate over blocks instead which leads to enhanced efficiency.

Algorithm 1 Threshold Activity Queries (PPN-MPS)

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

1: $S \leftarrow \{\langle \mu_0, 1, 0\rangle\}$
2: $R \leftarrow \emptyset$
3: $max_p \leftarrow$ the maximum probability for any trace of PPN
4: **for all** f frame in video sequence **do**
5: **for all** $\langle \mu, p, f_s\rangle \in S$ **do**
6: **for all** $t \in T$ enabled in μ s.t. $\delta(t) \subseteq \ell(f)$ **do**
7: $\mu' \leftarrow$ fire transition t for marking μ
8: **if** $f_s = 0$ and $\delta(t) \neq \emptyset$ **then**
9: $f' \leftarrow f$
10: **else**
11: $f' \leftarrow f_s$
12: **end if**
13: $p^* \leftarrow \Pi_{\{x \in P | x \rightarrow t\}} F(x, t)$
14: $p' \leftarrow p \cdot p^*$
15: **if** $p' \geq p_t \cdot max_p$ **then**
16: **if** $\forall x \in P$ s.t. $\mu(x) < \mu'(x)$ it holds that $p' \cdot p^*(x) \geq p_t \cdot max_p$ **then**
17: $S \leftarrow S \cup \{\langle \mu', p', f'\rangle\}$
18: **end if**
19: **end if**
20: **if** μ' is a terminal configuration **then**
21: $S \leftarrow S - \{\langle \mu', p', f'\rangle\}$
22: $R \leftarrow R \cup \{[f_s, f]\}$
23: **end if**
24: **end for**
25: **for all** $t \in T$ enabled **do**
26: Fire skip transitions if no other transitions fired and update $\langle \mu, p, f_s\rangle$
27: Prune $\langle \mu, p, f_s\rangle \in S$ s.t. $\forall x \in P$ s.t. $\mu(x) > 0, p \cdot p^*(x) < p_t \cdot max_p$ {Pruning always keeps $\langle \mu_0, 1, 0\rangle$ in S}
28: **end for**
29: **end for**
30: **end for**
31: Eliminate non-minimal subsequences in R
32: **return** R

5, transition t_6 is satisfied (and t_5 is not), hence it is fired and the state becomes $\langle \mu_5, .4880, 1\rangle$. Finally, t_7, t_8, t_9 are fired at frames 6, 7, 8 and bring the final state to $\mu_8, .4642, 1\rangle$, leading to a relative probability of 1 (we remind the reader that the elements of S hold the absolute probability).

The following theorem states that the PPN-MPS algorithm returns the correct result.

Theorem 1 (PPN-MPS correctness): Let $\mathcal{P} = (P, T, \rightarrow, F, \delta)$ 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 $\mathcal{P}$ with probability greater than or equal to p_t, one of the following holds:

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

Proof Let $v_0 \subseteq v$ be a subsequence of v that satisfies the activity definition, but is not returned by PPN-MPS. If v_0 has a subsequence returned by PPN-MPS, we are done. Otherwise, since v_0 satisfies the activity definition, according to Definition 2, there exists a trace $\langle t_1, \ldots, t_n\rangle$ and corresponding frames (in v_0) $f_1 \leq \ldots \leq f_k$ such that $\forall i \in [1, k]$, $\delta(t_i) \subseteq \ell(f_i)$. We will show by induction that the algorithm will fire all transitions $(t_i)_{i\in[1,n]}$. Since $\langle \mu_0, 1, 0\rangle$ always remains in S, and t_1 is executable from μ_0 and $\delta(t_1) \subseteq \ell(f_1)$, then t_1 must be fired on line 7 and f_1 becomes the current frame (*initial condition*). Assume that all transitions up to $(t_i)_{i\in[1,k]}$ have been fired; we want to show that t_k must fire next. Assume that the new state in S induced by the firing of t_{k-1} is $s_{k-1} = \langle \mu_{k-1}, p_{k-1}, f_1\rangle$ such that t_k is enabled in μ_{k-1}. Since $(t_i)_{i\in[1,k]}$ is a trace, then when frame f_k is analyzed, there must exist a state $s_k = \langle \mu_{k-1}, p, f_1\rangle \in S$ (if multiple such states exist, then we choose the one with the highest probability)[7]. Let p_k be the probability of firing t_k (which is enabled and has $\delta(t_k)$ satisfied on line 6) and let p_0 be the probability of the trace. Then $p \cdot p_k \geq \frac{p_0}{max_p} \geq p_t$, (condition on line 15 holds). Also, for all $x \in P$ that receive a token as t_k fires, $p \cdot p_k \cdot p^*(x) \geq \frac{p_0}{max_p} \geq p_t$ (condition on line 16 holds); line 17 therefore executes and the new marking μ_k (including the effects of firing t_k) is added to S (*induction step*). We showed by induction that a trace of probability greater than or equal to p_t corresponding to a subsequence of v_0 has been executed.

Theorem 2 (PPN-MPS complexity): Let $\mathcal{P} = (P, T, \rightarrow, F, \delta)$ 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[8]. Let v be a video sequence and ℓ its labeling. Then PPN-MPS runs in time $\mathcal{O}(|v| \cdot |T| \cdot |P|^k)$.

Proof. We will first look at how many possible markings there are if the PPN is bounded by k. If we denote the number of possible markings for a PPN with n nodes which is bounded by k by $x_{n,k}$, then by fixing one token (n possibilities), we have that $x_{n,k} = n \cdot x_{n,k-1}$. This leads to $x_{n,k} = n^{k-1} * x_{n,1} = n^k * x_{n,0} = n^k$. The algorithm process each frame in the video (line 4), at each iteration looking at every element of S (line 5); however, there are at most a constant number of elements in S for each marking, therefore the size of S is $\mathcal{O}(|P|^k)$. For every element in S, we look at at most $|T|$ transactions enabled for that marking, hence the complexity result. We normally expect that in regular activity definitions, the degree k of the polynomial is small; also, for a fixed activity definition, the algorithm is linear in the size of the video.

We should point out that the probabilities in the PPN already enforce a bound on the number of tokens in the network. Assume that there are no transactions with a probability of 1 (for instance, by adding skip transitions at every place). Let c be the upper bound of the probabilities in $\mathcal{P}$. New tokens are generated as a result of transitions with a postset of cardinality strictly greater than 1 – one can generate at most $|P - 2|$ new tokens on each such transition. Furthermore, for a threshold p_t, each token can be used to fire at most $\frac{\log p_t}{\log c}$ transitions before the probability is lower than p_t. Assuming the initial number of tokens is $|\mu_0| = \sum_{x \in P} \mu_0(x)$, this means we can have at most $|\mu_0| \cdot |P - 2|^{\frac{\log p_t}{\log c}}$ tokens in the network.

Handling Multiples Instances: Note that PPN-MPS already handles a certain degree of parallelism in the activities occurring in the video. This is a direct consequence of the fact that $\langle \mu_0, 1, 0\rangle$ is always a member of S, hence a new activity could be started at every frame. However, PPN-MPS does not directly handle the case in which multiple actions that start an activity (i.e., enable a transition from $\langle \mu_0, 1, 0\rangle$) occur in the *same frame*. In our implementation, we handled this case by *duplicating* the PPN for each action of a frame that can potentially start a new activity. The maximum number of such actions per frame is $\max_{f\in v}(|\ell(f)|)$. Hence, taking the complexity of the scene into account, the worst-case complexity becomes $\mathcal{O}(\max_{f\in v}(|\ell(f)|) \cdot |v| \cdot |T| \cdot |P|^k)$. However, in our two datasets we only needed duplication in under 1% of the frames, and $\max_{f\in v}(|\ell(f)|) \leq 3$ in all cases.

[7] p may be different than p_k by firing skip transitions.

[8] This is called a *bounded* PPN – i.e., one that does not generate an infinite number of tokens. We will see that the probabilities in the PPN may enforce a variant of this rule.

IV. Algorithms for Activity Recognition

Let us now consider the PPN-MPA problem. Let $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ be a given set of activity definitions. If we assume that there is only one activity $\mathcal{P}_l$ which satisfies the video sequence v with maximal probability, a simple binary search iterative method – which we shall call naivePPN-MPA – can employ PPN-MPS to find the answer:

1) Begin with a very high threshold p_t.
2) Run PPN-MPS for all activities in $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ and threshold p_t.
3) If more than one activity definition has a non-empty result from PPN-MPS, then increase the threshold by $\frac{1-p_t}{2}$ and go to step 2.
4) If no activity has a non-empty result from PPN-MPS, decrease the threshold by $\frac{p_t}{2}$ and go to step 2.
5) If only one activity $\mathcal{P}_l$ has a non-empty result from PPN-MPS, return $\mathcal{P}_l$.

Clearly, this method can be improved to account for the case in which multiple activity definitions are satisfied by the video sequence with the same probability. Even under these conditions, the algorithm requires multiple passes through the video (for each run of PPN-MPS) until a delimiter threshold is found. We will now present a much more efficient method – the PPN-MPA algorithm.

Algorithm 2 Activity recognition queries (PPN-MPA)

Input: Set of PPNs $\{(P_i, T_i, \rightarrow_i, F_i, \delta_i)\}_{i\in[1,m]}$, initial set of configurations μ_i^0, video sequence labeling ℓ
Output: Set of PPNs that is satisfied by ℓ with maximal probability
1: **for all** $\{P_i, T_i, \rightarrow_i, F_i\}$ **do**
2: $S \leftarrow S \cup \{\langle \mu_i^0, 1\rangle\}$
3: **end for**
4: $max_p \leftarrow 0$
5: $R \leftarrow \emptyset$
6: **for all** f frame in video sequence **do**
7: **for all** $\langle \mu, p\rangle \in S$ **do**
8: **for all** $t \in T$ enabled in μ s.t. $\delta(t) \subseteq \ell(f)$ **do**
9: $\mu' \leftarrow$ fire transition t for marking μ
10: $p^* \leftarrow \Pi_{\{x\in P|x\rightarrow t\}} F(x, t)$
11: $p' \leftarrow p \cdot p^*$
12: **if** $p' \geq p_t \cdot max_p$ **then**
13: **if** $\forall x \in P$ s.t. $\mu'(x) > 0$, $p' \cdot p^*(x) \geq p_t \cdot max_p$ **then**
14: $S \leftarrow S \cup \{\langle \mu', f'\rangle\}$
15: **end if**
16: **end if**
17: **if** μ' is a terminal configuration **then**
18: $S \leftarrow S - \{\langle \mu', p'\rangle\}$
19: **if** $p > max_p$ **then**
20: $R \leftarrow \{$ activity definition corresponding to $\mu'\}$
21: $max_p \leftarrow p$
22: **else if** $p = max_p$ **then**
23: $R \leftarrow R \cup \{$ activity definition corresponding to $\mu'\}$
24: **end if**
25: **end if**
26: **end for**
27: **for all** $t \in T$ enabled **do**
28: Fire skip transitions and update $\langle \mu, p\rangle$
29: Prune $\langle \mu, p\rangle \in S$ s.t. $\forall x \in P$ s.t. $\mu(x) > 0$, $p \cdot p^*(x) < p_t \cdot max_p$
30: **end for**
31: **end for**
32: **end for**
33: **return** R

PPN-MPA (Algorithm 2) uses the same storage structure as PPN-MPS. However, this time we are no longer interested in the start and end frames of the subsequences matching one of the activity definitions. Instead, we maintain a global maximum relative probability (max_p) with which the video satisfies any of the activities in $\{P_i, T_i, \rightarrow_i, F_i, \delta_i\}$. max_p is updated any time a better relative probability is found (lines 19–21); R maintains a list of the activity definitions that have been satisfied with the relative probability max_p. At the return statement on line 33, we are guaranteed that the activity definitions in R are satisfied by the video with maximal probability.

Theorem 3 (PPN-MPA correctness): 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 and ℓ its labeling. Let $\mathcal{P}_1, \ldots, \mathcal{P}_k$ be the activity definitions returned by PPN-MPA. Then the following hold:

(i) $\forall i \in [1, k]$, let p_i be the relative probability with which ℓ satisfies $\mathcal{P}_i$. Then $p_1 = \ldots = p_k$.
(ii) $\forall \mathcal{P}' \in \{(P_i, T_i, \rightarrow_i, F_i, \delta_i)\}_{i\in[1,m]} - \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$, ℓ does not satisfy $\mathcal{P}'$ with relative probability greater than or equal to $p_1 = \ldots = p_k$.

Proof. The proof of correctness for PPN-MPS shows that the algorithm does not "miss" transitions, but ignores only those that cannot end up above the threshold. In this case, the threshold is the variable max_p, which means the only transitions being ignored are those for which there is already a "better" activity (satisfied with the probability max_p) in R; this is sufficient for condition (ii). Condition (i) is satisfied by the fact that max_p always increases (lines 19–24) and all elements in R must be satisfied with the same probability (line 22).

Theorem 4 (PPN-MPA 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 and ℓ its labeling. PPN-MPA runs in time $\mathcal{O}(m \cdot |v| \cdot |T| \cdot |P|^k)$.

Proof. The same reasoning as for PPN-MPS applies here as well, with the addition of the factor m which accounts for the size of S when we analyze m different activities.

Similarly to PPN-MPS, PPN-MPA will clone the PPNs each time multiple actions in a single frame can potentially start a new activity. Taking into account the maximum number of such actions would yield a complexity of $\mathcal{O}(\max_{f\in v}(|\ell(f)|) \cdot m \cdot |v| \cdot |T| \cdot |P|^k)$, however in practice we only needed to clone the PPNs very rarely.

V. System Implementation

Our PPN-MPS, naivePPN-MPA and PPN-MPA algorithms were implemented in Java. In addition, our image processing library is based on the People Tracker framework of [25] with several modifications outlined in this section.

The People Tracker[9] [25], [26] uses object detection and motion algorithms to track persons, groups of persons, vehicles and other objects in a given sequence of images. From a functional point of view, the framework consists of four components:

1) The *Motion Detector* subtracts the background from the current image and thresholds the resulting difference image to obtain moving objects ("blobs").
2) The *Region Tracker* tracks all blobs identified in the previous step by a frame-by-frame matching algorithm. The next position of a blob is predicted using a first-order motion model and then matched against the positions of the blobs in the new frame.
3) The *Head Detector* is used to identify blobs corresponding to people by creating a vertical projection histogram

[9]Freely available under a public GPL license at http://www.siebel-research.de.

of the upper part of every tracked region. This is based on the algorithms of Haritaoglu et al. [27].

4) The *Active Shape Tracker* uses B-spline 2D shape models to resolve the type of a tracked region to a person, group of persons, vehicle, package, etc.

In addition to these, the following extensions were made:

1) Shadows reduced the accuracy of detecting object splits and merges. To take care of this, we implemented the algorithm of [28] for detection of moving shadows.
2) We implemented the face recognition algorithm of [29]. The algorithm detects faces in a frame using a wavelet transform based approach and then compares features of detected faces and faces stored in the database. We chose this particular face recognition algorithm because of its speed and good accuracy.
3) To accurately answer queries of type: "Is object O in frame f?", one needs to perform occlusion analysis. We implemented the occlusion detection algorithm of [30] for this task.

Based on the output of the above framework and the aforementioned extensions, we implemented a number of low-level predicates to answer simple queries such as:

1) Is object O present in frame f (in a particular zone Z)? This is accomplished through the object matching algorithm in the *Region Tracker* module.
2) Is object O_i occluding O_j in frame f ?
3) Are objects O_i and O_j splitting/merging in frame f ?
4) What is the type of object O_i (person, vehicle, group of persons, etc.) in frame f_i ?
5) Is object O identical to the person P from the database ?

All the above queries are typically answered with a degree of confidence reported as a probability by the underlying low-level module. Finally, we have combined these low-level queries into sequences in order to generate the higher-level labeling our algorithms take as input.

VI. Experimental Results

Our experiments were performed on two datasets. The first dataset (Bank) consists of 7 videos of 15–20 seconds in length, some depicting staged bank attacks and some depicting day-to-day bank operations. The dataset has been documented in [20]. Figure 1 contains a few frames from a video sequence depicting a staged bank attack. Figure 8(top) contains a few frames from a video sequence depicting regular bank operations. The second dataset consists of approximately 118 minutes of TSA tarmac footage and contains activities such as plane take-off and landing, passenger transfer to/from the terminal, baggage loading and unloading, refueling and other airport maintenance.

We compared the precision and recall of our algorithms against the ground truth provided by human reviewers as follows. Four human reviewers analyzed the videos in the datasets. The reviewers were provided with VirtualDub (http://www.virtualdub.org), which allowed them to watch the video in real time, as well as browse through it frame by frame. They received detailed explanations on the activity model, as well as sets of activity definitions for each video and were asked to mark the starting and ending frame of each activity they encountered. An average over the four reviewers was then considered as our ground truth dataset for the purpose of computing precision and recall. We compared the subvideos provided by the reviewers (note that probabilities were completely hidden from the users.) with those returned by our algorithms above a given probability threshold value.

All experiments were performed on a Pentium 4 2.8 GHz machine with 1 GB of RAM running SuSE Linux 9.3. The labeling data generated by the low-level vision methods was stored in a MySQL database. The labeling was compressed for blocks of frames, as neighboring frames often tend to have identical labeling. Running times include disk I/O for accessing the labeling database, but not the time taken to generate the labeling.

A. Experimental evaluation for the Bank dataset

For the Bank dataset, we defined a set of 5 distinct activities and constructed the associated PPNs. They are:

(a1) Regular customer-bank employee interaction.
(a2) Outsider enters the safe.
(a3) A bank robbery attempt – the suspected assailant does not make a getaway.
(a4) A successful bank robbery similar to the example in Figure 3.
(a5) An employee accessing the safe on behalf of a customer.

For the PPN-MPA experiments, we grouped these into different sets of activities as: $S_1 = \{a5\}$, $S_2 = \{a3, a5\}$, $S_3 = \{a3, a4, a5\}$ and $S_4 = \{a2, a3, a4, a5\}$.

For the PPN-MPS algorithm, we observed empirically that in the majority of cases, at most one instance of the activity occurs at any given time. For instance, it is highly unlikely that we have two bank attacks (a3, a4) interleaved in the same video sequence; although we might have a number of regular customer operations (a1), we assumed that the special events we are interested in monitoring are generally rare. We took this into account by developing two different versions of the algorithm. The first is the one depicted in Section III; the second version optimizes the algorithm by assuming there can be no interleaving of activities in a video; we will denote the latter with the suffix *NI* (no interleaving). This approach generally leads to an improvement in running time, at the expense of precision.

Running Time Analysis: In our first set of experiments, we measured the running times of PPN-MPS, naivePPN-MPA and PPN-MPA for the five activity definitions described above while varying the size of the labeling (the number of action symbols in the labeling[10]). The running times are shown in Figure 9. As expected, the non-interleaving (NI) version of PPN-MPS is more efficient than its counterpart. naivePPN-MPA exhibits a seemingly strange behavior – running time increases almost exponentially with labeling size, only to drop suddenly at a labeling size of 30. On further analysis of the results, we discovered that at this labeling size the activity

[10]We remind the reader that the videos are 15-20 second long and that labeling has been compressed from frame to block level.

Fig. 8. (top) Example video sequence of regular customer interaction at a bank; (bottom) Sample frames from the TSA dataset.

definitions first begin to be satisfied by the labeling. When no activity definition is satisfied, naivePPN-MPA performs many iterations in order to decrease the threshold close to 0. Somewhat surprisingly, it is comparable in terms of running time with PPN-MPA when the activity is satisfied by at least one subsequence.

Accuracy: In the second set of experiments, we measured the precision and recall of PPN-MPS, naivePPN-MPA and PPN-MPA w.r.t. the human reviewers. In the paper describing the dataset, Vu et al. [20] only provide figures for recall between 80% and 100% (depending on the camera angle), without further details. In this case, the minimality condition used by the PPN-MPS algorithm poses an interesting problem – we found that 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). Assume that both the human reviewer and the algorithm return a subsequence corresponding to an activity. We will denote the subsequence returned by the algorithm A and the one returned by a human reviewer H. Precision/recall measures at the frame level[11] are defined as: $P_f = \frac{|H \cap A|}{|A|}$ and $R_f = \frac{|H \cap A|}{|H|}$. However, we can relax this condition by stating that *any subsequence returned by the algorithm that intersects a subsequence returned by a human reviewer should be counted as correct*. Let the set of sequences returned by the algorithms be $\{A_i\}_{i \in [1,m]}$ and the set of sequences returned by the human annotators be $\{H_j\}_{j \in [1,n]}$. Then we can write precision as $P_e = \frac{|\{A_i | \exists H_j \ s.t. \ A_i \cap H_j \neq \emptyset\}|}{m}$ and recall as $R_e = \frac{|\{A_i | \exists H_j \ s.t. \ A_i \cap H_j \neq \emptyset\}|}{n}$.

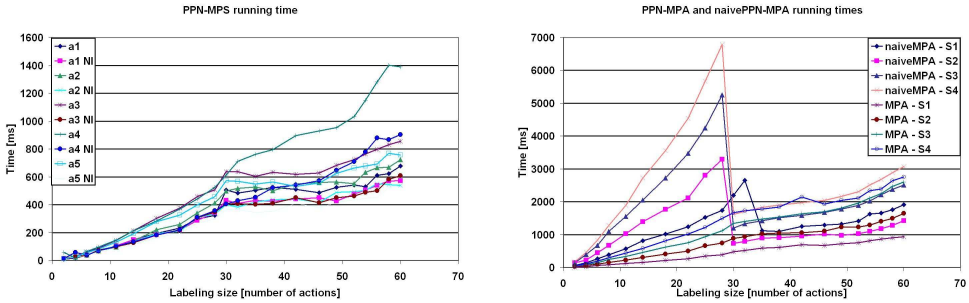
Figure 10(a) shows the frame precision and recall for the two versions of the PPN-MPS algorithm. We notice that precision is always very high and generally constant, while recall has a rather surprising variation pattern – namely, recall is monotonic with the threshold. The explanation for the apparently anomalous behavior comes from the fact that for low thresholds, the number of candidate subsequences is relatively high, hence the minimality condition causes *fewer frames* to appear in the answer. The monotonic behavior pattern stops when the threshold becomes high enough (.45 in this case). Furthermore, we see that the non-interleaving version of the algorithm has a slightly higher recall than the standard version. We observed that human annotators usually focus on detecting one activity at a time and very seldom consider two or more activities that occur in parallel. Hence, the NI version of the algorithm is informally "closer" to how the reviewers operate. With respect to the PPN-MPA algorithm, we see a very good match in terms of the activities returned by the human annotators and the ones returned by PPN-MPA for the $S_4 = \{a2, a3, a4, a5\}$ – the most complete dataset. Poorer performance on the other activity sets is explained by the limitation in the possible choices of the algorithm. PPN-MPA could only choose activities from a given set, sometimes missing some results. For instance, if a successful bank robbery occurs in the video sequence, PPN-MPA cannot return it from the choices in $S_2 = \{a3, a5\}$, and thus returns the closest possible match. Lastly, event precision was always 100% for PPN-MPS, meaning the algorithm did not yield any false positives. Recall was greater than 90% for the entire dataset for both PPN-MPS and PPN-MPS-NI.

[11]Only with respect to one subsequence, but easily extensible to many subsequences

TABLE I
MEASURE OF OVERLAP OF SEQUENCES RETURNED BY HUMAN REVIEWERS AND BY THE PPN-MPS ALGORITHM FOR $p_t = .7$. A MEASURE OF 1 INDICATES PERFECT OVERLAP AND 0 INDICATES NO OVERLAP.

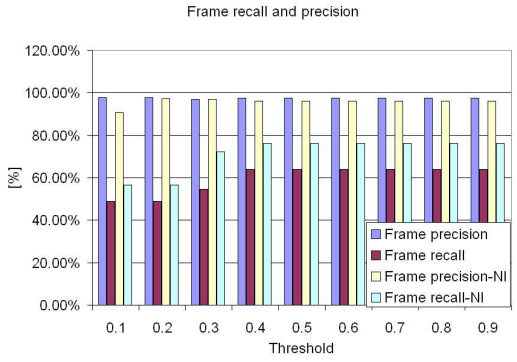
Video	Reviewer 1		Reviewer 2		Reviewer 3		Reviewer 4	
	$\frac{\lvert H \cap A \rvert}{\lvert H \rvert}$	$\frac{\lvert H \cap A \rvert}{\lvert A \rvert}$	$\frac{\lvert H \cap A \rvert}{\lvert H \rvert}$	$\frac{\lvert H \cap A \rvert}{\lvert A \rvert}$	$\frac{\lvert H \cap A \rvert}{\lvert H \rvert}$	$\frac{\lvert H \cap A \rvert}{\lvert A \rvert}$	$\frac{\lvert H \cap A \rvert}{\lvert H \rvert}$	$\frac{\lvert H \cap A \rvert}{\lvert A \rvert}$
1	.82	1	.83	1	.84	1	.93	1
2	.36	1	.37	1	.38	1	.41	1
3	.74	1	.83	.93	.67	.97	.72	1
4	.75	1	.78	1	.76	1	.94	.96
5	.56	.99	.58	.98	.58	1	1	.74

Temporal Overlap with Groundtruth: Since precision and recall at the event level was very high, we investigated to what extent the sequences returned by the reviewers and by PPN-MPS actually overlap. To see why this is relevant, consider an 8 frame-video such as the one in Figure 8. Assume that for activity (a5), PPN-MPS returns the subsequence from frame 1 to frame 4, whereas the human reviewer returns the subsequence between frames 4 and 8. The event precision and



(a) PPN-MPS (b) PPN-MPA and naivePPN-MPA

Fig. 9. Comparison of Running times for PPN-MPS, naivePPN-MPA and PPN-MPA on the Bank dataset.



(a)

Video	S_1	S_2	S_3	S_4	Human
1	(a5,.61)	(a3,.99)	(a3, .99); (a4,.99)	(a4, 1)	(a4,1)
2	(a5,.61)	(a3,.99)	(a3, .99); (a4,.99)	(a4, 1)	(a4,1)
3	-	(a3,.2)	(a3, .2)	(a4, .61)	(a4,.75); (a2, .75)
4	(a5,1)	(a5,.99)	(a5, 1)	(a5,1); (a4, 1); (a2,1)	(a4,1)
5	(a5,.61)	(a3,.99)	(a3, 1); (a4, .99)	(a2,1); (a4,.99)	(a4,1)
6	-	-	-	-	-
7	-	-	-	-	-

(b)

Fig. 10. (a) Comparison of frame precision and recall for the two versions of the PPN-MPS algorithm – PPN-MPS and PPN-MPS-NI, (b) Comparison of results returned by PPN-MPA algorithm and human subjects. Results indicate a good match of the obtained results with human annotators.

recall measures match the result returned by PPN-MPS with that of the reviewer, but the 1-frame overlap between the two is proof to the contrary. We therefore measured the amount of overlap between sequences from reviewers and PPN-MPS for all probability thresholds in .1 increments between .1 and 1. Results were similar for all thresholds; we show the overlap for $p_t = .7$ in Table I for the five video sequences containing bank attacks. The table gives the overlap as a percentage of the sequence returned by the algorithm and by the human reviewer. The data supports our initial intuition that reviewers tend to return larger sequences than the algorithm, since in most cases the result of the algorithm is a subset of that of the reviewers – indicated by the number of 1 values on the right-hand column for each reviewer.

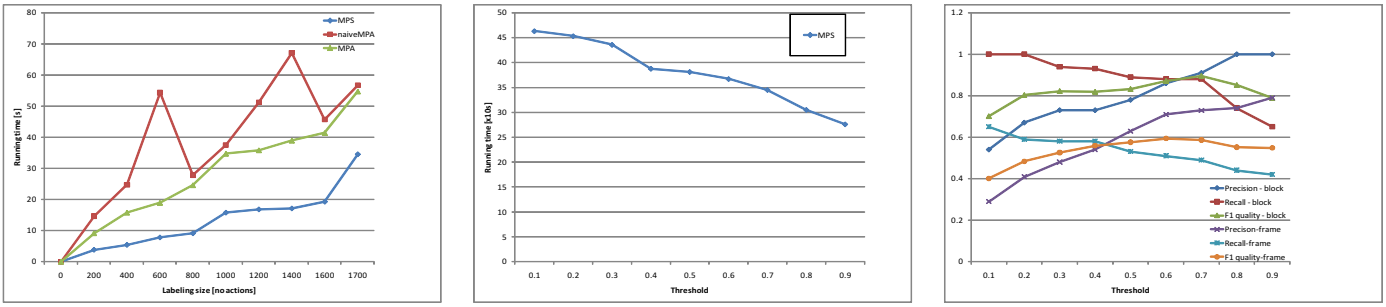
B. Experimental evaluation for the TSA dataset

For the TSA dataset, we used a set of 23 activities ranging from flight take-off and landing, embarkation or debarkation of passengers, baggage handling and other maintenance operations. We evaluated PPN-MPS, naivePPN-MPA and PPN-MPA on the dataset in an attempt to establish their performance on a large dataset, as well as find evidence for our earlier conclusions drawn from the evaluation on the Bank dataset. The labeling of the TSA dataset consists of 1716 action symbols and was obtained in two stages. In the first stage, we ran the low-level image processing primitives to obtain the initial labeling in approximately 3 hours. In the second stage, we semi-automatically processed the outcome of the previous process for about 3 more hours – for instance, to determine blocks of frames with identical labeling instead of using individual frames. The ground truth contains 77 instances of activities. naivePPN-MPA and PPN-MPA were given as input the entire set of activity definitions.

Running Time Analysis: First, we evaluated the running time of the algorithms for different labeling sizes. We obtained the data points by analyzing the subvideos starting from the beginning of the TSA footage up to a point where $|\ell(v)| = N$. We varied N in increments of 200 action symbols from 0 to 1716. The results in Figure 11(a) show the same type of variations as in the Bank dataset. Note that the entire video is processed by PPN-MPS in 346 seconds and by naivePPN-MPA and PPN-MPA in a little under 10 minutes[12]. We analyzed the videos that correspond to relatively stable values in both PPN-MPS and PPN-MPA and to the spike regions in naivePPN-MPA (between 400 and 800, and 1000 and 1400 action symbols respectively). We found that these are "quiet" portions of the video, when there are very few activities on the tarmac. We observed that the binary search algorithm in naivePPN-MPA causes it to deteriorate in terms of performance whenever there are very few activities being completed. The results for PPN-MPS in Figure 11(a) are provided for a threshold value of 0.7. We repeated the experiments for all threshold values between 0.1 and 0.9 and we obtained the same curve characteristics (Figure 11(b)).

Accuracy: Second, we evaluated the precision and recall of PPN-MPS and PPN-MPA on the TSA dataset. To evaluate

[12]We remind the reader that the labeling had been precomputed in a MySQL database.



(a) (b) (c)

Fig. 11. (a) Variation of Running time with size of label set $|\ell(v)|$, of PPN-MPS, naivePPN-MPA and PPN-MPA, (b) Variation of Running time of PPN-MPS with p_t, (c) Answer quality for PPN-MPS measured via precision/recall

PPN-MPS, we ran PPN-MPS for every of the 23 activities and compared the subvideos returned by PPN-MPS to those of the ground truth, both at the frame and at the block level. We repeated the process for different thresholds between 0.1 and 0.9. The results are displayed in Figure 11(c). We notice that the ideal probability thresholds (from the point of view of answer quality), both in the case of frame-based and block-based processing are between .6 and .7. To evaluate PPN-MPA, we divided the video into equal segments of approximately 2 minutes and 35 seconds and then ran PPN-MPA with the entire set of 23 activity definitions as input. We then compared the most probable activities obtained with those from the ground truth. The results of PPN-MPA were correct (from a precision point of view) for 44 of the 46 segments. The two false positive segments were due to false positives in occlusion processing. In terms of recall, PPN-MPA missed four activities; we observed that the false negatives only occurred for activity definitions that were almost subparts of larger activities. As we mentioned before, due to its greedy nature, PPN-MPA has a bias for shorter activities. We also measured the amount of overlap between the results of PPN-MPS and those of the ground truth. For $\frac{|H \cap A|}{|H|}$, the values were of .85, .89, .91, .86 respectively and for $\frac{|H \cap A|}{|A|}$, the values were 1, .97, 1, .93.

Impact of low-level processing: We also studied the impact of errors in the low-level image processing algorithms on higher-level reasoning using the Bank data. Specifically, we analyzed the accuracy of the following low-level tasks a) Object tracking (T1), b) Object detection (T2), c) Merge/split detection (T3), d) Occlusion detection (T4), e) Face Recognition (T5).

For all these tasks, we computed the average accuracy in the same set of experiments that evaluate PPN-MPS precision and recall for a probability threshold equal to .65. The results are shown in Table II. It is important to note that these recognition percentages are only indicative of the relative difficulty of each of the tasks. Performance on each of the tasks can be easily improved by careful domain-specific retraining of classifiers (for face recognition for instance) and estimation of detection parameters for individual deployments.

We also considered how the errors in the five tasks propagate to the higher level algorithms. We found that (T1) and (T2) had by far the greatest impact – approximately 30% of the misses in PPN-MPS and PPN-MPA were caused by errors in object tracking, whereas 45% were caused by the misclassification of object types. Two factors contribute to the relatively smaller impact of (T3)–(T5). First,usually errors in tasks (T3)–(T5) are localized to a relatively small number of frames where the camera angle or lighting maybe detrimental to the detection algorithms. However, misses in earlier frames might be corrected later on, when conditions in the video change. As an example, even if face recognition fails to identify a person from the database due to shadows, occlusions, etc., if the algorithm succeeds in a successive block of frames, activity detection in PPN-MPS and PPN-MPA will still be triggered. Second, as we intially suspected, the presence of skip transitions in the model allows us to filter out part of the noise produced by the low-level primitives. However, misses in (T1) (not detecting an object altogether) or (T2) (misclassifying an object) are generally not recoverable through skip transitions.

TABLE II
ACCURACY FOR LOW-LEVEL PROCESSING TASKS

Task	T1	T2	T3	T4	T5
Accuracy	87.9%	68.6%	76.5%	65.4%	61.5%

C. Learning the Parameters of the Petri Net

We do not explicitly deal with the problem of learning the probability distributions associated with the places of the PPN. To deal with this problem in a more systematic way, we hope to extend the general parameter learning techniques used in graphical models such as EM ([31]) to the Probabilistic Petri net setting. In the absence of training data, one could rely on an expert's belief about the uncertainty in the unfolding of an activity, or one can use information about the error rates of the low-level modules to associate probabilities to the skip transitions.

VII. CONCLUSIONS

There has been extensive work in computer vision on the problem of modeling and recognizing human activities. Numerous techniques have been developed to learn activity models – both statistical (e.g. HMM's) and knowledge based (e.g. logic based). In this paper, we build upon such methods. We propose the concept of a Constrained Probabilistic Petri Net using which it is possible to specify activities that can be executed in a multiplicity of ways. Moreover, constrained PPNs use "skip" transitions to account for the fact that

activities in the real world may vary from a hard-coded model of those activities.

We then propose very fast algorithms for answering threshold activity queries. These queries ask us to find **minimal** segments of a video that contain a given activity with a probability exceeding a given threshold. To our knowledge, we are the first to provably develop algorithms to find such minimal segments. We also show that the algorithm's complexity is reasonable. We further show experimentally that our algorithm is both efficient and highly accurate. We also propose a very fast algorithm for activity recognition – finding the activity that is maximally probable in a given video (or video segment). Again, we develop a provably correct algorithm. We analyze the complexity of the algorithm and experimentally show that our algorithms are both fast and accurate.

REFERENCES

[1] R. Chellappa, N. P. Cuntoor, S. W. Joo, V. S. Subrahmanian, and P. Turaga, *Understanding Events: How Humans See, Represent, and Act on Events*. Oxford University Press, To appear, ch. Computational Vision Approaches to Event Modeling.

[2] H. Buxton and S. Gong, "Visual Surveillance in a Dynamic and Uncertain World." *Artif. Intell.*, vol. 78, no. 1-2, pp. 431–459, 1995.

[3] F. Bremond and M. Thonnat, "Analysis of Human Activities Described by Image Sequences," in *Intl. Florida AI Research Symp.*, 1997.

[4] Q. Cai and J. Aggarwal, "Tracking Human Motion Using Multiple Cameras," in *13th International Conference on Pattern Recognition*, 1996.

[5] A. Bobick and J. Davis, "Real-time Recognition of Activity Using Temporal Templates," in *Proc. IEEE Workshop on Appln. in Comput. Vision*, 1996, pp. 39–42.

[6] N. Vaswani, A. Roy Chowdhury, and R. Chellappa, "Activity Recognition Using the Dynamics of the Configuration of Interacting Objects." in *CVPR (2)*, 2003, pp. 633–642.

[7] T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell, and J. Weber, "Automatic Symbolic Traffic Scene Analysis Using Belief Networks." in *AAAI*, 1994, pp. 966–972.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[9] S. Hongeng and R. Nevatia, "Multi-Agent Event Recognition." in *ICCV*, 2001, pp. 84–93.

[10] M. Brand, "Understanding manipulation in video," *Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition*, 1996.

[11] Y. Ivanov and A. Bobick, "Recognition of Visual Activities and Interactions by Stochastic Parsing." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 852–872, 2000.

[12] M. Albanese, V. Moscato, A. Picariello, V. S. Subrahmanian, and O. Udrea, "Detecting stochastically scheduled activities in video," in *IJCAI*, 2007, pp. 1802–1807.

[13] V. D. Shet, D. Harwood, and L. S. Davis, "Vidmap: Video monitoring of activity with prolog," in *IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS), Como, Italy*, 2005.

[14] C. Castel, L. Chaudron, and C. Tessier, "What is going on? A High-Level Interpretation of a Sequence of Images," in *ECCV Workshop on Conceptual Descriptions from Images*, Cambridge, U.K., 1996.

[15] N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis, "Representation and Recognition of Events in Surveillance Video Using Petri Nets," in *Second IEEE Workshop on Event Mining 2004, CVPR2004*, 2004, pp. 112–112.

[16] D. Moore and I. Essa, "Recognizing multitasked activities from video using stochastic context-free grammar," *AAAI*, 2002.

[17] S. W. Joo and R. Chellappa, "Recognition of multi-object events using attribute grammars," *International Conference on Image Processing*, pp. 2897–2900, 2006.

[18] C. Lesire and C. Tessier, "Particle petri nets for aircraft procedure monitoring under uncertainty," *ICATPN*, pp. 329–348, 2005.

[19] J. Cardoso, R. Valette, and D. Dubois, "Possibilistic petri nets," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 29, no. 5, pp. 573–582, 1999.

[20] V. T. Vu, F. Bremond, and M. Thonnat, "Automatic Video Interpretation: A Novel Algorithm for Temporal Scenario Recognition," in *Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[21] R. David and H. Alla, "Petri nets for Modeling of Dynamic Systems A Survey," *Automatica*, vol. 30, no. 2, pp. 175–202, 1994.

[22] T. Murata, "Petri nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[23] M. A. Marsan, G. Balbo, G. Chiola, G. Conte, S. Donatelli, and G. Franceschinis, "An Introduction to Generalized Stochastic Petri Nets." *Microelectronics and Reliability*, vol. 31, no. 4, pp. 699–725, 1991, newsletterInfo: 39.

[24] S.-M. Chen, J.-S. Ke, and J.-F. Chang, "Knowledge Representation Using Fuzzy Petri Nets." *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, no. 3, pp. 311–19, sep 1990.

[25] N. T. Siebel and S. Maybank, "Fusion of multiple tracking algorithms for robust people tracking," in *Proceedings of the 7th European Conference on Computer Vision (ECCV 2002)*, vol. IV, May 2002, pp. 373–387.

[26] ——, "The advisor visual surveillance system," in *Proceedings of the ECCV 2004 workshop Applications of Computer Vision (ACV'04)*, May 2004, pp. 103–111.

[27] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 809–830, 2000.

[28] A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: Algorithms and evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 7, pp. 918–923, 2003.

[29] Y. Zhu, S. Schwartz, and M. Orchard, "Fast face detection using subspace discriminant wavelet features," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2000, pp. 636–642.

[30] S. Khan and M. Shah, "Tracking people in presence of occlusion," in *Proceedings of Asian Conference on Computer Vision*, January 2000.

[31] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of Royal Statistical Society Series B,*, vol. 39, pp. 1–38, 1977.