Video Event Modeling and Recognition in Generalized Stochastic Petri Nets
Gal Lavee, Michael Radzsky, Ehud Rivlin, and Artyom Borzin

Abstract—In this paper, we propose the surveillance event recognition framework using Petri Nets (SERF-PN) for recognizing event occurrences in video. The Petri Net (PN) formalism allows a robust way to express semantic knowledge about the event domains as well as efficient algorithms for recognizing events as they occur in a particular video sequence. The major novelties of this paper are extensions to both the modeling and the recognition capacities of the Object PN paradigm. The first contribution of this paper is the extension of the PN representational capacities by introducing stochastic timed transitions to allow modeling of events which have some variance in duration. These stochastic timed transitions sample the duration of the condition from a parametrized distribution. The parameters of this distribution can be specified manually or learned from available video data. A second representational novelty is the use of a single PN to represent the entire event domain, as opposed to previous approaches which have utilized several networks, one for each event of interest. A third contribution of this paper is the capacity to probabilistically predict future events by constructing a discrete time Markov chain model of transitions between states. The experiments section of the paper thoroughly evaluates the application of the SERF-PN framework in the event domains of surveillance and traffic monitoring and provides comparison to other approaches using the CAVIAR dataset [1], a standard dataset for video analysis applications.

Index Terms—Action, activity, behavior, event, Petri Net, scenario, video analysis.

I. INTRODUCTION

THE ABILITY of humans to recognize events in video sequences greatly exceeds that of the most successful automatic video event recognition approaches. One reason for this may be that humans are equipped with a great deal of semantic knowledge about what constitutes a particular event within an event domain. While it is a difficult problem to encode all human knowledge, with a powerful representational formalism we contend that it is possible to distill the knowledge needed to correctly recognize and classify events in a particular event domain, particularly simple domains, such as those found in surveillance applications.

The Petri Net (PN) formalism is such a representational mechanism which allows us to express knowledge about an event domain in terms of semantic object properties and relationships. We can use the PN description of the event domain to devise efficient event recognition algorithms with good accuracy that are not susceptible to human failings, such as slowness, subjectivity, and fatigue (classifying hours of surveillance video can be tedious).

In this paper, we propose the surveillance event recognition framework using Petri Nets (SERF-PN) framework for semantic description of the events of interest in a particular event domain and for recognition of these events in unlabeled video sequences. Within SERF-PN we utilize a representation based on Petri Nets that captures the semantics of a given domain.

Provided such a model of the events in a particular domain, SERF-PN is able to generate an event description of an unlabeled video sequence. Additionally, using the statistics gathered from observing a set of training data, SERF-PN is able to make assertions on the likelihood of future events.

The major novelties of this paper are extensions to both the modeling and the recognition capacities of Object PN paradigm (which models objects as PN tokens). We first extend the representational capacities by introducing stochastic timed transitions to allow modeling of events which have some variance in duration. These stochastic timed transitions sample the duration of the condition from a parametric distribution. The parameters of this distribution can be specified manually or learned from available video data. A second representational novelty of the paper is the use of a single PN to represent the entire event domain, as opposed to previous approaches which have utilized several networks, one for each event of interest.

A third contribution of this paper is the capacity to probabilistically predict future events by constructing a discrete time Markov chain model of transitions between states. This model is constructed using dynamic marking analysis from a set of training data.

Of course, it is important that any algorithmic approach to recognition of events in video be computationally tractable, to this end we provide a formalization and complexity analysis of our proposed algorithm.

In summary, this paper proposes a framework for constructing event models that capture expert knowledge of the domain as well as allowing parameter tuning from training data. The constructed model can be viewed as a static entity which allows parameter tuning to capture dynamic aspects of specific applications. This representation can then be used as input to
an efficient recognition algorithm which allows recognition of events in an unlabeled video sequence.

This paper is organized as follows. Section II presents related works in video event representation and recognition. Section III introduces the Petri Net Formalism and the concept of marking analysis. Section IV discusses how the PN formalism is used in our SERF-PN model. Section V discusses the training of the SERF-PN system. Section VI gives the components of the SERF-PN process. Section VII provides detailed examples of a PN event model construction. Section VIII formalizes our algorithm and provides a complexity analysis. Experiments are discussed in Section IX. Finally, we provide some discussion in Section X and conclude in Section XI.

II. RELATED WORK

The specification of a semantic description of events has been an active research topic. R. Nevatia et al. presented the Event Recognition Language (ERL) [2] which can describe hierarchical representation of complex spatiotemporal and logical events. The proposed event structure consists of such units as primitive, single-thread, and multithread events. More recently, R. Nevatia et al. developed the Video Event Representation Language (VERL) to describe an ontology of events and the complementary Video Event Markup Language (VEML) to annotate instances of the events described in VERL. Another event representation ontology, called CASEVE, is based on natural language representation and was proposed by A. Hakeem et al. in [4] and then extended in [5].

Event recognition has also been a much-studied research topic. Most current approaches involve defining models for specific event types that suit the goal of a particular domain and developing methods for recognition that correspond to these models. The complexity that exists in the domain of video sequences requires models that can take into account a number of factors, including spatial, temporal, and logical relations. This has led to the proposal of a number of model formalisms, including nearest neighbor [6], [7], support vector machines [8], neural networks [9], [10], Bayesian networks (BN) [11]–[14], hidden Markov models (HMMs) [15]–[21], context-free grammars (CFG) [22]–[24], Petri Nets [25]–[27], and temporal constraint satisfaction problem (T CSP) solvers [28].

Each of these models has been successfully adopted to applications in the domain of event recognition [29].

PNs are a well-studied formalism widely used in many research areas. PNs have been recently suggested as a video event model which defines the semantics of state transitions within the domain. Two main classes of approaches using this formalism have emerged. The plan-based PN has been presented in the work by C. Castell et al. [25]. A series of semantically meaningful states related by temporal and logical relations are represented by the PN model. A successful traversal of the conditions encoded by the model indicates an event has been recognized.

M. Albanese et al. [27] proposed a plan-based PN approach for the recognition of human activities. In this paper, a probabilistic extension to the PN framework is used to allow the framework to cope with the uncertainty and errors inherent to the low-level observation layers in many video event recognition systems.

Object-based Petri Nets represent the evolution of object states within the video sequence. Events are defined as known transitions between these object states. N. Ghanem et al. [26] proposed using such PN models for mining of surveillance video. This formalism enables specification of the semantic structure of events using domain knowledge. Hierarchical structure, as well as spatial, temporal, and logical relations can be specified in a straightforward manner.

G. Lavee et al. [30] discussed how PN event models can be constructed from the ontology descriptions of the event domain.

III. THE PETRI NET FORMALISM

A. Graphical Representation

The Petri Net model is represented as a bipartite graph that is comprised of two node types (places and transitions) and two arc types (regular and inhibit arcs). In a graphical representation, the places are drawn as circles and the transitions are drawn as squares or rectangles. Arcs are drawn connecting place nodes to transition nodes (input arcs) or transition nodes to place nodes (output arcs). Regular arcs are drawn with arrow heads. Inhibit arcs are drawn with dot heads. Arcs are associated with a weight, also called the arc’s multiplicity. Arc multiplicity is taken to be one if not specified. Places connected to a transition by input arcs are called that transition’s input places (or input set). Similarly, places connected to a transition by output arcs are called that transition’s output places (or output set). A place node may contain a number of tokens (another graph component). Tokens are visualized as black dots within the place node which contains them.

At any given time, the state of the PN model is defined by the number of tokens in each of the model’s place nodes. This configuration is called a marking.

Transition from one marking to another occurs when one or more transition nodes “fire.” Transition nodes can only fire once they become “enabled.” A transition’s “enabling rule” is considered satisfied when: 1) all the input places connected to the transition by regular arcs contain at least as many tokens as the multiplicity of the arc connecting them to the transition, and 2) all the input places connected to the transition by inhibit arcs contain a number of tokens strictly less than the multiplicity of the arc connecting them to the transition. When the enabling rule is satisfied a transition is considered “enabled.” In extensions to the PN formalism, an enabling rule can be modified to contain conditions on the properties of tokens in the input places.

Once a transition node becomes enabled it may fire. The modification to the state of the PN model resulting from the firing of a transition is defined by a “firing rule.” A transition’s firing rule is defined as: 1) removing a number of tokens from each of the transition’s input places equal to the multiplicity of the arc connecting that place to the transition, and 2) creating a
number of new tokens in each of the output places equal to the multiplicity of the arc connecting that place to the transition. Depending on the PN structure, this situation can create a conflict, as firing of one transition may immediately disable another transition. Conflicts may be resolved in a controllable way by adding a priority parameter to each transition node. In case of conflict, the higher priority transition will always fire before the lower priority one.

The dynamic behavior of the PN model is defined by the enabling and firing rules associated with each transition node in the model. For a formalized comprehensive treatment of the PN model interested readers are referred to [31, ch. 2]. Timed Petri Nets extend the PN model by introducing timed transitions which have an associated duration parameter describing the interval of time needed to “carry out” the state change described by the transition. A timed transition may only fire if the period of time since it became enabled is greater than this duration parameter. The duration parameter is sometimes known as a “firing delay” [31, ch. 3].

In some cases, a PN model with strictly defined timed transition delays does not accurately describe the behavior of the underlying system. Another extension to the PN model that adopts timed transitions with stochastic distributions over the duration parameter is known as a stochastic Petri Net (SPN). Each timed transition duration is modeled using the negative exponential probability density function. A parameter indicating the rate of the distribution (the inverse of the mean) is associated with each individual transition.

In modeling real dynamic systems it is desirable to use immediate transitions together with timed transitions. This formalism is called a generalized stochastic Petri Net (GSPN) [31, ch. 4].

B. Marking Analysis

Prior to involving any PN model dynamics, it is possible to compute the set of all markings reachable from an initial marking and all the paths that the system may follow to move from state to state. The set of all reachable markings defines the reachability set of the PN graph. The reachability graph consists of nodes representing reachable markings and arcs representing possible transition from one marking to another. The reachability set represents all possible model states and the reachability graph represents all possible state transitions. However, the reachability graph is only finite if the model and the number of involved tokens are finite. In practice, this means that it is possible to calculate the full reachability graph only if we know the maximum number of the expected tokens in the system. This information is not always available which limits the extent a reachability graph of this sort can be used.

For this reason, in our approach we apply a dynamic marking analysis that only considers markings which appear in the training set (see Section V).

C. Relation Modeling With GSPN

A major strength of the GSPN formalism is its ability to model many of the semantic relations that are used by humans to define video events. These include logical, temporal, and spatial relations. Due to space restrictions we will not discuss the details of representing each of these relations in this paper. Section VII offers an illustrative example of a GSPN model construction which uses many of these relations. For more information regarding modeling semantic relations in the GSPN framework, interested readers are referred to [31].

IV. VIDEO EVENT REPRESENTATION IN SERF-PN

Our representation model follows the Object-based PN paradigm. Within this framework, the PN components have the following roles.

1) Tokens represent actors or static objects detected in the video sequence. Each token has a set of properties associated with it corresponding to object properties.

2) Places represent the possible object states. Each place containing more than one token indicates a group of objects in the same state.

3) Transitions represent the dynamics of the event model. Transition node firing can be equivalent to the object state change in the real scene or can be the result of a satisfied relation constraint.

Examples of these modeling practices are given in Section VII.

V. LEARNING THE SYSTEM PARAMETERS

In SERF-PN, the structure of the model is static and designed manually using knowledge of the domain and of the PN formalism by a knowledge engineer. Initial work toward automating this process to remove subjective interpretation of the domain, has been done in [30]. Parameters of the system, however, are dynamic and may be learned to assist us in predicting events based on the current system state. One set of these parameters includes duration parameters (i.e., exponential distribution rates) for each of the timed transitions in the PN model. Another set of parameters is the transition likelihoods between directly reachable markings.

To learn these parameters we require a set of examples which cover the typical occurrences in the domain of interest as well as a specification of our PN event model. In most surveillance tasks the existence of such a set of examples is a realistic assumption due to the repetitive nature of many events in the surveillance domain. Note also that the knowledge used by the engineer in constructing the static structure of the model is also based on experience with typical events.

A. Timing Parameters

The probability density of the firing delay of each timed transition is given by the PDF function

\[ D_n = 1 - e^{-\mu_n t_n} \]  

(1)

where \( t_n \) is an enabling period of timed transition \( n \) and \( \mu_n \) is an average delay of timed transition \( n \).

During the training process we wish to estimate the \( \mu_n \) parameters for each transition \( n \). This is achieved by running
the model with the timed transitions disabled from firing. We then collect statistics on how long (the number of frames) each of these transitions is enabled over the course of the training sequence. The mean of the length of these enabled periods is then used to estimate the parameter \( \mu_{m_0} \).

Once these parameters are learned the system can model the occurrence of a timed event relative to the training data.

### B. Marking Transition Probabilities

The combination of the PN model and the initial marking allows the construction of a reachability graph for the domain. Each marking node in this reachability graph represents a legal state of the system. An outgoing link from marking node, \( M_k \), to another marking node, \( M_l \), indicates that marking \( M_l \) is directly reachable from the marking \( M_k \). A graphical illustration of a reachability graph can be found in [32].

The reachability graph defines the space of possible states within the model. The probability mass in this space, however, is not uniformly distributed. In fact, the majority of the probability mass is concentrated in just a few markings relative to the large number of possible markings. In other words, while the theoretical number of possible markings is large (infinite in the case of an open model), in practice only a small subset of those markings is observed.

More specifically the possible number of markings for a PN model with \( n \) place nodes and \( t \) tokens is given by the recursion formula:

\[
\Pi(t, n) = 1 + \sum_{i=1}^{t} \Pi(i, n-1)
\]  

where the base case is

\[
\Pi(\epsilon, 1) = 1.
\]

The probability distribution over the marking set can be estimated using observed data, which may be viewed as a sample from this distribution. This is contingent upon the sample being representative of the underlying distribution, which we have claimed is a reasonable assumption in surveillance applications.

Another simplifying assumption made at this stage is the Markov assumption, which states that at each discrete time slice the current marking depends only on the previous marking. This allows us to construct a discrete time Markov chain (DTMC) representing the joint probability of a sequence of particular markings. This formulation may be used to predict future states or answer queries on the probabilistic distance from a particular marking.

Our proposed approach is to construct this DTMC dynamically from the training data. Again, we assume that the training sequence is a representative sample of the underlying probability distribution over event patterns. Thus, the training sequence covers the significant reachable markings in this domain. Each encountered marking in the training sequence corresponds to a state the Markov chain may take on. The probabilities for state transitions in the Markov chains are then estimated directly from the training data according to the formula

\[
\lambda_{i,k} = \frac{N_{i,k}}{N_i}
\]

where \( \lambda_{i,k} \) is the probability of moving to marking \( M_k \) from marking \( M_i \). \( N_{i,k} \) is the number of observed transitions from marking \( M_i \) to marking \( M_k \), and \( N_i \) is the number of \( M_i \) marking occurrences observed.

### VI. Surveillance Using SERF-PN

Our system was designed to aid in the human task of monitoring surveillance video. As such, given a description of the events of a surveillance domain (in a PN format) and an unlabeled (annotated) video sequence, the system should be able to (1) accept video sequences annotated with mid-level semantic concepts, such as object locations, trajectories, and classes. Many works in object detection, tracking, and classification focus on achieving this annotation from raw video data. In this paper, we have put the focus on event analysis and representation and have made use of existing low-level methods to create this annotation from real video sequences. This is referred to as the intermediate video processing unit.

To make our system modular to the process that created the video annotation we have made the video event recognition component of SERF-PN fully compliant with the CAVIAR [1] standard for annotation of video. Thus, the CAVIAR dataset (which has been manually annotated) and any other dataset in this format can be easily evaluated by our system.

The standardization of the video annotation format also allows the use of a synthetic video tool called video scene animator. This module takes object parameters as input and generates a CAVIAR compatible annotated video. This tool has been used to efficiently simulate real video sequences captured by a video camera, allowing us to extend the available dataset of real video clips. Furthermore, to train the parameters of the event model a large volume of "normal" video sequences is needed. Video Scene Animator, running in scenario mode, allows efficiently generating a number of similar scenes whose variance in object location and other properties is defined by a set of parameters. This feature may be used to generate the data required to train the system.
parameters from a small set of representative sequences (along with real videos). A large body of data also allows us to better evaluate the system on combined sets of real and synthetic annotated video sequences.

B. Event Modeling

The event modeling component allows the knowledge engineer to construct the PN event model using a graphical interface. All place and transition nodes are specified according to knowledge of the domain along with enabling rules corresponding to each transition. Stochastic timed transition parameters may be specified or left to be learned from the data using the training mode of the video event recognition component.

A visualization of a model created using the event modeling component of SERF-PN is seen in Figs. 1 and 2. The place and transition nodes and their connections can be seen in the figure. Associated enabling rule conditions are listed in the corresponding Tables I and II, respectively.

The conditions described in these tables specify the properties of the input tokens necessary for the associated transition to become enabled. This information, along with the model structure, defines the full event model.

Detailed examples of constructing event models are offered in Section VII.

C. Video Event Analysis

The video event recognition component of the SERF-PN system is used to aid a human analyst in monitoring video sequences online or used to annotate video sequences offline for content-based retrieval.

The input to this system is the PN event model, along with the CAVIAR format annotated video sequence. All objects appearing in the video annotation are entered as tokens into the PN event model in a specially marked "root" place node. From there, transitions which become enabled according to their enabling rules are allowed to fire. As the video sequence processing progresses the properties of tokens corresponding to objects are updated to reflect the properties of the object. Transitions which represent events of interest (indicated as such in event model) are output to the video analyst (or stored as annotation for the video sequence in an offline application). This annotation includes information on the event occurred (transition), the objects involved (tokens), and the frame number in the sequence.

If available, marking analysis data can be used to predict what is the next likely state of the system or what is likelihood of an abnormal or suspicious event given the current configuration. Currently, the most probable next system state is output along with the event annotation.

The results of the interpretation are presented in the log window of the graphical user interface and may then be stored in a text file.

To learn the parameters of the models the video event recognition module may run in a training mode. In the training mode, this module disables timed transitions and learns their parameters by observing their average enabling time. It also constructs the dynamic reachability graph and collects statistics on transitions between markings (see Section V).

VII. Constructing A PN Model: Illustrative Examples

In this section, we provide two comprehensive examples of using the qualities of the GSPN formalism described in previous sections to construct a model describing the semantics in a particular event domain. Our construction remains firmly within the Object PN paradigm. That is, each token corresponds to a scene object, each place node corresponds to an object state, and each transition corresponds to a change in object state or a relation verification.

It is important to note that a PN model structure along with the conditions defined on each of the transitions within the model is a reduction of a human decision process in a particular event domain. As such, it may be possible to define several different models that provide a reasonable semantic summary of an event domain. Determining if a particular model is sufficiently accurate can be done empirically by comparing the results of the model applied to a particular video interpretation to the interpretation provided by a human observer.
In this section, we consider the scenario where a single person guards the entrance to a particular location and is tasked with checking the bags of all those who enter the area in a time-efficient manner. We are particularly interested in two main events.

1) A visitor has entered the area without being checked.
2) The duration of the security check is excessively long.

To describe this scenario using the PN formalism we define two semantic areas within our camera view: the “Guard Post” zone and the “Not Checked” zone. The former zone is rather self-explanatory, it is the area in the camera view taken by the guard. The latter zone refers to an area in the camera view to which a visitor should only gain access after being checked.

All objects detected by the system are mapped to tokens and each of the object’s attributes is associated with the corresponding token. In the remaining part of this discussion, we will refer to objects and tokens interchangeably. Initially tokens are placed in the Root place node. The transition Person Appeared becomes enabled when any tokens that meet its enabling condition are present in the Root place node.

Upon firing of this transition the corresponding input tokens be removed from the place and the token of the output place is created. The transition is enabled when the appearance attribute of the input token takes on the value appear.

### Table I

<table>
<thead>
<tr>
<th>Transition</th>
<th>Name</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Person Appeared</td>
<td>Token.appearance=appear</td>
</tr>
<tr>
<td>T2</td>
<td>Person Disappeared</td>
<td>Token.appearance=disappear</td>
</tr>
<tr>
<td>T3</td>
<td>Guard Post Manned</td>
<td>Token.loc overlaps “Guard Post” Zone</td>
</tr>
<tr>
<td>T4</td>
<td>Guard Post Unmanned</td>
<td>Token.loc not overlaps “Guard Post” Zone</td>
</tr>
<tr>
<td>T5</td>
<td>Visitor Not Checked</td>
<td>Token.loc overlaps “Not Checked” Zone</td>
</tr>
<tr>
<td>T6</td>
<td>Visitor Entered</td>
<td>Token.loc overlaps “Guard Post” Zone</td>
</tr>
<tr>
<td>T7</td>
<td>Guard Meets Visitor</td>
<td>distance(Token(p3).loc,Token(p2).loc) &lt; Tmeeting Unsafe</td>
</tr>
<tr>
<td>T8</td>
<td>Meeting Is Over</td>
<td>distance(Token1(p3).loc,Token2(p3).loc) &gt; Tmeeting Safe</td>
</tr>
<tr>
<td>T9 (timed)</td>
<td>Security Check Too Long</td>
<td>distance(Token1(p3).loc,Token2(p3).loc) &gt; Tmeeting Safe for a period exceeding a value sampled from D9</td>
</tr>
<tr>
<td>T10</td>
<td>Meeting Is Over (after Delay)</td>
<td>distance(Token1(p3).loc,Token2(p3).loc) &gt; Tmeeting Safe</td>
</tr>
<tr>
<td>T11</td>
<td>Guard Post Manned (After Check)</td>
<td>Token.loc overlaps “Guard Post” Zone</td>
</tr>
<tr>
<td>T12</td>
<td>Visitor Continues (After Check)</td>
<td>Token.loc not overlaps “Guard Post” Zone</td>
</tr>
<tr>
<td>T13</td>
<td>Person Disappeared (After Check)</td>
<td>Token.appearance=disappear</td>
</tr>
</tbody>
</table>

### Table II

<table>
<thead>
<tr>
<th>Transition</th>
<th>Name</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Car Appears</td>
<td>Token.appearance = appear</td>
</tr>
<tr>
<td>T2 (a, b, c, d)</td>
<td>Car Approaching Intersection (from North, South, East, West)</td>
<td>Token.loc overlaps one of 4 “Approach” Zones</td>
</tr>
<tr>
<td>T3</td>
<td>Car Enters Intersection</td>
<td>Token.loc overlaps “Intersection” Zone</td>
</tr>
<tr>
<td>T4</td>
<td>Unsafe Interaction Occurring</td>
<td>Tokens1.p1.orientation is perpendicular to Tokens2.p1.orientation</td>
</tr>
<tr>
<td>T5 (timed)</td>
<td>Safe Interaction Occurred</td>
<td>Tokens1.p2.orientation is perpendicular to Tokens3.p1.orientation for a period exceeding a value sampled from D5</td>
</tr>
<tr>
<td>T6</td>
<td>Car Leaves Intersection</td>
<td>Token.loc not overlaps “Intersection” Zone</td>
</tr>
<tr>
<td>T7</td>
<td>Car Continues (After Safe Interaction)</td>
<td>Token.loc not overlaps “Intersection” Zone</td>
</tr>
<tr>
<td>T8</td>
<td>Car Leaves Intersection (After Safe Interaction)</td>
<td>Token.loc not overlaps “Intersection” Zone</td>
</tr>
<tr>
<td>T9</td>
<td>Car Leaves Intersection</td>
<td>Token.loc not overlaps “Intersection” Zone</td>
</tr>
<tr>
<td>T10</td>
<td>Car Disappears</td>
<td>Token.appearance = disappear</td>
</tr>
</tbody>
</table>
will be removed from the Root place node and placed in the P3 place node (see Fig. 1).

The Guard_Pass_Manned transition is similarly conditioned on the loc (object location) attribute of its input tokens. If an input token located in the Guard Post Zone exists, the transition will be allowed to fire. The transitions VisitorEntered, and Visitor_Not_Checked are similarly defined by conditions tied to the Location attribute.

The Guard_Met_Visitor transition requires input tokens in two separate place nodes (representing the guard and the visitor). It also has a condition on the distance between these two tokens’ locations. If this distance is below a certain threshold, denoted \(T_{\text{Meeting}}\), this transition is allowed to fire. In firing, it moves both of its input tokens to its output place node, P0.

Transitions Meeting_Is_Over and Meeting_Is_Over(After Delays) are defined similarly with a maximum threshold, denoted \(T_{\text{Meeting}}\), at and T_Met. Security Check Too Long is a stochastic timed transition with parameter \(\mu_9\) which indicates the average firing delay of the transition. This transition becomes enabled when two tokens are in input place P0 (the two-token requirement is captured by assigning a multiplicity of two to the input arc of this transition). Once the enabling condition is met the transition may only fire once a period of time, sampled from the distribution \(D_9\) (which is determined by parameter \(\mu_9\)), has elapsed. Note also that this transition becomes enabled at the same time as the transition Meeting_Is_Over. The fact that firing one of these transitions will disable the other, is called a conflict in Petri Net terminology. Our condition on the length of the meeting uses this. That is, once the security check has begun we will only allow a specific time interval (determined by parameter \(\mu_9\)) to elapse before alerting that the security check has gone on too long. Of course, if the meeting ends before this interval elapses, we do not need to provide such an alert.

The remaining transitions Guard Post Manned(After Check), Visitor_Constant(After Check), and Person Disappeared(After Check) are immediate conditional transitions conditioned on the loc and appearance attributes of their input tokens.

The PN graph corresponding to this model is illustrated in both Fig. 1 and 2. The transition names and conditions are enumerated in Table II.

Due to space restriction we will discuss only some of the designs of some of the transitions in this model. A more detailed description of the model design, please see [32].

The stochastic timed transition Safe Intersection Occurred becomes enabled when there exists a token in each of the two place nodes P2 and P3 (representing a car inside the intersection and a car approaching the intersection, respectively.) If this transition remains enabled for an interval of time, whose length is sampled from distribution \(P_0\) (determined by parameter \(\mu_9\)), it will be allowed to fire. Again, we place our timed transition in conflict with an immediate transition (two in this case). In this model, the firing of either the Car_Entered Intersection transition or the Car_Leaving_Intersection transition will disable the timed transition. These transitions check if the loc attribute is inside/outside the Intersection zone.

The Unsafe_Interaction_Occurred transition has an input arc of multiplicity two. This means at least two tokens must exist in its input place node, P5, for this transition to become enabled. Furthermore, it has a condition on the difference in orientation attribute values of each of the tokens participating in the enabling. The orientation attribute values must be perpendicular (a difference of 90 ± 15°) for this transition to fire.

If the “reasonable” amount of time has elapsed between the time a particular car has entered the intersection in the path of another oncoming car (i.e., Safe Intersection Occurred), our knowledge of the event domain states that an unsafe situation is less likely to occur and hence we can allow perpendicular trajectories within the intersection (e.g., a car is exiting to the west just as another enters from the north).

VIII. COMPLEXITY ANALYSIS

In this section, we aim to give a formal treatment to our algorithm for event recognition using the Petri Net modeling formalism. To this end we must define some notions and concepts. The notions described in the following are based on notations in [27], [31].
First, we will define a video sequence as the set of frames \( F \), the function \( p \) which maps a particular frame to a set of objects in the video, and the function \( l \) which maps a particular set of objects to a set of features. An example of possible features would be the CAVIAR format features which include continuous valued variables \( h, w, x, y \) (respectively representing the height, width, and centroid co-ordinates of the object bounding box) as well as discrete finite-domain categorical variables, such as movement, role, situation, and context. Essentially, the \( l \) function allows us to retrieve the semantic properties of each object (or set of objects).

A Petri Net event model can be described using \( P \), the set of all places and \( T \) the set of all transitions. Additionally, the function \( \text{flow} \) indicates the arc flow of the network. That is, if \( y \in \text{flow}(x) \) there is an arc from node \( x \) to node \( y \). The \( \delta \) function describes the condition associated with each transition by associating each transition \( t \) with a set of possible feature values. That is a set of objects, \( p \), whose labels \( l(p) \) are a subset of \( l(t) \) can be said to fulfill the condition of transition \( t \). Function, \( r \), returns the arity of a particular transition. The arity is defined as the number of tokens to which the transition’s condition is applied. For example, a transition which verifies a single object/token property such as \( \text{appearance} = \text{“visible”} \) has an arity of one. Similarly, a transition that verifies a relationship between two objects (tokens), such as \( \text{distance}(t1, t2) > \text{thresh} \) has an arity of two. Finally, the variable \( \text{rootnode} \) indicates the root place node in which tokens corresponding to new objects are initially placed. Table III summarizes the notation used in this section.

The algorithm for detecting a particular event in a video sequence is presented in pseudo code in Algorithm 1. The algorithm accepts as input the video sequence and PN model parameters, as well as the variable \( \text{event\_transition} \), which indicates which transition corresponds to the particular event we are interested in detecting. After initialization of temporary variables, the algorithm proceeds to loop over all frames in the video. At each frame the algorithm assigns tokens in the network to each object, determines which transitions are enabled, and loops over these enabled transitions to determine if their conditions for firing have been met. In order to make this determination, the algorithm must check each possible combination of tokens of size \( y \) in the input set of each transition, where \( y \) is the arity of the transition’s condition. Thus, we have a loop over all possible (size \( y \) ) combinations of tokens nested inside a loop over all enabled transitions, nested inside a loop over all frames in the video sequence. The number of outer loops is bounded by \( |F| \), the number of enabled transitions is bounded by the total number of transitions \( |T| \) which is a known small constant relative to the number of frames in the average input sequence. The number of token combinations of size \( y \) is given by \( \binom{|F|}{y} \), where \( k \) is the number of tokens in the transition’s input set. Clearly \( k \) is bounded by \(|K| = |O| \), the number of objects/tokens in our net. \( y \) in this expression is bounded by \( Y = \max_{t \in T} |r(t)| \). Thus, the worst case complexity of this algorithm is given by \( O(|F|) \binom{|F|}{y} \). That is, the running time is dependent on the number of objects and the maximal arity of the transition conditions. In practice, however the number of objects simultaneously appearing in a frame is very small compared to the number of frames in the video sequence. In the experiments reported in this paper the maximal number of objects in a single frame was 7. On average, this number is significantly less than this. Similarly, when designing our PN event models we seldom used transition conditions greater than two (e.g., \( \text{Guard\_Met\_Visitor} \)). Most often, we made use of transition conditions with an arity of one (e.g., \( \text{Car\_Entered\_Intersection} \)). Thus, under the reasonable assumption that both \(|K| = |O| \) and \( Y \) are small compared with the length of the video sequence in frames.
Detecting A Particular Event in A Video Sequence

Algorithm 1

Input: F, l, p /*video and object labeling
P, T, flow, k, r, rootnode /*PN model
event_transition /*transition corresponding

//Do particular event we are interested in

Output: true if event occurs in video sequence, false otherwise

1: //Initialization:
2: K ← ∅
3: for all places p in P do
4: μ(p) ← 0
5: end
6: //Add tokens for appeared objects:
7: for all objects o in μ(f) \ (K ∩ μ(f)) do
8: k ← new token
9: K ← K ∪ {k}
10: g(k) ← o
11: f(o) ← k
12: μ(rootnode) ← μ(rootnode) ∪ [k]
13: end
14: //Determine enabled transitions:
15: enabled_transitions ← ∅
16: for all transitions t in T do
17: enabled ← true
18: for all places p in t do
19: if μ(p) = 0
20: enabled ← false
21: end if
22: end
23: if enabled = true
24: enabled_transitions ← enabled_transitions ∪ t
25: end
26: end
27: end
28: //Fire transitions
29: for all in enabled_transitions do
30: Q = μ(t) //all tokens in t’s input set
31: H = ∅
32: for all q in Q do
33: H ← H ∪ (g(q) \ H) //if objects corresponding to tokens in t’s input set
34: end
35: y = r(t) //arity of transition t’s condition
36: for all subsets of objects in G(y, H) do
37: if k(p) ≤ k(t) //objects meet transition condition
38: do
39: k ← f(o)
40: for all places m in t do
41: if k ∈ μ(m)
42: μ(m) ← μ(m) \ [k]
43: break
44: end if
45: end
46: end if
47: end
48: end
49: for all places m in t do
50: μ(m) ← μ(m) ∪ {k}
51: end
52: end
53: if t = event transition
54: return true
55: end if
56: end if
57: end
58: end
59: end
60: return false

In this section, we will evaluate the merits of our approach in a real world video event domain. This is achieved by studying how the interpretation results output by the system based on a particular PN event model compare to a human
For our first two experiments we captured a number of real video clips containing our events of interest. We then applied a tracker, recently developed in our lab [34], to obtain the tracking information. This tracker makes use of constancy of color and color edge features. As expected, the tracking results were good but not without error. Object tracks were occasionally lost due to occlusions, lighting variation, and other reasons.

In parallel we also generated a set of animated clips corresponding to the video events captured on camera. This allowed us to increase the volume of data available to test our system and to introduce unusual events that are improbable in real video (but which are usually the object of interest). In generating these animated clips some randomness was added in the object sizes and locations, mimicking the variability observed in the real videos.

As stated previously, the ground truth video data was obtained by allowing a human observer to determine whether a specific event is occurring in a video sequence. The system output for a particular clip is then checked for this particular event.

The third experiment is intended to compare our approach to other representation and recognition approaches. In this experiment, we used available ground truth information in place of the tracker to supply low-level object information, such as location and bounding box properties. We used high-level ground truth information to evaluate the classification results of our model.

### A. Security Check

For the security check experiment we had a total of 19 real video clips and 100 animations, each containing events of interest. Clip lengths ranged between 11 and 48 s. The mean and median lengths over all clips used in this experiments were 23 s. In this experiment we have defined two events of interest among the possible events: “Security Check Too Long” and “Visitor not checked.” Not all clips include one of these events of interest. Some clips included multiple events of interest. More specifically in the real video clips both the “Security Check Too Long” event and the “Visitor not checked” event occurred five times. They occurred within the same clip twice. In the animations the “Security Check Too Long” event occurred 43 times and the “Visitor not checked” event occurred 14 times. The events occurred together in the same synthetic video sequence seven times. We built the model as described in detail in Section VII and calculated precision and recall statistics on our events of interest. In Table V, we report the results. This experiment was carried out on both the set of real videos with tracking info, the set of animations only and the combined data set. The length of the average activation was manually set using knowledge of the event domain (\( \mu_s \approx 20 \) s).

It is important to note that treating this model as an event occurred/haven’t occurred classifier was done for evaluation purposes and that the output of the system is, in fact, a semantic description of the events in a particular video sequence. Such a semantic description is shown in Table IV. Qualitative assessment of this description shows it to be quite similar to a possible human description of the same video sequence. As the table indicates the results are quite good in both the animation set, which corresponds to perfect tracking and detection, and in the real video dataset with the automatic tracker applied which is less than perfect. Our experiments showed that minor tracking error has little effect on the high-level analysis results. While an object completely lost by the tracker cannot be reasoned upon by the high-level system, the system can still provide an analysis up to the point of object loss. That is, we can still get meaningful semantic description from only partially correct tracks.

Prediction using the mechanism of marking analysis can also be illustrated using this example. We applied a dynamic marking analysis using our dataset of real and animated clips. The marking graph generated with meaningful names for each marking and associated transition probabilities is shown in Fig. 3. This figure reduces the full marking graph by consolidating markings with similar semantic meanings and neglecting markings which are reachable with very low probability. Note that in the full marking graph outgoing arrows must sum to one. Hence, we renormalize all remaining outgoing arcs from the same marking node to sum to one. A marking node with no outgoing arrows indicates that the system did not observe a transition from this marking to another during the training process.

Using this information to construct a Markov process as described in Section V-B allows answering such queries as...
what is the next most likely state given the current state and what is the probability of a particular state given the current state. For example, note that given that a check of a visitor has started there is only a small probability that the check will be too long and a moderate possibility of an unchecked visitor. However, once the check has gone over the time limit, the probability of an unchecked visitor entering the secure area increases.

It is also interesting to note that although the theoretical number of markings is exponential in the number of tokens, by using the dynamic marking analysis we can concentrate on the markings in which most of the probability mass is concentrated. In our example, if we assume three actors only, out of a possible 120 markings (obtained by plugging $\Pi(3, 8)$ into (2)) our marking analysis observed only 18 markings in our training data which we reduced to 6 with semantic meaning and significant probability mass.

B. Traffic Intersection

The traffic intersection scenario is slightly more complex semantically than the previously discussed security check. We built the model as described in detail in Section VII. The parameter of the stochastic timed transition modeling a safe interaction between two vehicles with collision course trajectories may not be obvious even to an expert in the domain. To this end, we apply our timed parameter training approach as discussed in Section V-A. We applied this method to several subsets of available data (animations and real) using 4-fold cross validation. That is, we divided the available labeled data across in the real video sequences.

It defines a strict semantic hierarchy of events. In the CAVIAR ground truth [35], each object is assigned a movement, role, situation, and context. Each of these variables can take on a value from a discrete set of values. Also each of these categories represents a more specific/general semantic level of information. Movement is the lowest level of information and can take on the states running, walking, active, inactive, or unknown. Role describes the role an object plays in a higher level event (Situation or Context). Role can take on the states fighter, left object. Situation is a higher semantic level and can take on values, such as moving, browsing, shop enter, shop exit and so on. Context can be described as the overarching semantic purpose of the object in question. In the CAVIAR ground truth each object is assigned only one context throughout the duration of its appearance in the scene. Context values may be shop enter, shop exit, shop-re-enter, walking, windowshopping, etc. CAVIAR also defines each particular context using a number of situations. For instance, the windowshopping context may consist of the situations moving, browsing, and then moving. That is we can define situations using the movement information, and contexts according to the situation information.

C. CAVIAR Dataset

In our third experiment, we evaluated our approach to modeling events on a portion of the CAVIAR dataset [1], which includes surveillance video of a shopping center in Lisbon, for the purpose of comparing our approach to others tested on the same dataset. The CAVIAR (context aware vision using image-based active recognition) dataset covers the potential events that can occur in a shopping mall surveillance context. It defines a strict semantic hierarchy of events. In the CAVIAR ground truth [35], each object is assigned a movement, role, situation, and context. Each of these variables can take on a value from a discrete set of values. Also each of these categories represents a more specific/general semantic level of information. Movement is the lowest level of information and can take on the states running, walking, active, inactive, or unknown. Role describes the role an object plays in a higher level event (Situation or Context). Role can take on such states as fighter, left object. Situation is a higher semantic level and can take on values, such as moving, browsing, shop enter, shop exit and so on. Context can be described as the overarching semantic purpose of the object in question. In the CAVIAR ground truth each object is assigned only one context throughout the duration of its appearance in the scene. Context values may be shop enter, shop exit, shop-re-enter, walking, windowshopping, etc. CAVIAR also defines each particular context using a number of situations. For instance, the windowshopping context may consist of the situations moving, browsing, and then moving. That is we can define situations using the movement information, and contexts according to the situation information.

TABLE VI

<table>
<thead>
<tr>
<th>Safe Interaction Occurred</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real</td>
<td>0.91</td>
<td>0.75</td>
</tr>
<tr>
<td>Animated</td>
<td>0.80</td>
<td>0.77</td>
</tr>
<tr>
<td>Combined</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>

The Recall and Precision for the Traffic Intersection Model.
As opposed to our previous experiments, where we attempted to classify whether an event has occurred or not (i.e., a particular transition has fired), in this experiment we attempt to classify the state of each object at every frame (i.e., the place node the object token is residing in). This is an equivalent problem as it is the events (i.e., transitions) that cause the changes in object states (place nodes containing tokens). We have chosen to present our results in this fashion to allow a more straightforward comparison.

Thus, to evaluate our PN representation approach we built two PN Models, the first to evaluate the situations based on object location and movement information. In principle, this information can be given by a tracker, as in our previous experiments, but for comparison purposes we have chosen to use the ground truth information provided as done in other approaches.

The other PN model evaluates the context of each object using the object information along with the situation label obtained from the ground truth.

The situation and context networks are shown in Figs. 4 and 5, respectively. Due to space constraints we will not go into the details of constructing these networks. For more details regarding their construction please see our technical report [32]. For a detailed example of constructing a PN event model, see Section VII.

The situation of an object may change throughout the scene, and so a frame by frame evaluation of the resulting situation against the situation label given in the ground truth is warranted. To this end we compared the situation label against the label of the place node in which the token corresponding to the object resided. Of course, each of the place node labels in this network corresponds to a possible situation value.

Context may also be measured in this frame by frame manner. However, when considering some types of contexts it is not reasonable to expect their detection from the first appearance of the object in the scene. For example, a context such as enter store can only be detected (by either a human or an automated system) when the object enters the store. Thus, when compared to the Oracle-like nature of the ground-truth context on a frame by frame basis even the best system would produce low classification performance on these types of context.

For this reason it is beneficial to use the frame by frame situation classification in a more sophisticated way to determine the context for the entire duration of the object’s existence within the scene. In our experiments we report results on both the frame by frame classification of the context and a simple but effective method of looking at the entire sequence to determine the overall context, using the “last frame” context classification.

For comparison we offer two approaches with results described in [1]. The first is that of a rule-based approach. This approach used semantic rules on both the role and movement classifications in the ground truth to arrive at situation and context classifications. This approach is somewhat similar to the use of semantic knowledge in our approach, but is missing the basic formalism (i.e., Petri Net) to allow us to model these rules efficiently and determine if they apply in a straightforward manner. This rule base also lacks the notion of states intrinsic to Petri Net markings. All this means that rules applied are less powerful and robust and less likely to capture the semantics of the event domain appropriately.

Another approach is that of a probabilistic model with parameters that can be tuned to the event domain and offers a probabilistic interpretation of the uncertainty in the event classification. Another strength of the probabilistic approach is its ability to accept soft probabilistic evidence as opposed to hard evidence expected by semantic rule-based approaches, including our approach. The particular probabilistic graphical model explored for the CAVIAR context interpretation problem is the hidden semi-Markov model (HSMM) [36], [19]. HSMMs extend the standard Hidden Markov model with an explicit duration model for each state. Using the situation values as the possible values of hidden states, and training a separate HSMM for each context model, the model is able to answer questions like what is the most likely state sequence given a particular observation, and also which is the most likely context model. Apart from the results offered in [37] the HSMM approach for event detection is also described in [36]. Essentially, the probabilities are used to determine which is the most likely context given a sequence of situations.

We ran our experiments on a total of 26 CAVIAR sequences ranging in length from 10 to 123 s. The mean length of these sequences was about 47 s and the median length about 49 s. Overall, we considered 235 objects, 5 possible situations, and 7 possible contexts. There were several sequences where no
objects were visible for a significant portion of the sequence, as well as several sequences where a number of objects appeared simultaneously. We built two PN event models, one to classify situations and one to classify contexts.

For our first experiment we evaluated the PN model’s ability to classify situations per frame. Each object’s situation was set to be equal to the label of the place node in the PN model in which the token corresponding to this particular object is residing. The accuracy of this classification is obtained by comparing this label to the situation label in the ground truth.

The confusion matrix for this experiment is shown in Table VII. Overall the frame by frame situation classification accuracy achieved is 91%. Another observation that can be made by examining the table is that some situations are better detected than others. Specifically, “shop enter” is often confused with other situations, such as “walking.” However, this error is subjective to interpretation and as we shall see does not greatly affect the eventual higher level context results that use these situation classifications.

The context evaluations are shown both using a frame by frame approach similar to that of the situation evaluations, and using a simple evaluation over the series of situations, namely selecting the last frame in the object life-span and using the context evaluation in that frame as the context of the object. In this evaluation, we thus evaluate the context of each object against its ground truth, as opposed to the frame by frame approach which evaluates each frame of each object.

For this reason the total contexts evaluated in this “last frame” approach is downgraded a smaller amount than the actual error in classification (10%).

For comparison purposes, we evaluate the “rule-based” and HSMM algorithms for evaluating both situation and context. Table XII shows the results reported in [37] on the CAVIAR dataset as compared to our results on the same set. As is shown in the table our frame by frame situation evaluation is a significant improvement over both the “rule-based” and the HSMM situation results. Although a PN model can be considered a type of rule-based model based on semantic knowledge, the framework it provides allows us to keep track of object states and model relationships, thus allowing for a more robust rule set and hence the improvement in results. With additional semantic information, specifically orientation, and speed information, these results possibly may be improved.
In the frame by frame context evaluation we show results comparable to those of the HMM context classifier. We should note, however, that the HMM classifier does not take a naive approach to classifying each frame and in fact performs inference on all the situation values at each frame to determine what is the most likely context. If we apply a similar approach, although admittedly much simpler, of simply taking the last frame of each object and using the context label as the context label for the entire sequence, our “last frame” classification strategy, we achieve significantly better classification using the PN model approach. This is due to the fact that the classification is made based on a series of rules reflecting semantic knowledge of the domain as opposed to a set of abstract state transition parameters (as in the HMM) with no inherent semantic value.

### X. Discussion

#### A. Comparison to Other Approaches

The GSPN formalism for video event understanding presented in this paper offers a way to apply semantic knowledge to automatic interpretation of video events. It is a somewhat different approach than those that model the statistics of video abstractions, such as hidden Markov models, Bayesian networks, and conditional random fields. In those methods, it would be difficult to recognize events defined by semantic relations, such as “Guard Meets Visitor” or “Car Enters Intersection” due to the fact that these events can have very large variance in their appearance.

Other Petri Net approaches have been suggested. Castel et al. [25] suggests keeping track of a PN “plan” for each

<table>
<thead>
<tr>
<th>TABLE IX</th>
<th>Context “Last Frame” Classification (Using Situation Ground Truth)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Browsing</td>
</tr>
<tr>
<td>Browsing</td>
<td>0</td>
</tr>
<tr>
<td>Immobile</td>
<td>0</td>
</tr>
<tr>
<td>Walking</td>
<td>0</td>
</tr>
<tr>
<td>Windowshop</td>
<td>0</td>
</tr>
<tr>
<td>Shop Enter</td>
<td>0</td>
</tr>
<tr>
<td>Shop Exit</td>
<td>0</td>
</tr>
<tr>
<td>Shop Reenter</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>235</td>
</tr>
<tr>
<td>Overall Accuracy</td>
<td>0.86</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE X</th>
<th>Context Frame by Frame Classification (Using PN Derived Situation Information)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Browsing</td>
</tr>
<tr>
<td>Browsing</td>
<td>0</td>
</tr>
<tr>
<td>Immobile</td>
<td>0</td>
</tr>
<tr>
<td>Walking</td>
<td>0</td>
</tr>
<tr>
<td>Windowshop</td>
<td>0</td>
</tr>
<tr>
<td>Shop Enter</td>
<td>0</td>
</tr>
<tr>
<td>Shop Exit</td>
<td>0</td>
</tr>
<tr>
<td>Shop Reenter</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>132673</td>
</tr>
<tr>
<td>Overall Accuracy</td>
<td>0.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE XI</th>
<th>Context “Last Frame” Classification (Using PN Derived Situation Information)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Browsing</td>
</tr>
<tr>
<td>Browsing</td>
<td>0</td>
</tr>
<tr>
<td>Immobile</td>
<td>0</td>
</tr>
<tr>
<td>Walking</td>
<td>0</td>
</tr>
<tr>
<td>Windowshop</td>
<td>0</td>
</tr>
<tr>
<td>Shop Enter</td>
<td>0</td>
</tr>
<tr>
<td>Shop Exit</td>
<td>0</td>
</tr>
<tr>
<td>Shop Reenter</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>235</td>
</tr>
<tr>
<td>Overall Accuracy</td>
<td>0.81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE XII</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rule-Based Method</td>
</tr>
<tr>
<td>Situation Classification</td>
<td>50%</td>
</tr>
<tr>
<td>Context Classification</td>
<td>57%</td>
</tr>
</tbody>
</table>
event. One of these plans must exist for each object (or set of objects) participating in a particular event. By adopting the Object PN paradigm we avoid large overhead of keeping track of many existing plans that might hamper even a simple model.

Another more recent framework which makes use of the plan PN paradigm is proposed by M. Albanese et al. [27]. This paper offers a probabilistic extension to the framework by assigning each token a probability and updating this probability using the probabilities assigned to fired transition as the token propagates through the network toward the events “terminal” state. At each conflict (two or more transitions sharing an input place) each of the possible transitions is assigned a probability of firing summing to one. To cope with the possibility that none of the transitions will fire a special “skip” transition is attached to each place node that has that place node as both its input and output place.

The ability to cope with uncertain input is important; however due to the inevitable pruning of low-probability token necessary in order to keep the inference in this framework tractable, this framework, in practice, is very similar to the deterministic method taken in our approach. Furthermore, the assumption taken in this approach that a transition fires at each time step (i.e., frame), each time reducing the probability, coupled with the aforementioned pruning enforces an implicit duration model on the event. While in some cases this may be useful, in general, events may have variable length and thus this aspect of the probabilistic framework may present difficulties in recognizing certain types of events. In our approach, we use semantic rules to fire transitions. While this may include duration information (timed transitions), in general we do not assume a specific time frame for every event (transition).

The Object PN paradigm was first introduced in Ghanem et al. [26]. Although our work is based on theirs, we have made several meaningful extensions in our work. Along with the major extensions (parameter learning, and marking analysis), discussed earlier we have embraced an approach that uses a single PN graph to capture a particular event domain as opposed to using a single graph for each possible event. This allows a more condensed representation of our domain and captures the shared properties of many events.

### B. Temporal Segmentation

Many of the recently explored event recognition approaches are actually event classification approaches in the sense that, as input, they are given a video segment in which an event occurs. The problem of event recognition is then reduced to determining which of the known kinds of events is the one that occurred. Probabilistic formalisms, such as the HMM usually take this approach, though some systems adjust for this bias by adding an additional “unknown” or “nothing happened” event model and comparing its computed likelihood on the observation to that of the other known event models.

The approach presented in this paper does not make the assumption that it is presented with a video sequence input that includes a specific number of events, or any events at all. This approach utilizes a single PN model of the event domain to describe everything that can occur and recognizes events by applying semantic knowledge to an object abstraction of the video sequence input. This empowers it to receive as input video sequences that have any number of interesting semantic events included in them. We have seen this in our experiments, from the security scenario in which a visitor evades the guard at the same time as a security check on another visitor is in progress. We have also seen this in our CAVIAR dataset experiments, where we have several objects simultaneously interacting with the environment within the same video sequence in some of the input video sequences, while still others in which there are no objects in the frame at all. In all these cases our approach is able to classify the multiple events of the video sequence correctly. Furthermore, we have seen that the temporal extent of a particular event does not have to be constant in each instant of the event. For example, in our security check experiment we are able to detect particular security check events even as they contain significant variance in their duration.

This robustness to temporal extent and natural temporal segmentation is achieved by defining events using semantic knowledge about the nature of their composition rather than by relying on a statistical model with abstract states whose parameters are tuned in a data-driven process which is much more sensitive to variance in duration of the event.

### C. Limitations of the PN Formalism and Future Work

Dealing with uncertain observations, scaling to larger problems, and automating model learning are all challenges facing the current specification of the PN formalism.

Uncertain observations which are handled well in fully stochastic models, such as HMMs may cause problems in the deterministic setting of the PN formalism. One direction of possible future work to address this is to allow the marking to be represented as a stochastic variable according to probabilities associated with the confidence in the observation of the tokens.

Scalability to larger problems may be handled by considering modular Petri Net fragment analyzing simple situations, nested within one another to consider a more complex situation.

Although a PN model is specified by semantic human knowledge, in some cases semantic relations can be inferred from data using supervised learning techniques. Another area of future work is isolating important relations among scene entities involved in an event and utilizing this information to semi-automate the human process of constructing a PN event model (which is itself based on the human’s supervised learning of semantic relations).

### XI. Conclusion

In this paper, we have proposed SERF-PN, a framework for the representation and recognition of video events based on the PN formalism. This formalism allows straightforward specification of semantic event domain knowledge, including object
properties and relationships. We extended the representational capacity to allow some variance in the duration of the events by allowing parameters of stochastic timed transitions to be learned from the data. We also developed a learning a predictive mechanism using the method of dynamic marking analysis. SERF-PN is aimed at aiding online monitoring of surveillance video and offline annotation for content-based retrieval. This system is based on a PN event representation.

We showed, using extensive examples, how the proposed formalism may be used to construct an event model in a particular domain. We then formalized the algorithm for event recognition and offer a formal complexity analysis which shows that under reasonable assumptions regarding the number of objects continuously appearing in the video sequence, our proposed approach is efficient for this task.

In the experimental section, we illustrated the strengths of the proposed approach to recognize events in unlabeled video and provide a comparison of the results achieved by our approach with those of other recognition approaches on the same dataset.

The PN formalism allows expression of complex semantic knowledge regarding an event domain in a straightforward way. It is our contention that for this reason it is more suitable for representing event domains where there exists great variance in appearance and duration of events. The nature of these events can usually be specified using a few semantic rules. However, the recognition of these types of events is often challenging for formalisms that make use of abstract parameters from data.

REFERENCES


Artiom Borzin received the B.S. and M.S. degrees in electrical engineering from the Technion-Israel Institute of Technology, Haifa, Israel, in 2001 and 2007, respectively. His research interests include video event understanding and behavior analysis.

Gal Lavee received the B.S. and M.S. degrees in computer science from the University of North Texas, Denton, and the University of Texas, Dallas, in 2003 and 2005, respectively. Currently, he is pursuing the Ph.D. degree from the Department of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel. His research interests include video event understanding, knowledge representation, and learning of semantic knowledge.

Michael Rudzsky received the Magister degree in physics in 1970, and the Ph.D. degree in physics and mathematics from the Institute of Space Research, Moscow, Russia, in 1980. He was with the Scientific and Industrial Association for Space Research, Baku, Azerbaijan until 1991. In 1991, he became a Research Fellow at the Department of Physics, Technion-Israel Institute of Technology, Haifa, Israel, and since 1995 he has been with the Department of Computer Science, Technion-Israel Institute of Technology. His current research interests include computer vision, pattern recognition, and compression of images.

Ehud Rivlin received the B.S. and M.S. degrees in computer science and the M.B.A. degree in business administration from the Hebrew University, Jerusalem, Israel, in 1986, 1988, and 1992, respectively. He also received the Ph.D. degree from the University of Maryland, College Park, in 1993. Currently, he is an Associate Professor at the Department of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel. His current research interests include machine vision and robot navigation.