Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 167–174

Object Tracking and Video Event Recognition with Fuzzy Semantic Petri Nets

Piotr Szwed* and Mateusz Komorkiewicz*
*AGH University of Science and Technology

Email: {pszwed,komorkie}@agh.edu.pl

***Abstract*—Automated recognition of video events is an important research area in computer vision having many potential applications, e.g. intelligent video surveillance systems or video indexing engines. In this paper we describe components of an event recognition system building up a full processing chain from low-level features extraction to high-level semantic information on detected events. It is comprised of three components: object detection and tracking algorithms, a fuzzy ontology and Fuzzy Semantic Petri Nets (FSPN), a formalism that can be used to specify events and to reason on their occurrence. FSPN are Petri nets coupled with an underlying fuzzy ontology. The ontology stores assertions (facts) concerning object classification and detected relations being an abstraction of the information originating from object tracking algorithms. Fuzzy predicates querying the ontology are used in Petri net transitions guards. Places in FSPN represent scenario steps. Tokens carry information on objects participating in an event and have weights expressing likelihood of an event's step occurrence. Introduced fuzziness allow to cope with imprecise information delivered by image analysis algorithms. We describe the architecture of video event recognition system and show examples of successfully recognized events.**

***Index Terms*—video events, surveillance, fuzzy Petri Nets, fuzzy ontology**

I. Introduction

RECOGNITION of video events is an important research area in computer vision. Developed methods may have many potential applications: intelligent video surveillance, video indexing engines and various systems in which human-computer interactions are based on interpretation of video content.

Automated video event recognition comprise several tasks including detection of objects, intelligent tracking, recognition of compound events or activities and finally reasoning about occurrences of high-level events. Each of them may involve various problems to solve, e.g. how to distinguish real objects from such visual phenomena as shadows or reflexes, how to merge objects that have split into multiple segments, how to maintain objects' identities in case of occlusion, what kind of information is required to describe scene and which formalism should be used to specify events and efficiently detect them. Solutions to these problems are never perfect, each processing step may produce noisy and uncertain data, moreover a mapping between elements of semantic event specifications and low level video features often incorporate vagueness.

This work was supported from the National Centre for Research and Development (NCBiR) under Grant 0128/R/t00/2010/12 and AGH UST under Grant No. 11.11.120.612

In this paper we describe components of a video event recognition system building up a full processing chain from low-level features extraction to high-level semantic information on detected events. A conceptual layout of the systems is shown in Fig. 1. Input video sequence is analyzed with object detection and tracking algorithms. A tracking information is then represented in form of assertions in a more abstract Fuzzy Ontology layer and finally video events are detected with Fuzzy Semantic Petri Nets (FSPN), a specific class of fuzzy Petri nets, which reference terms in an ontology.

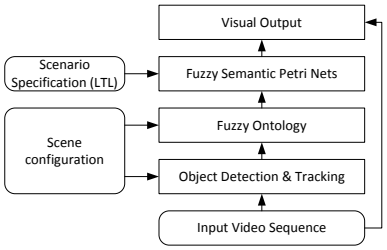


Fig. 1. Conceptual model of the event recognition system

Guards in FPNS are conjunctions of unary or binary predicates examining asserted class membership facts or object relations. Tokens in FSPN are tuples of selected scene objects participating in an event with associated weight factors. Such approach make event scenarios tolerant to classification errors, imprecise measurements and missed subevents or conditions. As transitions are fired, fuzzy weights obtained from guards evaluation are combined with token weights. In consequence, FSPN are not only capable of reasoning about scenario occurrences, but also about their likelihoods. This property is particularly important, as it allows to fine-tune the system operation by appropriately selected thresholds and filter output of less probable scenarios. Another important feature of the proposed approach is that several scenarios, starting at different time points and with different participating objects can be analyzed concurrently.

The paper is organized as follows: the next Section II reports known approaches to event specification and analysis. It is followed by Section III, in which an algorithm used to detect and track objects is briefly described; the next Section IV discusses the fuzzy ontology. FSPN are defined in Section V.

978-1-4673-4471-5/$25.00

The detection system implementing the proposed approach is presented in Section VI and finally Section VII gives concluding remarks.

II. Related works

Recognition of video events has been intensively researched over last fifteen years. A large number of approaches is reported in recent surveys: [1] and [2]. Systems for video recognition usually have a layered architecture, e.g. [3], [4], in which lower level layers provide an *abstraction* of meaningful aspects of video sequences, whereas higher level layers are related to formalisms used for *event modeling* and algorithms that detect events based on formal specifications.

Probabilistic state-based methods use models comprised of states and transitions, in which transitions are attributed with probability factors learned from annotated video. During an analysis of an input video sequence a likelihood of a situation is computed. This group include methods based on neural networks [5], Hidden Markov Models, Dynamic Bayesian Networks [6] and stochastic Petri nets [7].

In grammar based methods complex activities are represented by production rules that generate strings of atomic actions. Hence, complex events can be recognized by language parsing techniques [8], [9]. In the review [2] a limitation of these methods as regards concurrent activities was indicated. The criticism seems to be founded in a case, where sequences of single actions are analyzed. However, in a more general setting, e.g. this provided by the Kripke structure [10], each string element is a set of low level events occurring in parallel and in consequence high-level concurrent events can be tracked.

Description based approaches specify events and scenarios using high level languages, either textual [11], or graphical as Situation Graph Trees [4], [12] and Petri nets [7], [13], [14]. The methods falling into this category are considered to be *semantic*, as specifications are prepared by experts, who give meaningful names to events, engaged objects, actions and conditions. Descriptions are often hierarchical: complex events can be expressed as graphs of subevents. Models can also include constraints and knowledge about scene objects, e.g. in [4] they are expressed as formulas of a Fuzzy Metric-Temporal Horn Logic. In some approaches scenarios and their ingredients: types of participating objects and relations are defined as ontologies [15], [16].

Petri Nets (PNs) are applied in the field of event detection in two modes [1]. In the first mode of *object PNs* tokens represent objects, places object states and transitions events of interest. Such approach was applied in surveillance of traffic [13] and people [17]. In the second mode of *plan PNs* places correspond to subevents building up a plan. Presence of a token in a place indicates that a particular event assigned to the place is occurring. The latter approach was applied to parking [18] and more recently people [14] surveillance.

The semantics of Petri nets proposed in this paper is closer to *plan PNs*, as tokens represent combination of objects participating in scenarios. There are, however, some salient differences. 1) In probabilistic PNs discussed in [14] in case of a conflict (e.g. two enabled transitions sharing input place with a single token) only one transition with a higher learned probability would fire, whereas in our model they both can be executed and produce two tokens with weights aggregating the weight of the input token and transition guards. This allows to reason concurrently about scenario alternatives. Moreover, a weak initial likelihood of a scenario branch can be amplified by future events. 2) In our approach all enabled transitions are executed in a single parallel step. Such behavior rather resemble reasoning with Fuzzy Cognitive Maps [19] than the most often utilized interleaving PN semantics. 3) Petri nets modeling scenarios are actually state machines. Their structure is sufficient to construct a Büchi automaton [20] representing a LTL formula.

There are a vast number of tracking algorithms and it would be hard to present all previous works in this field. However there exist two very interesting surveys which give an in-depth view of all methods [21] and [22]. Based on classification proposed in these surveys, the tracking algorithm used in this work can be assigned to a group, which detect objects by background modeling and subtraction. The final tracking is based on kernel tracking methods (region based tracking).

III. Object detection and tracking

The detection and tracking algorithm maintains a set of tracked objects O and updates it after an arrival of a new video frame. Each object has several attributes: a history of its bounding box position and size at current and N-1 previous frames, a unique object ID, information about object type (pedestrian, graffiti, group of objects etc.) and flags denoting object occlusion or information, that object can't be tracked and its position must be estimated.

For each i-th frame the set O is updated with a procedure comprised of the following steps:

A) A background is updated and foreground object segmentation is performed.
B) A set of segments S is extracted, labeled and tracked.
C) Segments S which are similar to objects from O are assigned to them.
D) All segments S which were not assigned to O are submitted to a classification process and detected objects are added to O.
E) Overlapping objects from O are merged.
F) Merged objects from O are split, if they have separate areas.
G) Positions of objects from O, that cannot be tracked by segments, are estimated.

These steps are explained in detail in the next paragraphs:

A. Foreground object segmentation

The method is based on background generation and foreground object detection described in [23]. It consists in creating a binary mask by background substraction and processing it in order to extract connected components (segments) and label them. A single segment can be a group of objects (e.g.

a crowd), single object (e.g. a pedestrian) or part of the object (e.g. a torso). Partitioning may be caused by failures of segmentation algorithm.

B. Segment tracking

Number of segments, their position and size may vary significantly from frame to frame. Segmentation errors can be caused by many factors e.g. camera noise, changes in lighting condition, shadows, occlusion by other objects etc. To make sure that only segments, which are correctly extracted will be used in the next processing stage, a simple tracking mechanism is applied that is based on checking bounding box positions of current and previous segments. If two bounding boxes overlap in two consecutive frames, segments are linked with the *history* relation describing evolution of segments in time. For further analysis only segments, which have clear history based on observation form N previous frames are used.

C. Segments to object assignment

In the next step all segments S detected in a frame i are assigned to objects from previous frame O. Let us introduce a function $h\colon O \times \mathbb{N} \to 2^S$ that defines a mapping between an object o at a frame i and a set of segments. The Algorithm 1 takes at input the sets of objects O and segments S, updates the function h and computes the set of assigned segments S_u.

Algorithm 1

procedure ASSIGN(O, S, i, S_u, h)
  for all $o \in O$ **do**
    for all $s \in S$ **do**
      if $d(o, s) \geq \epsilon$ **then**
        Add segment to an object:
        $h(o, i) \leftarrow h(o, i) \cup \{s\}$
        Add s to the set of used segment S_u:
        $S_u \leftarrow S_u \cup \{s\}$
      end if
    end for
  end for
end procedure

The function d calculates a normalized to $[0, 1]$ similarity between an object o and a segment s considering overlapping of segments in $h(o, i-1)$ and s (bounding boxes or image masks). The threshold ϵ is a small constant, e.g. 0.1.

After executing the procedure new positions and sizes of all objects $O_u = \{o \in O\colon h(o, i) \neq \emptyset\}$ are computed based on position and size of segments assigned to them.

D. Object detection and classification

All segments, which were not assigned to any object ($S_{NA} = S \setminus S_u$) are further analyzed by a set of classifiers. Simple geometrical rules are applied (perspective is compensated) e.g. an adult person should be higher than 160 cm and wider than 40 cm. If a segment with desired properties is detected, a new object is created o_{new} with a unique ID and added to object list $O \leftarrow O \cup \{o_{new}\}$. If an object is divided into several segments, a correct classification is not possible. In this case the system will not detect it on the current frame. As the video sequence is analyzed, it is very likely that in next few frames it will appear not split and the algorithm will be able to detect it. The benefit of this approach is that only very reliable objects are detected, what leads to smaller false detection rate.

E. Object merging

In the next processing step, the algorithm handles cases, when two previously tracked objects merge, e.g. if two pedestrians approach and finally their silhouettes overlap. As it is then impossible to track objects based on the information on segments position and history, the algorithm sets a special flag in the overlapping objects descriptors. From that time their positions are not computed based on the position of segments belonging to them. Instead, a prediction mechanism is used, which is based on information about previous sizes, movement direction and speed. Also a new temporal object is created (so called merged object). Its size and position is updated based on segments which previously belonged to merged objects. Thanks to this, the system is able not only to estimate the object position, but also to keep track of real area occupied by the estimated objects.

F. Object splitting

In the next step it is checked, if previously merged objects are split. Such situation occurs, e.g. when two previously joined pedestrians move away far enough to allow for total separation of segments belonging to each object. In this case, all segments are checked to test, if they resemble an object within a merged group. If appropriate correspondences can be found, the unique IDs are restored based on estimated positions of the original objects. Another scenario is also possible. A person may leave a luggage and start to move away (possible bomb planting scenario). In such case it is possible that an object, which has only one ID starts to split. To cover such situations, a maximum object size is checked and, if it exceeds the maximum allowed object size, the tracking object is removed and classification is rerun for all segments belonging to this object. The largest detected object is given the old ID, all other detected objects are given new unique IDs.

G. Position estimation

In the last step, for all objects, which are marked as lost or impossible to track, i.e. $h(o, i) = \emptyset$, a history of their positions on previous frames is analyzed to compute a mean velocities. Object's velocity is then used to estimate a new position. The object size is not estimated, the last known size is used instead. This estimation method is working well in most cases. It can also cause wrong results, if objects change speed or movement direction during the estimation. To overcome this types of errors, a guard mechanism was introduced. It is based on counting the number of pixels belonging to foreground objects within an estimated bounding box. If it drops below a fixed

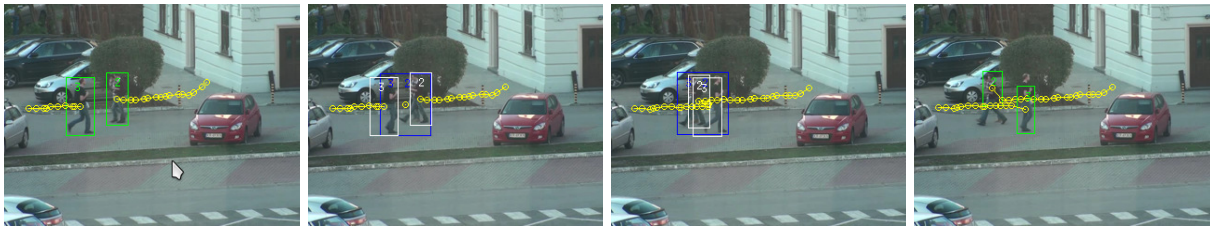


Fig. 2. Sample tracking sequence, green boxes - tracked pedestrians, white boxes - estimated pedestrians positions, blue boxes - group position, yellow circles - trajectories

threshold, it is a signal that the object position is not estimated correctly. In such case the estimated object is removed.

A sample tracking sequence of two passing pedestrians is presented in Figure 2.

IV. Fuzzy ontology

The fuzzy ontology constitute an intermediate layer between information on tracked objects and fuzzy Petri nets. Whereas objects within the tracking model are described with numeric values, like size, distance or speed, the ontology provide a kind of linguistic abstractions, e.g. *a small object*, *objects are close* or *a person is walking*.

There are several benefits of this approach:

- Scenario specifications can be prepared in a more general and meaningful manner, they can be decoupled from the code and implementation details can be hidden .
- It is much easier to customize a recognition system to specific needs and conditions, because the translation between numeric values and linguistic terms is accomplished in isolated and easy to identify functions
- Facts concerning classifications of objects and detected relations are materialized, hence they can be evaluated only once, what generally increases performance.

Ontologies are often described as unions of two layers: terminological (*TBox*) and assertional (*ABox*). The *TBox* defines concepts and types of relation including: taxonomic relations between concepts, *object properties* and *datatype properties*. The *ABox*, in turn, gathers facts about individuals and existent relations. In Description Logic, being a counterpart of ontology languages, concepts and relations can be expressed by means of unary and binary predicates, e.g.: $Person(x)$ – x is a member of the class $Person$, $isWalking(x)$ – a boolean datatype property $isWalking$ of an individual x or $isClose(x, y)$ – an object property between two individuals x and y.

For *fuzzy ontologies* and corresponding Fuzzy Description Logics the ontology relations are extended by adding weights being real numbers from $[0, 1]$. They can be used to express uncertainty, e.g. with respect to class membership or relation occurrence. Formalizations of fuzzy ontology languages including fuzzy classes, roles (object properties) and datatype properties can be found in [24] and [25].

In the case of a fuzzy ontology used with FSPN, its TBox is a stable part, whereas the ABox is updated for each frame. A crucial element of the described approach is that relations in the ABox are practically never fully evaluated. Only their subset that is requested from FSPN is calculated by making calls to plugged in functions (function objects in object-oriented implementation) called *evaluators*. They examine the tracking model and calculate fuzzy weights of predicates. In opposition to approach proposed in [25] evaluators are external entities beyond the ontology. In many cases they have a form of membership functions described by line segments, as in Fig. 3, but they can be also based on other features, as Jaccard metrics applied to object areas (Fig. 4). In this case a bounding box of a detected object is divided into a $n \times m$ grid and each cell is assigned with a probability density p_{ij}. The weight returned by the evaluator is calculated according to the formula: $w = \frac{1}{Z} \sum_{i=1}^{n} \sum_{j=1}^{m} h_{ij} p_{ij}$, where Z is a normalizing constant $Z = \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij}$ and $h_{ij} = 1$ if a cell (i, j) intersects with an object or 0 in other case. The selection of probability distribution is arbitrary and depends on the type of interaction, e.g. the grid in Fig. 4a was used to calculate intersection values for vertical objects, e.g. walls, whereas the grid in Fig. 4b for horizontal ones, e.g. forbidden zones on a floor.

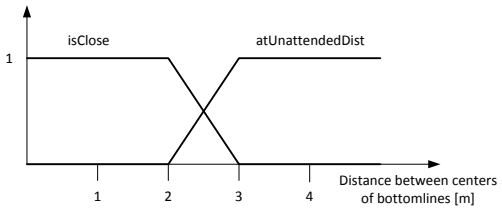
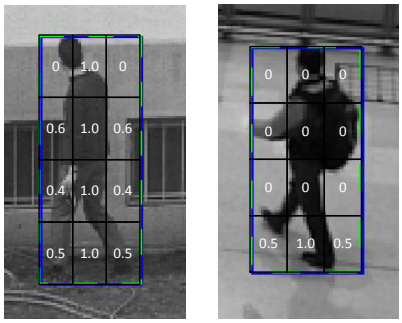


Fig. 3. Membership functions used by *evaluators*

V. Fuzzy Semantic Petri Nets

In this section we define Fuzzy Semantic Petri Nets and describe their behavior. FSPNs are placed at the top of video event recognition stack (Fig. 1 and are responsible for interpretation of low-level events and conditions represented as assertions in a sequence of ABoxes of the coupled fuzzy ontology.

It should be noted, that the presented semantics of FSPN is dedicated to a particular case of state machines, i.e. Petri nets, in which transitions link single places. Such restrictions



a. b.

Fig. 4. Evaluators based on Jaccard metrics.

stems from fact that we use Linear Temporal Logic (LTL) [26], [27] to specify video events. LTL specifications are then transformed to corresponding FSPN structures following the rules for translating them to Büchi automata [20].

Relations between FSPN and LTL are even deeper, an input sequence of ABoxes analyzed by a FSPN can be considered a specific kind of Kripke structure [10], which is defined as a sequence of states $s_0, s_1, s_2, \ldots, s_n, \ldots$ and a function that assigns sets of true propositions P_i to states s_i. In our case a state s_i corresponds to an i-th video frame and a set of propositions P_i to a set of assertions in an i-th ABox of the fuzzy ontology. Hence, detection of a video event can be considered a checking if a model (a sequence of ABoxes) satisfies an LTL formula that is translated to a FSPN.

In the presented approach we generalize and relax the acceptance requirements:

- Instead of Büchi automata, fuzzy Petri nets are used as a tool for scenario analysis. This allows to process concurrently scenarios, in which participate various combinations of objects.
- To manage uncertainty and inexactness of input data, fuzzy predicates returning values from $[0, 1]$ are used. These values are then combined with weights of tokens flowing through a net. Tokens, in turn, represent scenario occurrences. This enables monitoring scenario steps and reasoning about their likelihood.
- Sequences of accepted states strictly defined with LTL formulas can be interleaved with states not satisfying the specified conditions. In such case, the weight determining scenario satisfaction gradually decrease and, after passing a certain threshold, the scenario is rejected by token removal.

A. Definition of Fuzzy Semantic Petri Nets

Formal definition of Fuzzy Semantic Petri Nets is comprised of three concepts: Petri net structure, binding and fuzzy marking. We start with some auxiliary definitions. Unary predicate is defined as a pair (n, v_s) where, n is a predicate name and v_s is a variable name referring to a *subject* of the predicate. Binary predicate is a triple (n, v_s, v_o); the variable v_o is a predicate object. Set of all unary and binary predicates is denoted by $Preds$. By $Vars(p)$ we denote a set of variables appearing in the predicate p. Analogously, for a set $C \subseteq Preds$ we define $Vars(C)$, as $\bigcup_{p \in C} Vars(p)$.

Definition 1 (Petri net structure)**.** Petri net structure PN is a tuple $(P, T, F, Preds, G, L, H)$, where P is a set of places, T is a set of transitions, P and T are satisfying $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$. $F \subseteq P \times T \cup T \times P$ is a set of arcs (flow relation), and $Preds$ is a set of unary and binary predicates. $G\colon T \to 2^{Preds}$ is a guard function that assigns sets of predicates to transitions. $L\colon P \to \mathbb{N} \cup \{0\}$ is a function assigning lower bound to a place; this value defines how long a token should stay in a place to be allowed to leave it. $H\colon P \to \mathbb{N} \cup \{\omega\}$ assigns upper bound to a place. The symbol ω represents infinity.

Following [28] the set of input places for a transition $t \in T$ is denoted as $\bullet t = \{p \in P\colon (p, t) \in F\}$ and the set of output places as $t\bullet = \{p \in P\colon (t, p) \in F\}$

Definition 2 (Binding)**.** Let V be set of variables and I a set of objects. *Binding* b is defined as a partial function from V to I. A variable v *is bound* for a binding b, iff $v \in \operatorname{dom} b$. A set of all bindings is denoted by B.

Let $p \in Preds$ a predicate and $b \in B$ be a binding. Predicate value for a binding $val\colon Preds \times B \to [0, 1]$ is a function that assigns value from the interval $[0, 1]$ to a pair (p, b), $p \in Preds$ and $b \in B$. If $Vars(p) \setminus \mathbf{dom}\, b \neq \emptyset$, then $val(p, b) = 0$.

Definition 3 (Fuzzy marking)**.** A set of fuzzy tokens FT is defined as $FT = B \times \mathbb{R} \times (\mathbb{N} \cup \{0\}) \times (\mathbb{N} \cup \{0\})$. Components of a token tuple $(b, w, c, \tau) \in FT$ are the following: $b \in B$ denotes a binding, $w \in [0, 1]$ is a fuzzy weight, $c \geq 0$ is a counter storing information, how long a token rests in a place and τ is a time stamp. *Fuzzy marking* for a Petri net $PN = (P, T, F, Preds, G)$ is defined as a function that assigns sets of fuzzy tokens to places $FM : P \to 2^{FT}$.

B. Execution

The behavior of FPNs defined in previous section differs from the standard semantics for Petri nets, as they are not intended to focus on concurrency and conflicts, but to perform a kind of fuzzy reasoning and classification of sequences of events.

Single i-th step of execution of a fuzzy Petri Net is comprised of three basic stages:

1) *Firing enabled non-initial transitions and generating new tokens.* During this stage each token-transition pair (t, m), where $t \in T$ and $m = (b, w, c, \tau) \in FM(\bullet t)$ is analyzed. If a guard $G(t)$ references unbound variables, i.e. $Vars(G(t)) \setminus \mathbf{dom}\, b \neq \emptyset$, an attempt is made to create a new binding b' by grounding free variables with ontology individuals; in other case $b' = b$. Then, a weight of the guard is calculated: $w_g = \min\{val(p, b)\colon p \in G\}$ and aggregated with the old token weight: $w' = a\, w_g + (1 - a)w$. If w' is greater than a certain threshold (in experiments 0.2 value was used), a new token $m' = (b', w', c', t')$ is created and put

into the transition's output place $t\bullet$. The current iteration number i is assigned as token timestamp $\tau' = i$ and, if the transition t is a self-loop, the counter is updated: $c' = c + 1$. It should be mentioned, that only self-loop transitions can fire if token counter c does not belong to the interval $[L(\ bullett), H(\bullet t)]$ (see Definition 1).

2) *Removing old tokens.* A transition occurrence performed in the previous stage can be regarded as a triple (m, t, m'), where m is an input token, m' an output token and $t \in T$ a transition. Let C denote a set of such triples, and $w(m)$ a weight of a token. For each input token m a sum of weights of output tokens m' is calculated and subtracted from its weight: $w_{new}(m) = w(m) - \sum_{(m,t,m')\in C} w(m')$. If the value falls below a certain threshold, the token m is removed. Also in this step multiple tokens having the same binding and assigned to the same place are aggregated.
3) *Firing initial transitions.* Finally, new tokens are introduced into the net, by firing initial transitions (i.e. satisfying $\bullet t = \emptyset$). For each initial transition variables appearing in its guard are bound to objects, then the guard value is calculated and used as a weight of new tokens. To avoid analyzing scenarios with low likelihoods, a threshold preventing from creating tokens with small weights is defined. The mechanism is also protected against introducing tokens with a binding already present in the net.

C. Video event specification

We start preparing a specification of a video event by outlining a general scenario in form of Temporal Logic formula, then events appearing in the scenario are refined into conjunctions of low-level events or conditions expressed as predicates. The resulting LTL specification is used in two ways: (1) it is translated into FSPN and (2) predicates are included into TBox of the fuzzy ontology.

To give an example: a high-level video event, in which a person violates a forbidden zone can be expressed in LTL as a sequence of three medium-level events: $init \Rightarrow \Diamond move \Rightarrow \Box violate$, where $init$ defines conditions to start recognition, $move$ denotes a situation, when a person is moving towards a zone and $violate$ a situation, when the person enters the zone. During the refinement step the scenario is transformed into formula (1), which is further translated into FSPN shown in Fig. 5.

$$\begin{aligned} &Person(x) \wedge isWalking(x) \wedge atBorder(x) \wedge Zone(y) \\ &\Rightarrow \Diamond(isWalking(x) \wedge movesTowardsZone(x,y)_{\{8,\infty\}} \\ &\Rightarrow \Box(bottomInZone(x,y))_{\{4,\infty\}} \end{aligned} \quad (1)$$

Fig. 6 shows a FSPN defining a complex event, during which a person leaves unattended luggage. Its scenario (in a narrative form) with accompanied video material was published as a benchmark for PETS 2006 workshop [29]. The event is defined as a sequence of four simple steps: *init* – a still person appears, *separate* – the person puts a luggage on the floor and remains close to it, *leave* – distance between the person and luggage grows above a certain threshold (equal to 3 meters in PETS specification) and finally *remain* – the person disappears and the luggage remains alone.

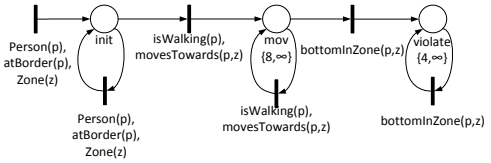


Fig. 5. Fuzzy Semantic Petri Net representing a scenario, in which a person violates a forbidden zone

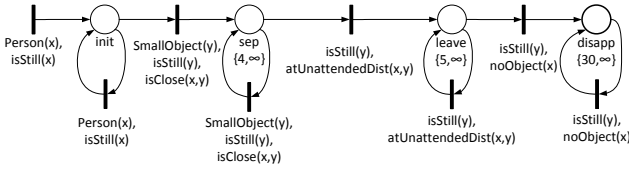


Fig. 6. Fuzzy Semantic Petri Net representing luggage left scenario

A more complex FSPN is presented in Fig. 7. It defines an event of graffiti painting on a wall decomposed into medium level events: a person moves towards a wall, then appears in front of the wall, optionally: a person is widening (what may indicate painting graffiti), then a new object emerges on a wall (but not inside a window) and remains still.

VI. Detection system

In this section we describe a prototype system allowing to test recognition of events based on specifications in FSPN. The system takes at input a video sequence with an XML file defining tracking information. For each frame a list of segments and identified objects is provided. The data originate from tracking algorithms described in Section III. The architecture of the prototype scenario detection system is presented in Fig. 8. Main components are: the *Fuzzy ontology*, a set of *Evaluators* and the *Fuzzy Semantic Petri Net* execution engine. The system is also equipped with GUI providing visual output shown in Fig. 9.

The control flow during a single iteration was marked in Fig. 8 with numbers in circles.

1) After a new frame appears, asserted relations between objects are removed from the ontology, then newly identified objects are added as individuals.
2) In the next step all enabled transitions in concurrently analyzed Petri nets are fired. Preparation of transitions requires calculations of guards and in some cases extensions of bindings.
3) In order to obtain weights of predicates appearing in guards, appropriate queries are made to ontology. If a weight for a predicate was evaluated earlier, it is immediately returned.
4) In other case an *evaluator* assigned to the predicate is called, and returned value is asserted in the ontology as a weight of corresponding fuzzy relation.

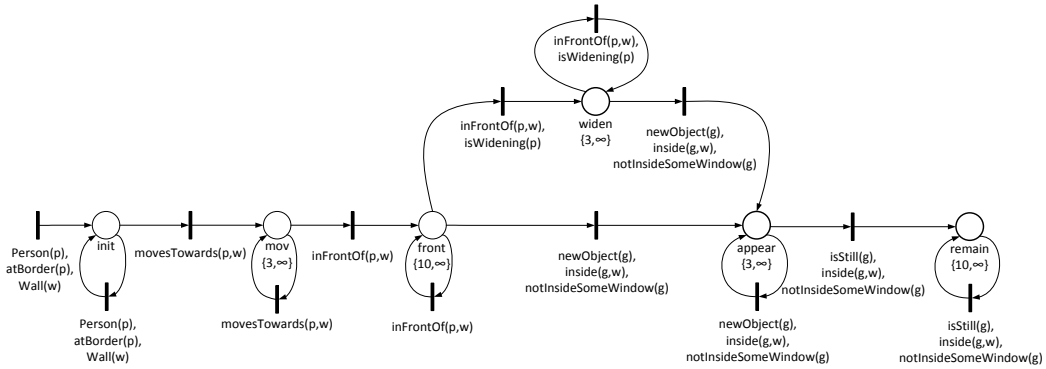


Fig. 7. FSPN representing graffiti painting scenario

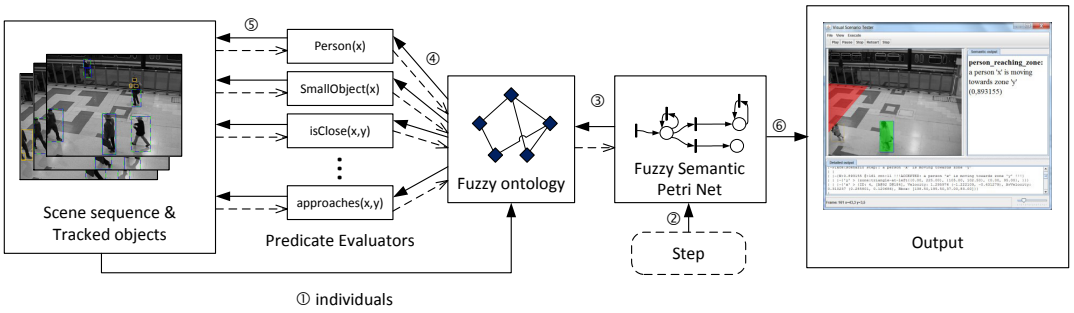


Fig. 8. Architecture of the detection system

5) Evaluators examine the tracking information. As a tracking history covering a number of past frames is kept, evaluators are capable of returning temporal properties, e.g. $newObject(x)$ - an object is considered *new* if it has recently appeared.
6) After the net state is updated, a reached marking is analyzed. If a token stays in a selected place long enough (observed places are defined in FSPN specification) its presence is reported as an important scenario step or a final stage.

Fig. 9. Visual scenario tester. The displayed video frame and messages correspond to the place *leave* ($isStill(y) \wedge atUnattendedDist(x, y)$) of the FSPN specifying luggage left scenario.

The software is entirely written in Java. Its performance is quite good: for three concurrently analyzed scenarios and a scene with a few tracked objects, a single reasoning iteration, during which the ontology is updated, evaluators are called and multiple transitions in Petri nets are fired, is executed within 0.1ms to 1.6ms (average 0.45 ms) on a Pentium i7 2.2 GHz machine.

The system was successfully tested to recognize a number of events, including these specified in Fig. 5, Fig. 6 and Fig. 7, returning in each case high likelihood value close to 1.0.

Fig. 10 presents visual output corresponding to the steps of the graffiti painting scenario specified by FSPN in Fig. 7. Filled bounding boxes mark objects included into the binding of tokens that reached places (scenario steps), for which semantic messages are displayed.

VII. Conclusions

In this paper we describe components of video events recognition system building a full processing chain: from objects detection and tracking, through transforming tracking information into more abstract representation of Fuzzy Ontology, to reasoning on events occurrences with Fuzzy Semantic Petri Nets.

An advantage of FSPN is their capability of detecting concurrently occurring events, in which participate various combinations of objects, analyze scenario alternatives and their likelihoods. Petri nets state (marking) gives general overview of the situation, of *what's going on*. A presence of a token in a

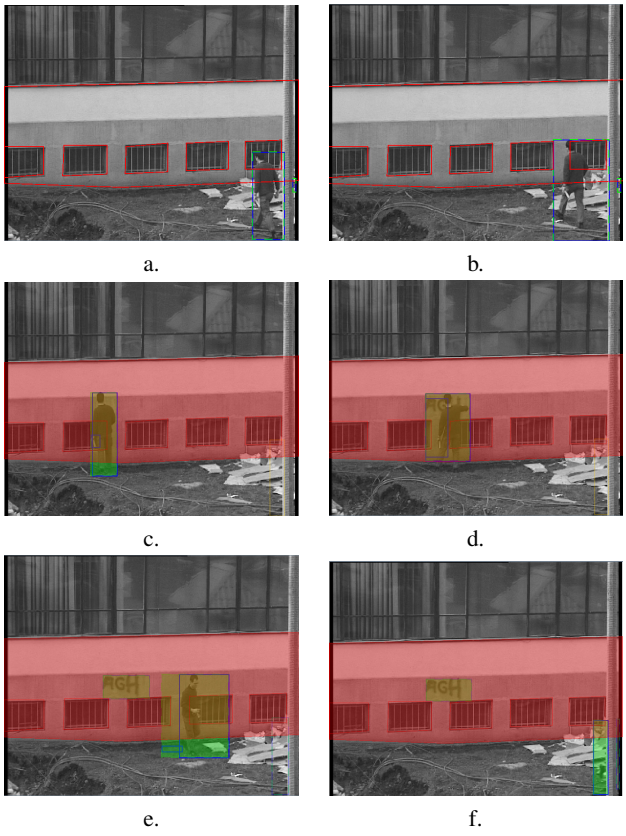


Fig. 10. Recognized steps of a graffiti painting event: a – *init*, b – *move*, c – *front*, d – *widen*, e – *appear*, f – *remain* (scenario completion)

place can be reported as *semantic output*, e.g. to a surveillance system operator.

The proposed scenario detection system is a generic framework, that can be adapted to specific needs by: 1) defining an ontology including classes of objects and relations of interest; 2) implementing evaluators, i.e. functions responsible for calculating values of fuzzy predicates, and plugging them into the framework; 3) configuring scene objects (their types must be defined in the ontology) 4) writing a scenario using in formulas entities (classes and relations) from the ontology.

REFERENCES

[1] G. Lavee, E. Rivlin, and M. Rudzsky, "Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 5, pp. 489–504, Sept. 2009.

[2] J. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[3] F. Brémond, M. Thonnat, and M. Zúniga, "Video-understanding framework for automatic behavior recognition," *Behavior Research Methods*, vol. 38, no. 3, pp. 416–426, 2006.

[4] D. Munch, J. Jsselmuiden, M. Arens, and R. Stiefelhagen, "High-level situation recognition using fuzzy metric temporal logic, case studies in surveillance and smart environments," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, Nov. 2011, pp. 882–889.

[5] M. Barnard, J.-M. Odobez, and S. Bengio, "Multi-modal audio-visual event recognition for football analysis," in *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, Sept. 2003, pp. 469–478.

[6] J. Aggarwal and S. Park, "Human motion: modeling and recognition of actions and interactions," in *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, Sept. 2004, pp. 640–647.

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

[8] S.-W. Joo and R. Chellappa, "Attribute grammar-based event recognition and anomaly detection," in *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, June 2006, pp. 107–107.

[9] G. Guerra-Filho and Y. Aloimonos, "A language for human action," *Computer*, vol. 40, no. 5, pp. 42–51, May 2007.

[10] S. Kripke, "Semantical considerations on modal logic," *Acta philosophica fennica*, vol. 16, no. 1963, pp. 83–94, 1963.

[11] V.-T. Vu, F. Bremond, and M. Thonnat, "Automatic video interpretation: A novel algorithm for temporal scenario recognition," in *International Joint Conference on Artificial Intelligence*, vol. 18. Lawrence Erlbaum Associates Ltd, 2003, pp. 1295–1302.

[12] H.-H. Nagel, "Steps toward a cognitive vision system," *AI Magazine*, vol. 25, no. 2, p. 31, 2004.

[13] N. Ghanem, D. DeMenthon, D. Doermann, and L. Davis, "Representation and recognition of events in surveillance video using Petri nets," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*, June 2004, pp. 112–112.

[14] 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," *Multimedia, IEEE Transactions on*, vol. 10, no. 8, pp. 1429–1443, Dec. 2008.

[15] F. Bremond, N. Maillot, M. Thonnat, V.-T. Vu *et al.*, "Ontologies for video events. Research report number 51895," INRIA Sophia-Antipolis, Tech. Rep., 2004.

[16] U. Akdemir, P. Turaga, and R. Chellappa, "An ontology based approach for activity recognition from video," in *Proceedings of the 16th ACM international conference on Multimedia*. ACM, 2008, pp. 709–712.

[17] A. Borzin, E. Rivlin, and M. Rudzsky, "Surveillance event interpretation using generalized stochastic Petri nets," in *Image Analysis for Multimedia Interactive Services, 2007. WIAMIS'07. Eighth International Workshop on*. IEEE, 2007, pp. 4–4.

[18] C. Castel, L. Chaudron, and C. Tessier, "What is going on? a high level interpretation of sequences of images," 1996.

[19] J. Aguilar, "A Survey about Fuzzy Cognitive Maps Papers (Invited Paper)," *International Journal*, vol. 3, no. 2, pp. 27–33, 2005.

[20] J. R. Büchi, "On a Decision Method in Restricted Second-Order Arithmetic," in *International Congress on Logic, Methodology, and Philosophy of Science*. Stanford University Press, 1962, pp. 1–11.

[21] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006.

[22] S. Vishwakarma and A. Agrawal, "A survey on activity recognition and behavior understanding in video surveillance," *The Visual Computer*, pp. 1–27, 2012. [Online]. Available: http://dx.doi.org/10.1007/s00371-012-0752-6

[23] T. Kryjak and M. Gorgoń, "Real-time implementation of moving object detection in video surveillance systems using FPGA," *Computer Science Journal, Wydawnictwa AGH*, vol. 12, pp. 149–162, 2011.

[24] S. Calegari and D. Ciucci, "Fuzzy ontology, Fuzzy Description Logics and fuzzy-OWL," *Applications of Fuzzy Sets Theory*, vol. 4578/2007, no. D, pp. 118–126, 2007.

[25] T. Lukasiewicz and U. Straccia, "Managing uncertainty and vagueness in description logics for the Semantic Web," *Web Semantics Science Services and Agents on the World Wide Web*, vol. 6, no. 4, pp. 291–308, 2008.

[26] Z. Manna and A. Pnueli, "Temporal logic," in *The Temporal Logic of Reactive and Concurrent Systems*. Springer New York, 1992, pp. 179–273.

[27] M. Fisher, *An Introduction to Practical Formal Methods Using Temporal Logic*. Wiley, 2011.

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

[29] PETS 2006, "Ninth IEEE international workshop on performance evaluation of tracking and surveillance - benchmark data," 2006, [Online; accessed 04-March-2013]. [Online]. Available: http://www.cvg.rdg.ac.uk/PETS2006/data.html