IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 14, 2020, accepted February 16, 2020, date of publication February 19, 2020, date of current version March 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2975095

Multi-Event Modeling and Recognition Using Extended Petri Nets

JI QIU[1], LIDE WANG[1], YIN WANG[1], (Member, IEEE), AND YU HEN HU[2], (Life Fellow, IEEE)

[1]Department of Electrical Engineering, Beijing Jiaotong University, Beijing 100044, China
[2]Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI 53705, USA

Corresponding author: Ji Qiu (15117393@bjtu.edu.cn)

This work was supported by the Natural Science Foundation of Beijing Municipality under Grant L171009.

ABSTRACT This paper addresses the event modeling and recognition problem in video surveillance systems using the system net on the Petri Net (PN) formalism. The single event on the foundation of prior knowledge is first modeled via arranging event schemes from the design document. Then, finite sequential runs (FSRs) determined by semantic features in training clips drive elements of the proposed single event model. Finally, the multi-event model is built automatically from single event models via a proposed integration method using multi-level features (including high-level semantic features and low-level features like numerical characters of individual trajectories). We provide a novel solution to event conflict issue through an extended system net where the resultant event type is determined by the decision tree technique. The comparison between the proposed methodology and other approaches in the literature is reported via experiments on an acknowledged public dataset.

INDEX TERMS Event recognition, high-level video event modeling, video surveillance system, Petri nets, video analysis application.

I. INTRODUCTION

Due to the prosperity of video surveillance systems, long-term surveillance of multiple channels simultaneously challenges the capacity of human operators and gives rise to demand on automated reasoning. Hence, various event analysis approaches are developed for video surveillance systems. The typical framework of event analysis application is comprised of subtasks that are generally grouped under three categories: low-level vision, intermediate-level vision, and high-level vision. Low-level vision tasks comprise operations such as image acquisition and pre-processing. Intermediate-level tasks pertain to segmentation, symbolic representation, classification, recognition, and tracking. High-level vision tasks are concerned with achieving a conceptual understanding of information and realizing further semantic tasks.

This paper devises an event modeling framework that realizes multi-event recognition via an extended elementary system of Petri net. It concentrates on the high-level event modeling in a particular application and for recognition of these events in video sequences. As the knowledge-based framework depicted in Fig. 1, proposed methods of event modeling and recognition are applied after the module of knowledge representation. That is, we assume that states of an object in each frame with multi-level features from low-level numerical values of characters in individual trajectories (like object size, location, speed) to high-level semantic features (like semantic actions, semantic region) are given. Similar to [1]–[5], we assume that an event label is assigned to each object in every frame. Contribution of this work can be highlighted by the followings:

1) We provide a concise event recognition framework for proposed event models. Analysis of a given video is turned into moving of tokens inside the system net for the proposed framework. State of the event reasoning machine, that is,

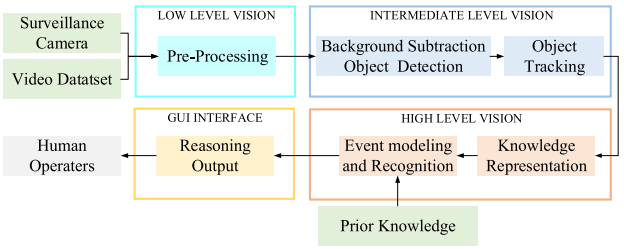

FIGURE 1. Architecture of high-level event analysis applications.

The associate editor coordinating the review of this manuscript and approving it for publication was Guitao Cao.

an event model whose intrinsic structure is an extended system model, triggers the operations of the event recorder.

2) We devise an intuitive finite sequential run (FSR) centered single event modeling strategy which hasn't been proposed to the extent of our knowledge in existing models. Mapping rules of entities from both prior knowledge and training dataset into elements of the Petri net model are depicted. Algorithms are given to build the single event model from training clips automatically and to associate each FSR with corresponding clips.

3) Algorithms that directly built the multi-event model from the training clips are presented. We propose a novel solution to the event conflict that different events share the same FSR. It is addressed by extending the system net with specific place types with rules together with the decision tree technique.

4) Experimental results verify that our proposed framework achieves significant performance advantages against others.

We structure the remainder of this paper as follows: Section II covers the preliminaries and related work, Section III presents details of the proposed framework and algorithms, Section IV provides the experimental results, evaluations, and discussion, and Section V makes a short summarization.

II. VIDEO EVENT MODELING USING PETRI NET

A. DEFINITIONS IN PETRI NET

Petri Nets is a formal and graphical appealing language which is appropriate for modeling systems. Basic components of a Petri net structure include three kinds of elements called places and transitions as well as directed arcs that connect between them. Besides, the token is designed to reflect the dynamic states of a Petri net. In this section, elementary conceptions of the standard Petri net are introduced. For more information on Petri nets, please refer to [6].

Definition 1 (Net Structure): Let P and T be sets and let $F \subseteq (P \times T) \cup (T \times P)$, $N =_{def} (P, T, F)$ is a net structure. P, T and F contain the *places*, *transitions* and *arcs* of N, respectively.

A place p always models a passive component that has discrete states while a transition t always models an active component that is capable of producing, consuming, transporting or changing. The arc describes abstract, sometimes only notional relation, such as logical connections, access rights, spatial proximities or immediate linking. A transition t can be labeled with a condition and the arcs around t with expressions. These labels show the various situations (modes) in which t is enabled, and the respective effects at the occurrence of t. Lastly, every transition is either hot or cold, where cold transitions are indicated by ε.

Definition 2 (Marking and Final Marking): Assume a universe U that contains all examined kinds of *tokens* and $\mathcal{M}$ as the set of all multisets over U, a mapping $M : P \to \mathcal{M}$ is a marking of N. A system has reached a final state if it can remain in this state forever. A final state of a system corresponds to a final marking. In such a marking, no hot transitions are enabled.

A *marking* M of N is a distribution of tokens (multisets) across the places of N, which describes a state of the modeled system. Typically, the initial marking of N is denoted by M_0 and is explicitly drawn into N.

Definition 3 (System Net): Let $N = (P, T, F)$ be a net structure, let U be a set, let M_0 be a marking of P over U, let $\tau : T \to cond$, let $\ell : F \to \mathcal{M}$, let $C \subseteq T$. Then $S =_{def} (N, M_0, \tau, \ell, C)$ is a system net over U.

System nets are used to model real, discretely changeable systems. A net structure together with an initial marking, transition conditions, arc labeling, and cold transitions form a system net. Each place of a system net models a state component of the system and each currently existing token in a place models a currently given, but changeable, characteristic of that component. Each transition of a system net represents an action of the system. The occurrence of a transition describes the occurrence of the respective action. If in doing so, a token reaches or leaves a place, the action respectively creates or terminates the corresponding characteristic of the state component.

A marking M of a 1-bounded system net can be described as a string of marked places. As the event reasoning model is designed for the recognition of a single object event, the system net in our reasoning machine is a 1-bounded elementary system net.

Definition 4 (Finite Sequential Run): A sequential run of a system net N

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} M_n$$

is a finite sequential run of N *iff* M_n is a final marking of N.

B. RELATED WORK ON EVENT ANALYSIS

Event modeling involves defining inference models for specific event types and developing methods for recognition that corresponding to these models. To capture the complexity existing in the domain of video sequences, the inference model is required to take into account factors including spatial, temporal and logical relations. To this end, many approaches have been applied in the event modeling and recognition task. Models including Convolution Neural Networks (CNN), Bayesian Networks (BN), Hidden Markov Models (HMM), Petri Nets (PN), and Context-Free Grammars have been successfully adapted to provide solutions for event recognition as it will be explained in next paragraphs of this section.

Pattern recognition approaches take advantage of deep networks to derive powerful and distinguishing feature portrayals. Rather than using high-level semantic features, these models usually adopt video frames or spatial-temporal trajectories as input. Zhou *et al.* [7] proposes a deep neural network including a motion fusion block, a feature transfer block, and a coding block. They are designed to keep the temporal

and spatial connection between the motion and appearance cues and extract discriminative features by exploiting the transferability of the neural network from different domains. Abnormal frames are detected by the reconstruction error of the dictionary encoding normal events. Ullah *et al.* [8] investigates the convolutional layers of a pre-trained CNN to detect human saliency in the surveillance stream. Learned salient regions are further used for salient feature extraction and shot segmentation, resulting in representative shots suitable for industrial video stream analysis and activity recognition.

In [9], an event classification system is proposed which learns the relative significance of the features from the training set of data. A triple-channel model is designed in [10] which yields the final detection results via fusion method on triple channels including spatial CNN features, temporal CNN features and their fusion at convolution layers.

Bayesian networks can encode spatial and logical relations. BN models in [11], [12] are usually tailored to the application domain to recognize simple events. Hidden Markov models (HMMs) are the extension of BNs over time which allow computational tractability. Hence they are effective for recognizing sequential events with different temporal durations and more complexities. Epaillard and Bouguila [13] derives the variational learning for the Dirichlet HMM and extended it to the generalized Dirichlet case. Besides, a realistic solution is proposed to the problem of unusual event detection in crowds of pedestrians that yields convicting results compared with other HMM models like [14].

Another popular event model is Context-Free Grammars which naturally encode sequence and hierarchy using grammar models. In [15], a knowledge representation framework is proposed for video event recognition. It is capable of describing patterns of knowledge via specific representations including Elements of Context Representation (ECR), Action representation (AR), and General Descriptors of Context (GCD) with corresponding reasoning rules. However, existing approaches [4], [16], [17], both knowledge-based and rule-based schemes through semantic representations, require heavy manual modeling work.

Compared with other event analysis models, the Petri net is chosen as the inference mechanism for the following advantages: 1) Both deterministic and stochastic inference of event occurrences can be addressed by Petri nets. Much fewer observations are required compared to deep network methods. 2) Events can be represented at various levels of abstraction using Petri nets. 3) Petri nets have intuitive graphical representations with well-defined semantics. The place of the token indicates the current state of the tracking target. Past places and transitions describe what happened and connecting branches of the place predict what will happen in the further.

The Petri net is a graphical and mathematical modeling tool to describe and study information processing systems. They are particularly useful since they permit to express concurrency and to use smart control strategies. Formally, a Petri net is a directed bipartite graph composed of places, transitions, and arcs. To enable the target of event detection, automatic mappings of ontology entities into Petri nets must be defined. Researches define the event representation according to their understandings of the Petri net. Petri nets and its variations are widely used in modeling video events for their detection and recognition since plan-based PN presented by Castel *et al.* [18]. Ghanem *et al.* [19] adopts PN models for both representation and recognition. A graphical user interface is provided where Petri nets can be generated from queries. These Petri nets are provided with primitive events detected from video streams and are used as filters to recognize composite events. Albanese *et al.* [20] proposes a PN approach for the recognition of human activities. This work describes a probabilistic Petri Net representation of complex activities based on atomic actions that are recognizable by image understanding algorithms. Lavee *et al.* [21] provided a methodology and examples of how formal ontology language definitions for an event can be transformed into a Petri net formalism. After that, a Particle Filter Petri Net is proposed in [22] to model and recognize activities in videos. That method allows a recognition of the activity which takes into account the uncertainty of events obtained by low-level processing of surveillance videos. Hamidun *et al.* [23] translates the event sequence in the crossing scenario to the PN model. The combined effects of spatial and temporal information are analyzed using the steady-state analysis built in the model which indicates that modeling with Petri Nets also allows the development of a model in the hierarchical structure.

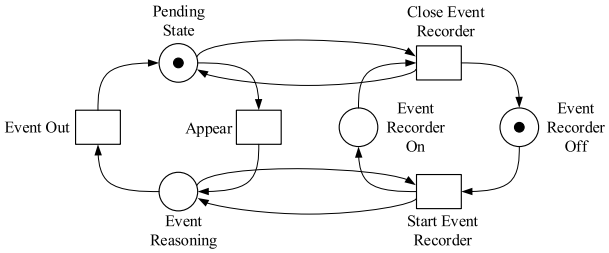

FIGURE 2. System net of high-level event detection framework.

III. PROPOSED METHODS FOR VIDEO EVENT MODELING AND RECOGNITION

To realize event modeling and recognition from subsequent video sequences, the proposed framework is made up of two parts, namely, an event reasoning machine and an event recorder. Their interactions are shown in Fig. 2 where the left part depicts states of the event reasoning machine and the right part present stats of the event recorder. A given video will be cut into clips and frames in each clip share the same event label determined by the reasoning machine.

In the initial state, the reasoning machine is on a pending state while the event recorder is closed. The inial state stays until the object appears on a certain frame. The transition 'Appear' will be enabled upon the target appears because its incoming arc starts at a place containing a token and the transition condition is satisfied. The effect can be deduced

from the arrows starting or ending in it: the token will move from 'Pending State' to 'Event Reasoning'. Intuitively, the appearance of the target leads to the start of an event recognition process in the reasoning machine. The intrinsic pattern of the reasoning machine is an extended system net which will be described in Section III-C. Immediately, the transition 'Start Event Recorder' will be also fired as both of its incoming arcs are satisfied. That results in the movement of the token from 'Event Recorder Off' to 'Event Recorder On'. That is, the event recorder will start to work and the clip to be recognized starts at this appearance frame.

Then, the current marking keeps still until the transition 'Event Out' is triggered. Determination conditions of the inference machine for single event and multi-events will be explained in details in Section and III-C, respectively. After the transition 'Event Out' fires, the last frame will be recorded as the end frame of the clip to be recognized. The event reasoning machine will assign an event label for this clip which is further recorded by the event recorder and the left token will return to 'Pending State'. Besides, the transition 'Close Event Recorder' will occur because the arrows ending in it are satisfied. Hence the right token will be back to 'Event Recorder Off' leading to an identical marking of the initial one as depicted in Fig. 2. After that, things continue to happen for the same process inside this framework and the event recorder writes down the labels of frames.

A. SINGLE EVENT MODELING: PRIOR KNOWLEDGE

Both prior knowledge and training datasets can be the source of the single event model. In some real-world applications, especially abnormal event detection tasks, concerned events rarely happen while having important practical significance. Prior knowledge from experts will compensate for the possible absence of vital events in training datasets.

The prior knowledge provided by field experts usually includes qualitative representations for elements and schemes of events. Technically, it can be described as a directed graph consisting of states and arrows. States of the tracked object are described by high-level semantic features. The state change of the target from one to another following the connecting arrow depicts the execution of a step. The sequence of actions form a routine and various routines may represent for the same event. Note that generally only high-level semantic features are exploited during the single event modeling stage.

A modeling process of event 'Windowshop' using prior knowledge is depicted in Fig. 3 which occurs in a shopping center. In such a context, a person walks past a shop window at various places and then enters the shop or walks away. The prior knowledge is introduced in the form of an event scheme where the graph of the situation sequence is presented. Each flow, starting from an input arrow to an end arrow, represents an individual routine of this event.

Following the arrow flows, each event of interest can be arranged into routines made up of semantic feature sequences manually. Each transformation from one situation to another corresponds to an individual step. That is, a step occurs

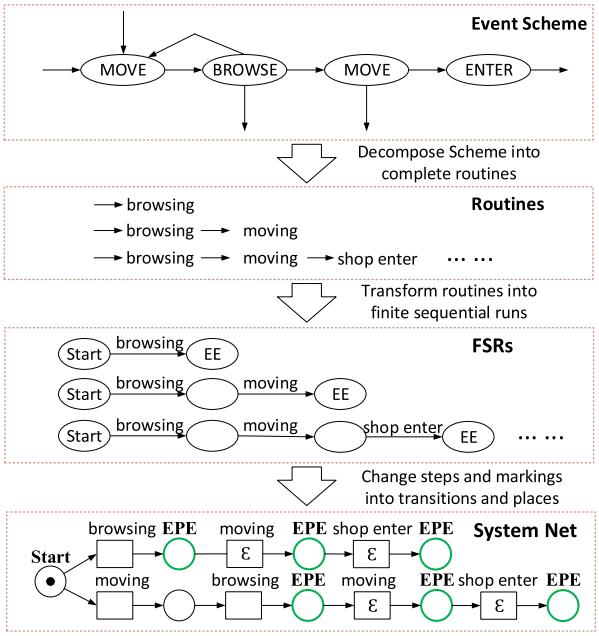

FIGURE 3. An example of the single event modeling approach. The prior knowledge is introduced in the form of an event scheme graph, which is finally transformed into a system net via arranging FSRs.

when the state comes to a fresh situation. The loops in schemes, like 'MOVE-BROWSE' in this example, might occur repetitious in an event leading to infinite possible routines. In some cases, the infinite routines can be represented by the combination of other existing routines. For example, the routine 'MOVE-BROWSE-MOVE-BROWSE' of the scheme falls to the former case since it is the connection of two 'MOVE-BROWSE's which is already an independent routine. Then, the combinatorial sequence will be discarded since the designed framework is capable of recognizing the event through sub-routines. For the other case where the infinite routines cannot be represented through combinations, our solution is to cover the common pattern with high probabilities.

After that, each independent and complete routine will be transformed into an FSR. As a 1-bounded elementary system net which designed for single target, the marking of our event models is exactly the place where the only token stays. In practice, that is exactly the state of the target. The step, which is also the transition condition in system net, is hence matching the new situation. When the situation meets the requirement in a frame, the state change happens. The initial marking is donated as 'Start' for the unknown start upon the appearance of the target. The final marking which might be lots of is event end is short as 'EE' in the figure. Note that markings in the 'FSRs' block are not shown in the formal format of the Petri net as they are not shown in the names of places. Since places could be repeated in routines, their labels are omitted in the reasoning stage to magnify the steps and the network structure.

Finally, the single event model will be generated using the backstepping algorithm since the inference machine is

a 1-bounded system net. Places, transitions, and arcs are produced from the markings and steps of FSRs. When a place corresponds to the end marking in an FSR, the place is determined as an end place drawn in bold circles. As aforementioned, ε inside rectangles donates for cold transitions. The cold transitions ensure automatical checking of finite sequential runs whose end place still have subsequent branches. For details of the system net model for the proposed single event model, please refer to Section III-B. Besides, a token will be put into the place 'Start' when the target appears.

B. SINGLE EVENT MODELING: TRAINING DATASET

While prior knowledge can generate routines, they are lack of real-world cases with enriched information. A video clip includes details from the low-level features like trajectory attributes to high-level semantic features per frame for each target. Additional information enhances the discrimination of the reasoning machine as more conditions can be supplied. When a training dataset is exploited together with the event scheme, it automatically provides these manual routines with clip examples and verifies the prior knowledge in turn.

Rather than modeling directly from videos, the training dataset is further segmented into training clips according to the event label of a target. In single event modeling, the training clip of an object is consecutive frames which owns the event label to be learned. That is, each training clip describes a single object continuously conducting an event e_k. These training clips with multi-level features generated from the pipeline in Fig. 1 will further be exploited for two effects: one is to extracted implied FSR to gather possible routines of the event, the other is to provide the FSR with real-world clips. The latter connection supplies the event model with enhanced discriminative ability. To be consistent with the prior knowledge solution, only semantic features are exploited in a single event model.

To support both situations that prior knowledge is given or not, the handling of training clips is separated into two stages. At first, they are extracted into FSRs and then a unified system net is built upon. Hence the representations are identical to the prior knowledge solution where the step of FSR denotes the matching of semantic features in the following state.

The method for the first stage is proposed as Algorithm 1 which extracts FSRs from the training dataset. It assumes the existence of prior knowledge. When FSRs determined by prior knowledge are absent, it still works with leaving the input PK blank. As it searches over all the frames in training clips, the complexity is linearly correlated to the number of observations in training clips. Let the number of training clips for event e_k be N_k and the number of frames in clip TCi be Fr_{ki}, the computational complexity of extraction stage in Algorithm 1 would be $O(\sum_{i=1}^{N_k} Frki)$. Denote the number of FSRs in FSR set for TCi as R_{ki}, the complexity of FSR repetition checking stage varies from 1 to R_{ki}. Hence the computation complexity in this stage would be a value between $O(N_k)$ and $O(\sum_{i=1}^{N_k} R_{ki})$, for the best and the worst respectively.

Algorithm 1 FSR Extraction

Input: PK: FSRs of e_k determined by the prior knowledge
TC: Training clips of objects conducting the event e_k
Output: FSR: complete FSRs of e_k

$FSR = PK$ //Initialize FSR using prior knowledge
for every clip TC_i of TC **do**
  // Extract steps of routine in current clip
  Initialize empty SR, $stepID = 1$
  for each frame in TC_i **do**
    if initial frame **then**
      $SR(stepID) = TC_i(1).sf()$
    else if $FireCheck(TC_i(CRT))$ **then**
      $stepID++$
      $SR(stepID) = TC_i(CRT).sf()$
    end if
  end for
  // Check the repetition of current routine
  $APflag = []$ // initialize the repetition flag
  for each FSR FSR_m in FSR **do**
    if $IdenticalSR(FSR.Steps(m), SR)$ **then**
      $APflag = [APflag, m]$
    end if
  end for
  // Check repetition flag and update
  if $isempty(APflag)$ **then**
    // Create a new FSR FSR_{new}
    $FSR.Steps(new) = SR$
    $FSR.Clips(new) = [i]$ //add clip i as an example
  else
    $FSR.Clips(flag) = [FSR.Clips(flag), i]$ // Update examples of the repeated routine
  end if
end for

For each training clip containing a concerned event, it is divided into consecutive intervals with identical semantic features inside and different semantic features from neighbors. A new step of the current routine will be generated when a new interval starts. In the proposed algorithm, CRT is short for the current frame and $sf()$ represents semantic features. The output of function $FireCheck(TCi(CRT)))$ is a binary flag. It is true if and only if the semantic feature of $TC_i(k)$ is different from $TC_i(k-1).sf()$.

After that, each training clip is arranged into corresponding routine composed of steps. Further, they are gathered to merge identical routines. For each clip in order, the steps of current clip is compared with existing FSR set. Given two FSRs FSR_a and FSR_b, the binary flag of function $IdenticalFSR(FSR_a, FSR_b)$ is true if and only if the input FSR_a and FSR_b :

1) $length(FSR_a.Steps) = length(FSR_b.Steps) = N$
2) $\forall n = 1, \ldots, N$, $FSR_a.Step(n) = FSR_b.Step(n)$.

Algorithm 2 Single Event Modeling

Input: FSR: FSRs of the kth event
Output: System net N_k for e_k

Initialize empty $N = (P, T, F), \tau, \ell, C$
$P_0.Type = \text{'}Start\text{'}, M_0 = P_0$
// Construct Petri net structure
for each $FSR(r)$ in FSR **do**
  while step $n \leq length(FSR(r))$ **do**
    Set a token in P_0
    if $FSR(r).Steps(n) \subseteq P^{\cdot}_{CRT}$ **then**
      Fire the fitting transition
    else
      $T(new).condition = FSR(r).Steps(n)$
      Add a new place $P(new)$ and corresponding arcs
      Move the token to $P(new)$
    end if
    if $n == length(FSR(r))$ **then**
      $P_{CRT}.Type = EPE$
      $P_{CRT}.Clips = FSR.Clips(CRT)$
      Clear the token
    end if
  end while
end for
// Generate cold transitions τ
for each place fitting '$P.Type = EPE$' **do**
  if exist($P^{\cdot}_{CRT}$) **then**
    $P^{\cdot}_{CRT}.Label = \varepsilon$
  end if
end for

When an identical routine is found in existing FSRs, the clip is added as a new example of this FSR. If not, a new FSR is built in the FSR set and this clip is also attached. After this procedure, the output *FSR* is a structure including complete FSRs and attached clips for each FSR.

In the second stage, the system net for a single event is modeled based on the *FSR* using algorithm 2. It construct the system net for a single event for all its FSRs. Let the number of FSRs for event e_k be R_k and the number of steps in $FSR(r)$ be L_{kr}, the computational complexity of Algorithm 1 would be $O(\sum_{r=1}^{R_k} Lkr)$.

Note that the event model together with specific termination mechanisms is the reasoning machine in proposed framework. Two special types of place are determined in the proposed single event model: *initial place* P_0 and *end place of event* (EPE), which are labeled with 'Start' and 'EE', respectively. Upon appearance, the tracked object will be modeled as a token in the initial place of this system net. It is an instant state as immediately the connected transitions $P^{\cdot}_0$ will be checked leading the movement of the token into other places. A place will be determined as an EPE if and only if it represents the final state of an FSR. Refer to *Definition* 4, transitions in $EPE^{\cdot}$ are cold transitions.

The reasoning termination of a single event model is caused by three factors: the disappearance of the tracked target(including the termination of testing video), the end of the event and any unexpected step. For a token staying in p, the unexpected step occurs when a fresh step that is different from any transition in ${}^{\cdot}p \cup p^{\cdot}$ happens. Note that ${}^{\cdot}p$ is the transition leading to the place p hence implying the occurrence of an unchanged state while $p^{\cdot}$ is transitions that directly connect p with following places which result in the change of state into a certain predefined state of the single event model. If any termination condition happens when the token stays in an EPE, the current routine fits an independent FSR of the event hence frames of this reasoning interval (clip) will be assigned as this event. If not, the routine mismatches any existing FSRs and will be determined as not the event. After termination of the inference process, the current token will be reset and the 'Event Out' transition in Fig. 2 will be triggered.

C. MULTI-LEVEL FEATURE BASED MULTI-EVENT MODELING

Several system nets of single events cannot integrate into a multi-event model directly because of event conflicts. Event conflict happens when clips share identical FSR while having different event labels. In the view of the Petri net, a place would be EPE for multiple events. When a testing clip ended here, the reasoning machine cannot decide which event ought to be the ground-truth from multiple event labels of this EPE. Confronted with event conflicts, corresponding clips of each event provide additional features to train a classifier to determine the event label.

The proposed method of multi-event model construction from single event models is presented in Algorithm 3. To the best of our knowledge, this approach has not been exploited before for the application of the classification technique in high-level event modeling. Besides initial place and EPE, *Conflict Place of Event* (CPE) is added in the multi-event model to address the event confliction. After the system net with uncertainty is preliminarily composed, the EPEs with more than one event labels are CPEs. The low-level features of clips are gathered as predictors while its event labels are recorded as the response. With the predictor and response, the classifier attached to a CPE is trained. The construction of these classifiers is explained in next paragraphs of this section. Therefore, the information in low-level features that hasn't been extracted by knowledge representation is organized as a supplement. They form an additional semantic feature via classification when needed. They are not used when the sematic features are capable of event modeling. After the formation of the classifier attached to the CPE, additional transitions and places are introduced with the same number of event labels contained in it. The type of newly added place is EPE and each place owns an event label previous contained in the CPE. The conditions of newly added transitions are the classification results correspondingly.

Algorithm 3 Multi-Event Modeling

Input: N: System nets of events N

FSR: finite sequential runs of events

Output: ISN: Integrated system net for multi-event recognition

// Construct a system net with uncertainty

for each event e_k of interest **do**
  if $k == 1$ **then**
    $ISN = N$
    Continue
  end if
  for each $FSR(kr)$ in $FSR(k)$ **do**
    Add a token in P_0
    for each step $FSR(kr).Steps(n)$ **do**
      if Last Step **then**
        $P_{CRT}.Type = EPE$
        $P_{CRT}.eventID = [P_{CRT}.eventID, k]$
        $P_{CRT}.Clips.eventID = P_{CRT}.Clips$
        Clear the token
        Continue
      end if
      if $FSR(kr).Steps(n) \subseteq P_{CRT}^{\cdot}$ **then**
        Fire fitting transition
      else
        Add $T(new) = FSR(kr).Steps(n)$
        Add $P(new)$ and corresponding arcs
        Move the token to $P(new)$
      end if
    end for
  end for
end for

// Modify the uncertainty in net structure

for each place fitting $length(P.eventID) > 1$ **do**
  $L = length(P_{CRT}.eventID)$
  Initialize an empty training dataset TC
  Add $T(New1) \cdots T(New(length(P_{CRT}.eventID)))$
  Add $P(New1) \cdots P(New(length(P_{CRT}.eventID)))$
  for each column in $P_{CRT}.eventID$ **do**
    $TC(new).Label = P_{CRT}.eventID(CRT)$
    $TC(new).Clips = P_{CRT}.Clips.eventID$
    $T(New(CRT)) : Class = P_{CRT}.eventid(CRT)$
    Add arc: $P(CRT)$ to $T(New(CRT))$
    Add arc: $T(New(CRT))$ to $P(New(CRT))$
  end for
  Train the classier attached to P_{CRT} using TC
end for// Generate cold transitions τ
for each place fitting '$P.Type = EPE'$ **do**
  if exist($P_{CRT}^{\cdot}$) **then**
    $P_{CRT}^{\cdot}.Label = \varepsilon$
  end if
end for

To explain the solution intuitively and simply, people moving inside a shopping mall might be explained as either

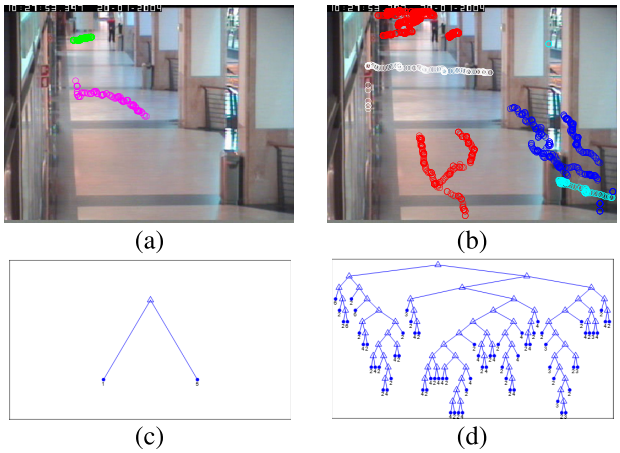

FIGURE 4. Example clips of CPEs and their trained decision trees correspondingly. Circles in the same color donate for trajectories of a single event type. 100 frames since the last step of the clip are selected as predictors of the decision tree while corresponding event IDs are recorded as labels of classification results.

'walking along the hallway' or 'entering a shop'. For the two events, their FSRs are the same with only step 'moving'. Hence the system net cannot distinguish the truth event given a new clip with such routine. Now, if we gather the locations in training clips of the two events and train a classifier. The trained classifier can determine the boundary of events automatically. Given region subset A as the range of locations belongs to 'walking' while region subset B as the range of locations belongs to 'shop enter', we may determine the event label of a new clip with the help of this classifier. Intuitively, subset A semantically belongs to the hallway while B belongs to areas around the shop doors. If the locations of this new clip belong to subset A, then it should be 'walking'. Two detailed examples are depicted in the first row of Fig. 4. Each subfigure plots the observations of a CPE. They are drawn in small circles with different colors indicating corresponding event labels.

In the proposed reasoning machine, the classification of an event in the CPE is automatically realized through the decision tree technique. To cooperate with real-world situations, only features in the early frames of current state, that is, frames after the last transition are gathered as predictors of the observation. Each observation shares the same weight. The cost during the training stage is a square matrix, where the rows and columns are responses. $COST(I, J)$ is the cost of classifying a observation into class J if its true class is I. $COST(I, J) = 0$ if $I = J$, and $COST(I, J) = 1$ if $I \neq J$. That is, any misclassification is equally punished. The split predictor in each node is chosen by maximizing the gain in the split criterion over all possible splits on all predictors. And Gini's diversity index is selected as the split criterion. In the following experiment, there is no limitation for the maximum split number. To enhance the robustness, two numerical values are predefined for the minimum observation size of the parent node and leaf node. Hence, the splitting process is stopped when any of the following condition is hold:

TABLE 1. Transition list for single event models in Fig. 5.

ID	Transition condition	ID	Transition condition	ID	Transition condition	ID	Transition condition
a1	Token.situation='moving'	a12	Token.situation='browsing'	c9	Token.situation='inactive'	e8	Token.situation='browsing'
a2	Token.situation='browsing'	a13	Token.situation='moving'	d1	Token.situation='moving'	e9	Token.situation='moving'
a3	Token.situation='moving'	b1	Token.situation='moving'	d2	Token.situation='shop enter'	f1	Token.situation='shop exit'
a4	Token.situation='shop enter'	c1	Token.situation='inactive'	d3	Token.situation='shop enter'	f2	Token.situation='moving'
a5	Token.situation='browsing'	c2	Token.situation='moving'	e1	Token.situation='browsing'	f3	Token.situation='moving'
a6	Token.situation='moving'	c3	Token.situation='inactive'	e2	Token.situation='moving'	g1	Token.situation='shop exit'
a7	Token.situation='browsing'	c4	Token.situation='moving'	e3	Token.situation='browsing'	g2	Token.situation='moving'
a8	Token.situation='moving'	c5	Token.situation='inactive'	e4	Token.situation='moving'	g3	Token.situation='shop enter'
a9	Token.situation='browsing'	c6	Token.situation='moving'	e5	Token.situation='moving'		
a10	Token.situation='moving'	c7	Token.situation='inactive'	e6	Token.situation='browsing'		
a11	Token.situation='shop enter'	c8	Token.situation='moving'	e7	Token.situation='moving'		

1) The node is pure. That is, the mean squared error (MSE) for the observed response in this node drops below the MSE for the observed response in the entire data multiplied by the tolerance on quadratic error per node.

2) The node has fewer observations than predetermined parent node size or any split imposed on this node procures children with fewer observations than predetermined leaf node size.

Algorithm 3 is made up of two stages: the construction of system net with uncertainty and the modification of CP using the decision tree technique. The computational complexity of the first stage depends on the number of FSRs for all the event. Let the number of concerned events be K, the number of FSRs for event e_k be Rk, and the number of steps in FSR_{kr}, maximin of its computational complexity would be $O(\sum_{k=2}^{K}\sum_{r=1}^{R_k} FSR_{kr})$. The complexity of the second stage is determined by the number of conflicted places and the decision tree constructions. When a decision tree grows, all dimensions of features are candidate for split condition. Let the number of CPs be M, the sample number for CP_m be S_m, the dimension of features be Q, the depth of decision tree attached to CP_m be D_m, and the number of parent nodes be P, the computational complexity of the second stage would be $O(\sum_{m=1}^{M} S_m * Q * D_m)$.

The reasoning termination of a single event model is caused by three factors: the disappearance of the tracked target(including the termination of testing video), the end of the event and any unexpected step. For a token staying in p, the unexpected step occurs when a fresh step that different from any transition in $\dot{p} \cup p^{\cdot}$ happens. If any termination condition happens when the token stays in an EPE, the current routine fits an independent FSR of the event hence frames of this reasoning interval (clip) will be assigned as this event. If not, the routine mismatches any existing FSRs and will be determined as not the event. After termination of the inference process, the current token will be reset and the 'Event Out' transition in Fig. 2 will be triggered.

The system net for multi-event recognition shares the same termination factors with that of the single event model. If the termination happens on EPE or naive places, the decision mechanism remains identical. When the termination happens on CPE, additional operations are introduced. The features of the first frame when the token reaches the CPE are given to the attached decision tree. Its classification result will trigger an existing transition resulting in the movement of the token from this CPE to a neighbor EPE, whose event label will be determined as the event during this interval.

IV. EXPERIMENTAL RESULTS

In this section, experimental results are presented to verify the performance of the proposed event recognition method. Results of methods in the published literature [1] and [5] are chosen as comparisons.

The CAVIAR (Context Aware Vision using Image-based Active Recognition) dataset [24] 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, high-level semantic descriptors including movement, role, situation, and context are assigned to each object for every frame. Besides, low-level features are also written, including location and size information. Each clip is recorder from two different points of view. The first one shows a view of the corridor, while the second shows a frontal view of the scenario. Choosing the corridor as the scene, the training dataset includes 24 videos whose length ranging from 11 seconds to a few minutes and is further separated into 246 clips according to rules in Section III-B.

Context can be described as the overriding semantic purpose of the object in question. Therefore, the context is chosen as the ground-truth of event label as other literature [1], [5] did. The situations and locations are semantic features and low-level features respectively, which composed into the multi-level features under the proposed framework.

A. RESULTS ON SINGLE EVENT MODELING

The system nets for single event recognition built from the training dataset are depicted in Fig. 5. Each context in the dataset is treated as a type of event. Start places are those with a token while EPEs are drawn in bold circles with various colors. The transition conditions are written beside the rectangle transition boxes. As aforementioned, the situation is the only feature exploited in single event modeling. Hence these abbreviations omit 'Token. situation = ' to fit

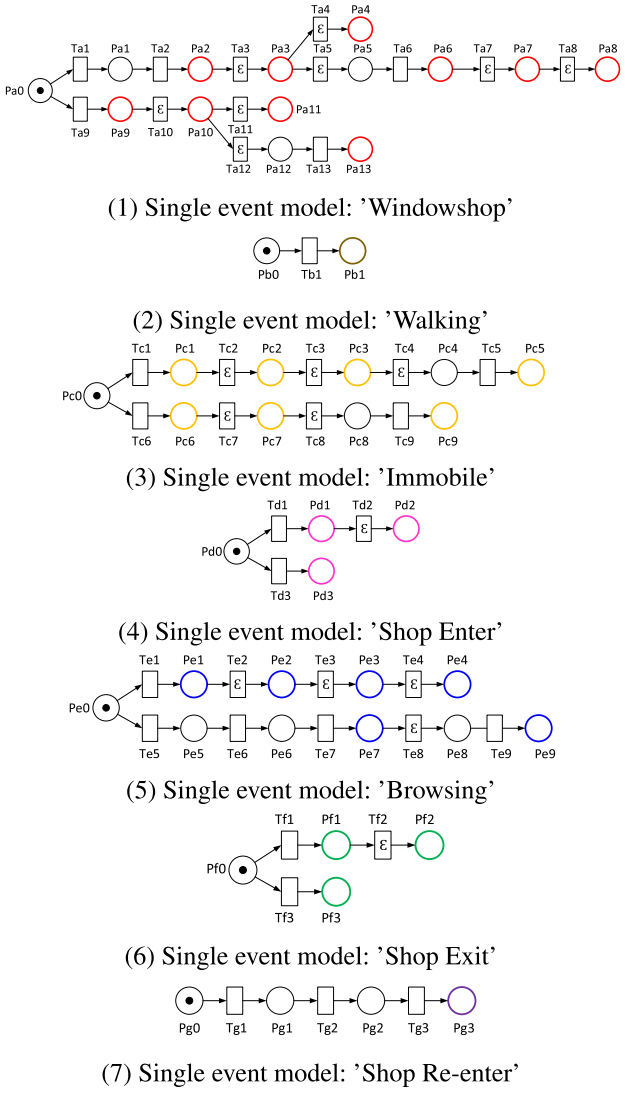

FIGURE 5. Single event models produced by algorithm1 and algorithm2. The transition conditions are given in Table 1.

the figure space. For example, the label 'browsing' is 'Token. situation = browsing' in short, which represents the transition will be fired if the token located in its previous place comes to a frame with transition 'browsing'.

We may find that duplicate finite sequential runs exist in different event models. Hence single event model merely built on situation information would mistake other events whose FSR is identical to its branch resulting in lots of false positives. The performance of single event recognition is obtained via checking clips using every single model.

As shown in Table .2, false positives exist in six of the seven types. The perfect precision and recall of event 'shop reenter' is realized because the only FSR it occupies is unique and does not arise any collisions with others. A severely sharing routine is that of the 'Walking' event, which contains an object with only a 'moving' situation in the sequence. The sharing events 'Walking', 'Shop Enter', 'Browsing' and 'Shop Exit' are all effected in precision. At the same time, all

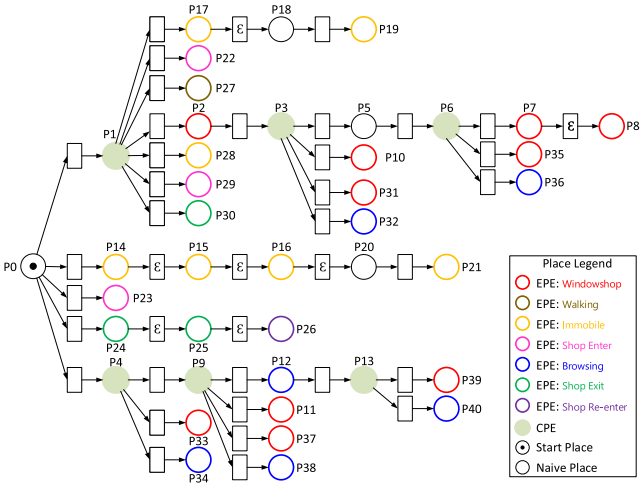

FIGURE 6. The multi-event model produced by algorithms 1 to 3. Transition conditions are given in Table 3.

the positive events are recognized correctly since their recalls are exactly 100%. Hence it proves that the proposed single event model can recognize events based on the semantic feature.

B. RESULTS ON MULTI-EVENT MODELING

In the following experiments, we simply apply the 2-dimension location as the predictor of the classifier as it is already sufficient for the experimental dataset. Further, only a few frames since the last step are exploited in the training stage. The decision tree determines the event label according to the location of the first frame in the last step.

After single event modeling, the multi-event model can be realized via Algorithm .3. The system net of the training dataset is shown in Fig. 6. All the concerned events can be recognized with this model. Those filled circles represent the CPEs and bold circles represent EPEs. The transition conditions are depicted in Table 3. The transition number is the same as its posterior place $t^{\bullet}$. And the 'Token. LocClass' stands for the classification result of the decision tree attached to its previous place. We use the location of the tracking object on the first frame of the last step as the input of the decision tree. And the decision tree is attached to the previous CPE of the transition.

The performance of the model is shown in Table 4. There is only one mistake in all the 246 predictions. An event whose ground-truth event is 'Walking' is recognized into 'Shop Exit'. That lead to the flaw in the recall of 'Walking' and precision of 'Shop Exit'. The false prediction is made by the decision tree attached to place $P1$ since the initial location of the clip is mixed with other shop exit locations. Compared with Table. 2, huge improvement happens to the false positives. That verifies the effectiveness of the proposed framework. Those event conflicts between 'Windowshop' and 'Browsing' are completely handled via several decision trees. Two types of the four conflicted events in the CPE

TABLE 2. Performance of single event models in Fig. 5.

EventID	Context	True Positive	False Positive	False Negative	True Negative	Precision(%)	Recall(%)
1	Windowshop	17	7	0	222	70.83	100
2	Walking	86	6	0	154	93.47	100
3	Immobile	22	90	0	134	19.64	100
4	Shop Enter	49	89	0	108	35.51	100
5	Browsing	8	10	0	228	44.44	100
6	Shop Exit	59	91	0	96	39.33	100
7	Shop Re-enter	5	0	0	241	100	100

TABLE 3. Transition list for the multi-event model in Fig. 6.

ID	Transition Condition	ID	Transition Condition	ID	Transition Condition	ID	Transition Condition
1	Token.situation='moving'	11	Token.situation='shop enter'	21	Token.situation='inactive'	31	Token.LocClass='1'
2	Token.situation='browsing'	12	Token.situation='browsing'	22	Token.situation='shop enter'	32	Token.LocClass='5'
3	Token.situation='moving'	13	Token.situation='moving'	23	Token.situation='shop enter'	33	Token.LocClass='1'
4	Token.situation='browsing'	14	Token.situation='inactive'	24	Token.situation='shop exit'	34	Token.LocClass='5'
5	Token.situation='browsing'	15	Token.situation='moving'	25	Token.situation='moving'	35	Token.LocClass='1'
6	Token.situation='moving'	16	Token.situation='inactive'	26	Token.situation='shop enter'	36	Token.LocClass='5'
7	Token.situation='browsing'	17	Token.situation='inactive'	27	Token.LocClass='2'	37	Token.LocClass='1'
8	Token.situation='moving'	18	Token.situation='moving'	28	Token.LocClass='3'	38	Token.LocClass='5'
9	Token.situation='moving'	19	Token.situation='inactive'	29	Token.LocClass='4'	39	Token.LocClass='1'
10	Token.situation='shop enter'	20	Token.situation='moving'	30	Token.LocClass='6'	40	Token.LocClass='5'

TABLE 4. Performance of the multi-event model in Fig. 6.

EventID	Context	True Positive	False Positive	False Negative	True Negative	Precision(%)	Recall(%)
1	Windowshop	17	0	0	229	100	100
2	Walking	85	0	1	160	100	98.84
3	Immobile	22	0	0	224	100	100
4	Shop Enter	49	0	0	197	100	100
5	Browsing	8	0	0	238	100	100
6	Shop Exit	59	1	0	186	98.33	100
7	Shop Re-enter	5	0	0	241	100	100

TABLE 5. Recognition performance comparison of multi-event models.

EventID	Context	Precision(%)			Recall(%)		
		[1]	[5]	Proposed	[1]	[5]	Proposed
1	Windowshop	54.55	80.00	100	80.00	80.00	100
2	Walking	83.33	91.67	100	97.40	100	98.84
3	Immobile	100	100	100	12.5	87.5	100
4	Shop Enter	96.30	100	100	92.86	94.64	100
5	Browsing	0	57.14	100	0	57.14	100
6	Shop Exit	95.00	100	98.33	96.61	96.61	100
7	Shop Re-enter	83.33	100	100	100	100	100
Overall accuracy		Results reported in [1]			86.38%		
		Results reported in [5]			94.47%		
		Results of the proposed model			99.59%		

$P1$ can be perfectly recognized. Besides, additional location classification can be easily checked by human operators through intuitive plots. In [1], semantic location information is introduced while still underperform.

C. COMPARISON RESULTS ON EVENT RECOGNITION

In Table 5, the performances of other Petri net-based event recognition methods on the experimental dataset are given together with that of the proposed method. In [5], the models of SE-Tree and ME-Tree allocate probability to the transition between places to realize event reasoning while this paper solves the uncertainty in reasoning via additional low-level features. Compared with the manual event model or ME-Tree, our model can be constructed automatically via proposed algorithms as well as easily interpretable. Besides the perfect performance on events owning a unique step sequence, those conflicted events are solved better with the proposed model. Confronted with identical the routine in 'Browsing'

and 'Windowshop' which cannot be solved by [5], we separate them thoroughly using the automatic classification of location information. And this solution is interpretable and reasonable. The location feature automatically supplies the deficiency of the semantic feature.

V. CONCLUSION

We present a framework for multiple event recognition in surveillance videos that exploit multi-level features. Experimental results demonstrate that the reasoning machine automatically generated by presented algorithms outperforms other Petri net models in both precision and recall. In terms of event contexts, more branches are formed using the FSR centered strategy that captures more routines of events in a natural way.

The combination with the decision tree technique provides an effective and intuitive solution to event conflicts. Observations and the trained decision tree can be shown to operators which is more interpretable and comprehensible compared with other extended Petri net models.

Besides, we provide an approach that constructs the model from documents of events in case the absence of concerned events in the training dataset. This is significant for the recognition of rare events in real-world applications, particularly anomaly detection tasks. High-level applications of video analysis require high accurate low-level techniques including object detection and tracking. Hence the lack of automatic and effective knowledge representation methodologies impede the further development of high-level applications.

REFERENCES

[1] G. Lavee, M. Rudzsky, E. Rivlin, and A. Borzin, "Video event modeling and recognition in generalized stochastic Petri nets," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 102–118, Jan. 2010.

[2] Y. Zhang, Y. Zhang, E. Swears, N. Larios, Z. Wang, and Q. Ji, "Modeling temporal interactions with interval temporal Bayesian networks for complex activity recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 10, pp. 2468–2483, Oct. 2013.

[3] A. Al-Raziqi and J. Denzler, "Unsupervised framework for interactions modeling between multiple objects," in *Proc. 11th Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, 2016, pp. 509–516.

[4] K. Kardas and N. K. Cicekli, "SVAS: Surveillance video analysis system," *Expert Syst. Appl.*, vol. 89, pp. 343–361, Dec. 2017.

[5] Z. Xiao, J. Jiang, and Z. Ming, "High-level video event modeling, recognition, and reasoning via Petri net," *IEEE Access*, vol. 7, pp. 129376–129386, 2019.

[6] W. Reisig, *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*. Berlin, Germany: Springer, 2013.

[7] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh, "AnomalyNet: An anomaly detection network for video surveillance," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 10, pp. 2537–2550, Oct. 2019.

[8] A. Ullah, K. Muhammad, J. Del Ser, S. W. Baik, and V. H. C. de Albuquerque, "Activity recognition using temporal optical flow convolutional features and multilayer LSTM," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9692–9702, Dec. 2019.

[9] S. J. Shri and S. Jothilakshmi, "Crowd video event classification using convolutional neural network," *Comput. Commun.*, vol. 147, pp. 35–39, Nov. 2019.

[10] Y. Li, R. Ge, Y. Ji, S. Gong, and C. Liu, "Trajectory-pooled spatial-temporal architecture of deep convolutional neural networks for video event detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 9, pp. 2683–2692, Sep. 2019.

[11] J. Binder, D. Koller, S. Russell, and K. Kanazawa, "Adaptive probabilistic networks with hidden variables," *Mach. Learn.*, vol. 29, no. 2, pp. 213–244, 1997.

[12] H. Buxton and S. Gong, "Visual surveillance in a dynamic and uncertain world," *Artif. Intell.*, vol. 78, nos. 1–2, pp. 431–459, Oct. 1995.

[13] E. Epaillard and N. Bouguila, "Variational Bayesian learning of generalized Dirichlet-based hidden Markov models applied to unusual events detection," *IEEE Trans. Neural Netw. Learn.Systems*, vol. 30, no. 4, pp. 1034–1047, Apr. 2019.

[14] E. Epaillard and N. Bouguila, "Proportional data modeling with hidden Markov models based on generalized Dirichlet and beta-liouville mixtures applied to anomaly detection in public areas," *Pattern Recognit.*, vol. 55, pp. 125–136, Jul. 2016.

[15] L. Caruccio, G. Polese, G. Tortora, and D. Iannone, "EDCAR: A knowledge representation framework to enhance automatic video surveillance," *Expert Syst. Appl.*, vol. 131, pp. 190–207, Oct. 2019.

[16] D. Cavaliere, V. Loia, and S. Senatore, "Towards an ontology design pattern for UAV video content analysis," *IEEE Access*, vol. 7, pp. 105342–105353, 2019.

[17] F. Ziaeetabar, T. Kulvicius, M. Tamosiunaite, and F. Wörgötter, "Recognition and prediction of manipulation actions using enriched semantic event chains," *Robot. Auto. Syst.*, vol. 110, pp. 173–188, Dec. 2018.

[18] C. Castel, L. Chaudron, and C. Tessier, "What is going on? A high level interpretation of sequences of images," in *Proc. ECCV Workshop Conceptual Descriptions Images*. London, U.K, 1996, pp. 13–17.

[19] N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis, "Representation and recognition of events in surveillance video using petri nets," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshop*, Jun. 2004, p. 112.

[20] M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V. S. Subrahmanian, P. Turaga, and O. Udrea, "A constrained probabilistic Petri net framework for human activity detection in video," *IEEE Trans. Multimedia*, vol. 10, no. 6, pp. 982–996, Oct. 2008.

[21] G. Lavee, A. Borzin, E. Rivlin, and M. Rudzsky, "Building Petri nets from video event ontologies," in *Proc. Int. Symp Vis. Comput. (ISVC)*, vol. 4841, 2007, pp. 442–451.

[22] G. Lavee, M. Rudzsky, and E. Rivlin, "Propagating certainty in Petri nets for activity recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 326–337, Feb. 2013.

[23] R. Hamidun, N. E. Kordi, I. R. Endut, S. Z. Ishak, and M. F. M. Yusoff, "Estimation of illegal crossing accident risk using stochastic Petri nets," *J. Eng. Sci. Technol.*, vol. 10, pp. 81–93, Aug. 2015.

[24] *Caviar: Context Aware Vision Using Image-Based Active Recognition*. Accessed: Jan. 7, 2006. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

JI QIU received the B.Eng. degree in electrical engineering from Beijing Jiaotong University, Beijing, China, in 2014, where she is currently pursuing the Ph.D. degree with the Electrical Engineering School. From 2017 to 2018, she was a Visiting Ph.D. Student with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI, USA. Her research interests include computer vision and intelligent video surveillance systems.

LIDE WANG received the B.Eng. and M.Eng. degrees from Southwest Jiaotong University, China, in 1982 and 1986, respectively. He is currently a Professor with the School of Electrical Engineering, Beijing Jiaotong University, and a Senior Member of the China Railway Society. His research interests include control of the electric traction systems, computer vision, and embedded system and application.

YIN WANG (Member, IEEE) received the B.Eng. degree in electrical engineering and automation from Tianjin Polytechnic University, in 2013. He is currently pursuing the Ph.D. degree with the Electrical Engineering School, Beijing Jiaotong University, Beijing, China. His research interests include computer vision and intelligent video surveillance systems.

YU HEN HU (Life Fellow, IEEE) received the B.S.E.E. degree from National Taiwan University, Taiwan, in 1976, and the M.S.E.E. and Ph.D. degrees from the University of Southern California, Los Angeles, CA, USA, in 1982. He was a member of Faculty with the Electrical Engineering Department, Southern Methodist University, Dallas, TX, USA, from 1983 to 1987. In 1999, he was a Visiting Researcher with Bell Laboratories, Homedel, NJ, USA, and Microsoft Research-China, Beijing, China. He was a Visiting Professor with National Taiwan University, in 2007 and 2015. Since 1987, he has been with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI, USA, where he is currently a Professor.