An APRIORI-based Method for Frequent Composite Event Discovery in Videos

Alexander Toshev
University of Pennsylvania
Philadelphia, PA 19104
toshev@seas.upenn.edu

François Brémond Monique Thonnat
INRIA, 2004 route des Lucioles BP 93
Sophia Antipolis, France
{fbremond,thonnat}@sophia.inria.fr

Abstract

We propose a method for discovery of composite events in videos. The algorithm processes a set of primitive events such as simple spatial relations between objects obtained from a tracking system and outputs frequent event patterns which can be interpreted as frequent composite events. We use the APRIORI *algorithm from the field of data mining for efficient detection of frequent patterns. We adapt this algorithm to handle temporal uncertainty in the data without losing its computational effectiveness. It is formulated as a generic framework in which the context knowledge is clearly separated from the method in form of a similarity measure for comparison between two video activities and a library of primitive events serving as a basis for the composite events.*

1 Introduction

The problem of video event recognition has been studied extensively ([7]). In most of the approaches explicit models of events are used which are either created manually or learned from labelled data. In this work we focus on the problem of detecting frequent complex activities without a model.

In this work an event is a spatio-temporal property of an object in a time interval or a change of such a property ([12]). An example of an event at a parking lot is 'vehicle on the road' (see fig. 1(a)). For the recognition of events an algorithm was developed in the above work as a component of a system for video event recognition called `VSIP` ([2]). The events are formally defined in an event description language which allows us to define complex events in terms of simpler ones and in this way to build a hierarchical structures of events. For instance, in a parking lot a complex event is a 'parking manoeuvre' which consists of 'a vehicle on the road', 'a vehicle on the parking road', 'vehicle on a parking place' and 'person coming out from the vehicle'. The simplest events at the bottom of this description are referred as primitive while the complex ones are called composite. In order to provide a complete description of a domain an extensive library of events is needed containing a formal description for each possible behavior. To simplify the library and decrease the deployment efforts we retain in this library only the most simple and general events which are the primitive ones. These are simple spatial relations like 'object in a zone' or 'object near another object'. The frequent composite events are deduced in an automatic way from the set of all detected primitive events.

For the latter task we adapt the data mining APRIORI algorithm ([1]) which uses the so called APRIORI property: the subpatterns of frequent patterns are also frequent. Therefore, starting with short patterns we count the occurrence only of those patterns whose subpatterns were marked as frequent in a previous step. In this way the search space is reduced. A pseudo code can be found in Algorithm (1). In line 3 all pairs of patterns of length $i-1$ from the previous iteration which have exactly $i-2$ mutual elements are merged into patterns of length i. In the next step equal patterns are combined in classes. Finally, only the classes with a size greater than a given size threshold s_{th} are retained.

Algorithm 1 APRIORI(S, l, s_{th})

INPUT: A set of states $S = \{s_1 \dots s_n\}$.
OUTPUT: All l-pattern classes $C^{(l)}$ with size $s(C^{(l)})$ greater than a given size s_{th}.

1: **natural** $i \leftarrow 2$; **pattern set** $P_i \leftarrow \emptyset$; $K_1 \leftarrow \{\{s_1\} \dots \{s_n\}\}$
2: **while** $i \leq l$ **do**
3: $P_i \leftarrow$ CREATENEXTLENGTHPATTERNS(K_{i-1}, i)
4: $\hat{K}_i \leftarrow$ COMBINEPATTERNSINTOCLASSES(P_i)
5: $K_i \leftarrow$ RETAINFREQUENTPATTERNS$(\hat{K}_i, s_{th})$
6: $i \leftarrow i + 1$
7: **end while**
return K_l

The difficulties of a direct application of the above algorithm in the domain of video event analysis arise from the uncertainty of the data: there are no equal but only similar

occurrences of the same behavior type. Precisely, we have to answer two questions:

(i) How do we measure the number of occurrences of an activity (its frequency) and how the frequency of a subactivity is related to the frequency of an activity (line 5)?

(ii) How do we decide which patterns of events represent the same event type (line 4)?

The answer to these questions depends on the way we compare event patterns. Therefore, a similarity measure is necessary which evaluates to which extend two patterns represent the same activity. Using this similarity as a basis we can answer in a domain-independent way the above questions. In particular, the number of occurrences of a behavior and thus implicitly its frequency can be defined in a soft manner taking into account the similarity as shown in section 3. Additionally, the APRIORI property does not hold in case of similarity because subpatterns of patterns can be less similar than the patterns themselves. Therefore, we formulate in section 6 a WEAK-APRIORI property which decreases the frequency threshold for shorter patterns in order to prevent losing subpatterns of frequent patterns and thus to guarantee their detection in the merge step in line 3. The second point - combination of patterns into classes - uses an entropy-based clustering algorithm presented in section 5.

The domain knowledge is provided in two forms: besides the similarity we must specify a library of generic primitive event types. The occurrences of these events are the input to the algorithm and therefore serve as a basis for the composite events which can be detected.

We present a generic framework for high-level frequent event discovery. The context knowledge is clearly separated from the algorithm and thus makes the approach applicable in different domains for which primitive events and similarity can be defined. By solving the above issues we guarantee robust event detection in noisy environments. Moreover, the algorithm outputs a set of composite events which are hierarchically ordered and thus generates a clear event structure.

2 Related Works

Although the research in the field of unsupervised event detection and learning is at its beginning there are several approaches studied.

One of the most widely used techniques is to learn in an unsupervised manner the topology of a Markov model. [3] use an entropy-based function instead of the Maximum-Likelihood estimator in the E-step of the EM-algorithm for learning parameters of Hidden Markov Models (HMM). This leads to a concentration of the transitional probabilities just on several states which correspond in most of the cases to meaningful events. Another approach is based on variable length Markov models which can express the dependence of a Markov state on more than one previous states ([6]). While this method learns good stochastic models of the data it cannot handle temporal relations. A further similar technique is based on hierarchical HMMs whose topology is learned by merging and splitting states ([15]). The advantage of the above techniques for topology learning of Markov models is that they work in a completely unsupervised way. Additionally, they can be used after the learning phase to recognize efficiently the discovered events. On the other hand, these methods deal with simple events and are not capable of creating concept hierarchies. The states of the Markov models do not also correspond always to meaningful events.

Another method was proposed by [10] who use inductive logic programming to generalize simple events. Although being promising this system was developed only for simple interactions without taking into account any temporal relations. For low-level event detection and learning several standard techniques were also used. [8] use Self-Organizing Maps to learn typical trajectories of moving objects. [16] learn gestures by extracting and clustering prototypes of gesture components from trajectories of hand movements using the k-means algorithm. The authors use the Minimum Description Length principle to determine the optimal number of gesture components. All these approaches perform well in the case of the problem they are specified for but cannot be generalized.

The data mining community has been studying the task of frequent pattern extraction for several decades. However, more emphasis is put on the computational effectiveness than on the robustness against noise. There are only a few approaches coping with some specific problems arising from uncertainty: [14] deal with false attribute values; temporal variability is addressed in [11]. Unfortunately, they do not propose a general way of dealing with different types of uncertainty.

3 A Model of Frequent Patterns

In this section we introduce some basic notation. We are interested in reoccurring structures in a primitive event set, expecting that these structures correspond to meaningful complex activities:

Definition 1 *An **m-event pattern** p is a set of primitive events with cardinality m: $p = \{e_1 \dots e_m\}$. The set of all m-event patterns is denoted by $P^{(m)}$. A **subpattern** of a pattern p is a pattern $\hat{p}$ whose events are contained in p.*

Although a pattern p is just a set it describes implicitly a structure through the relations between its events. Addi-

tionally, a similarity measure is needed in order to express the degree of similarity between patterns:

Definition 2 *A **similarity measure** $\mathrm{sim}^{(m)}(p_1, p_2)$ of order m between two m-patterns p_1 and p_2 is a mapping $\mathrm{sim}^{(m)} : P^{(m)} \times P^{(m)} \rightarrow [0, 1]$, where $P^{(m)}$ is the set of all m-patterns.*

The concrete form of this mapping depends on the domain and therefore it is not possible to require further properties. Using the above notions of patterns and similarity we can build pattern classes:

Definition 3 *An m-**pattern class** $C^{(m)}$ of order m is a set of m-event patterns. A **subclass** of a class C is a class $\hat{C}$ whose patterns are subpatterns of patterns in C.*

We expect that a pattern class stands for a behavior type and thus contains examples of this behavior which should be similar to each other. This property is guaranteed through the class building step (line 4 in Algorithm 1) which is realized as clustering maximizing the weighted entropy of the class (see section 5).

A frequent subclasses of a class stands also for a frequent shorter activity which should be described additionally if it occurs more frequently than activity represented by the class itself. Using the APRIORI-algorithm (1) from section 1 we can maintain links from subclasses to classes formed in the next loop of the algorithm and retain the subclasses containing more examples than the number of examples in the class. In this way a hierarchical description of composite events can be achieved.

Based on the above similarity we can define in a domain-independent way a measure for the membership of a pattern in a pattern class and the pattern class size:

Definition 4 *A **class similarity measure** $\mathrm{sim}^{(m)}(p, C)$ between an m-event pattern p and a pattern class C of order m is a mapping $\mathrm{sim}^{(m)} : P^{(m)} \times K^{(m)} \rightarrow [0, 1]$:*

$$\mathrm{sim}^{(m)}(p, C) \stackrel{def}{=} \frac{1}{|C|} \sum_{p' \in C} \mathrm{sim}^{(m)}(p, p') \qquad (1)$$

where $P^{(m)}$ is the set of all m-patterns and $K^{(m)}$ is the set of all pattern classes of order m.

Definition 5 *The **size** $s(C)$ of a pattern class C is the sum of the similarities between all patterns in C to C:*

$$s(C) \stackrel{def}{=} \sum_{p \in C} \mathrm{sim}(p, C) \qquad (2)$$

A large class size means both a large number of examples of the behavior represented through the class (large number of summands in eq. (2)) and also high degree of representativeness of the behavior through its examples (high value of the summands in eq. (2)). In the case that we use an equality instead of a similarity the above class size corresponds to the number of the patterns in the class: $s(C) = |C|$. The class size can be viewed also to be proportional to frequency of the event type described by the class.

4 WEAK-APRIORI Property

Using the above notation, the APRIORI property cited in the introduction and used as a basis for the algorithm can be expressed as follows:

$$s(C^{(m-1)}) \geq s(C^{(m)})$$

where $C^{(m-1)}$ is a subclass of $C^{(m)}$. In the case of similarity instead of equality the APRIORI property does not have to be always valid: it can be violated by m-patterns which have $(m-1)$-subpatterns that are not so similar as the patterns itself. A reason may be a pair of strongly dissimilar events in the subpatterns whose impact on the total similarity between the patterns loses strength with increasing pattern length. This property can hold for all pairs of patterns in a class and thus leads to a smaller size of its subclasses. In this case the APRIORI property will not hold.

A remedy is a different version of the APRIORI property which requires a weaker bound on the size of the subclass:

Definition 6 WEAK-APRIORI-*Property: For all subclasses $C^{(m-1)}$ of a pattern class $C^{(m)}$ holds:*

$$s(C^{(m-1)}) \geq g(m)s(C^{(m)}) - f(m)|C^{(m-1)}|$$

where $g(m)$ and $f(m)$ are positive functions of the class order m.

The functions $g(m)$ and $f(m)$ serve as a correction for the smaller size of a subclass. The concrete form of $f(m)$ and $g(m)$ depends on the similarity measure and an instantiation is given in eq. (3) for the similarity defined there. Provided the property in Def. 6 holds for a similarity measure we can see that Algorithm 1 correctly recognizes all l-classes with size at least s_{th} if we use in line 5 in the i^{th} loop for the size of a class C the following dynamic threshold ($i \in [2, l]$):

$$s_{th}^{(i)}(|C|) = s_{th} \prod_{k=i+1}^{m} g(k) - |C| \sum_{k=i+1}^{m} f(k) \prod_{j=i+1}^{k-1} g(j)$$

This threshold results from the WEAK-APRIORI property by propagating the class size decrease from step l till step i. It guarantees that we will not miss in earlier loops of the algorithm any subclasses which can be used to construct and consequently detect all classes of order l with size at least s_{th}. A proof of this fact can be found in [13].

5 Pattern Clustering

The objective of line 4 in Algorithm (1) is to classify the generated patterns into clusters which correspond to the same activity type. Precisely, this clustering step must result in large coherent classes which are also clearly distinguishable from each other.

We propose an entropy-based agglomerative hierarchical clustering method ([9], [4]). Starting with classes containing only one pattern, in each successive step we merge those two classes C_i and C_j whose merge leads to the highest increase of a utility function $U(C_i, C_j)$. This function is based on the weighted entropy $H_w(C)$ of a class C which is defined as the product of the class size $s(C)$ and the class entropy $H(C)$. The entropy $H(C)$ is defined by interpreting a class as a random variable with values equal to the patterns and probabilities $P_C(\cdot)$ of those values proportional to the class similarities:

$$P_C(p) \stackrel{\text{def}}{=} \frac{\text{sim}(p, C)}{\sum_{p' \in C} \text{sim}(p', C)} = \frac{\text{sim}(p, C)}{s(C)}$$

where $p \in C$. From the above definitions follows:

$$H_w(C) \stackrel{\text{def}}{=} s(C)H(C) = \sum_{p \in C} \text{sim}(p, C) \log \left(\frac{s(C)}{\text{sim}(p, C)} \right)$$

A class of high quality is characterized by a large value of the weighted entropy: large class size indicates a lot of mutually similar patterns in the class and large class entropy indicates good coherence and lack of outliers.

During the clustering we merge classes if this step leads to an increase of the weighted entropy of the new class compared with the old classes:

$$U(C_i, C_j) \stackrel{\text{def}}{=} H_w(C_i \cup C_j) - H_w(C_i) - H_w(C_j) > 0$$

We iterate the above merge step until no further increase can be achieved. In this case we have hopefully all patterns describing the same activity type in one class.

6 Similarity Measure

In this section we describe informally a similarity measure between video events. As an application we use videos recorded at a parking lot divided into zones (see fig. (1(b))) in which *vehicles* and *persons* are tracked. The system recognizes the events 'an object being in a zone' and 'an object close to another object'. These events can be described completely by a tuple of attributes: event name, object types, zone name and start/end time.

A similarity measure compares event patterns using the attributes of the primitive events. We distinguish between

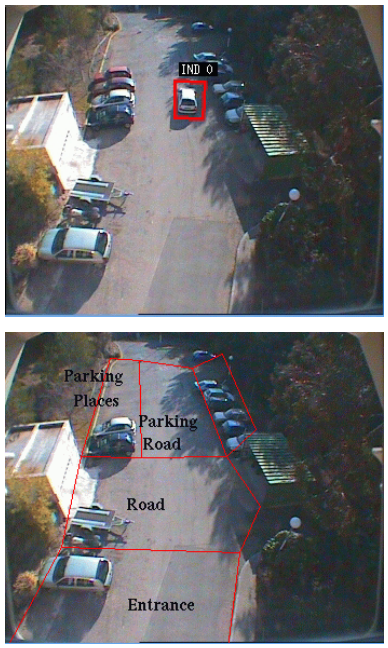


Figure 1. (a) Visualization of the event 'vehicle on the parking road'. (b) Manually defined zones in the scene.

symbolic and *numeric* attributes. In the above domain examples for the first attribute types are event name, object type, and zone name; examples for the second type are event duration, event start/end time. The former are in most of the cases unordered and can be compared only for equality while the latter must be treated with a soft comparison function:

$$C_{symb}(x, y) \stackrel{\text{def}}{=} \begin{cases} 1, & x = y \\ 0, & \text{otherwise} \end{cases}$$

$$C_{num}(x, y) \stackrel{\text{def}}{=} e^{-\frac{(x-y)^2}{\alpha xy}}$$

where $x, y \in \mathbb{R}$ and $\alpha \in \mathbb{R}^+$ is a parameter. The usage of the denominator xy makes the function more sensitive to differences between small values. This corresponds to the assumption that attributes with small values are more susceptible to changes.

Another taxonomy of the attributes is based on their usage. Some attributes can be compared directly like event names, object types, zone name, and event duration. In other cases we must evaluate an attribute in relation with attributes of other events: comparing event start/end times directly does not make sense but only in the context of another event in order to express the temporal relation between them. Based on this distinction between attributes we define

the **attribute-structure similarity**:

$$\mathrm{sim}_{\text{a-s}}(p_i, p_j) \stackrel{\text{def}}{=} w_a \,\mathrm{attr}(p_i, p_j) + w_s \,\mathrm{struct}(p_i, p_j)$$

with $w_a, w_s \geq 0, w_a + w_s = 1$, which compares not only the properties of the primitive events in a separate manner through its **attribute similarity** $\mathrm{attr}(p_i, p_j)$ but also compares the structures of the patterns expressed in terms of the temporal relations between the events in a pattern formulated as **structure similarity** $\mathrm{struct}(p_i, p_j)$. Using the above two notions we combine direct comparison with principles of analogy reasoning ([5]).

The above components of the similarity must be defined manually for each domain. Hereby, the attributes of a primitive event $e_k^{(i)}$ of a m-pattern $p_i, k = 1 \ldots m$, are compared with the attributes of exactly one event $e_k^{(j)}$ of the other m-pattern p_j using an appropriate compare function: symbolic attributes are compared only about equality; for numeric attributes x and y we use $e^{-\frac{(x-y)^2}{\alpha x y}}$. The similarity between patterns is the average of the similarities between all primitive events:

$$\mathrm{attr}(e_k^{(i)}, e_k^{(j)}) \stackrel{\text{def}}{=} \prod_{a^{(i)} \in D(e_k^{(i)})} C(a^{(i)}, c_{i,j}(a^{(i)}))$$

$$\mathrm{attr}(p_i, p_j) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{k=1}^{m} \mathrm{attr}(e_k^{(i)}, e_k^{(j)})$$

where $c_{i,j}(a) \in D(e_k^{(j)})$ is the corresponding attribute in $e_k^{(j)}$ to an attribute a from $e_k^{(i)}$. The **structure similarity** compares the temporal relations of the primitive events. The temporal relation between $e_k^{(i)}$ and $e_l^{(i)}$ from p_i can be compared with the temporal relation between $e_k^{(j)}$ and $e_l^{(j)}$ from p_j as follows:

$$\mathrm{struct}(e_k^{(i)}, e_l^{(i)}, e_k^{(j)}, e_l^{(j)}) \stackrel{\text{def}}{=} C_{num}(\mathrm{dist}(e_k^{(i)}, e_l^{(i)}), \mathrm{dist}(e_k^{(j)}, e_l^{(j)}))$$

where $\mathrm{dist}(\cdot, \cdot)$ compares the temporal distance between events:

$$\mathrm{dist}(e_i, e_j) \stackrel{\text{def}}{=} \begin{cases} d(e_i, e_j), & |d(e_i, e_j)| \leq |d(e_j, e_i)| \\ d(e_j, e_i), & \text{otherwise} \end{cases}$$

$d(e_i, e_j) = b(e_i) - e(e_j)$ with $b(e)$ and $e(e)$ start und end time of an event e. Finally:

$$\mathrm{struct}(p_i, p_j) \stackrel{\text{def}}{=} \frac{1}{m(m-1)} \sum_{k,l=1, k \neq l}^{m} \mathrm{struct}(e_k^{(i)}, e_l^{(i)}, e_k^{(j)}, e_l^{(j)})$$

configuration		sample pattern 1		sample pattern 2	
pert.	noise	length	rank	length	rank
5	0%	5	1	6	2
	25%	5	1	6	1
	50%	5	1	6	2
10	0%	5	1	6	2
	25%	5	2	5	1
	50%	5	1	6	1

Table 1. Results with synthetic data. For each configuaration of perturbation variance, noise portion, and sample pattern the rank and the length of the most meaningful detected pattern are displayed.

It can be shown that the attribute-similarity similarity satisfies the WEAK-APRIORI property ([13]):

$$s(C^{(m-1)}) \geq \underbrace{\left(1 - \frac{1}{m-1}\right)}_{g(m)} s(C^{(m)}) - \underbrace{\left(\frac{w_a}{m-1} + \frac{2w_s}{m-2}\right)}_{f(m)} |C^{(m-1)}| \quad (3)$$

With increasing pattern length m the bound on the right side of the above inequality converges towards $s(C^{(m)})$ because $g(m) \stackrel{m \to \infty}{\longrightarrow} 1$ and $f(m) \stackrel{m \to \infty}{\longrightarrow} 0$. Hence, the violation of the initial APRIORI property decreases with increasing pattern length and so the computational effectiveness of the initial algorithm is preserved.

7 Evaluation

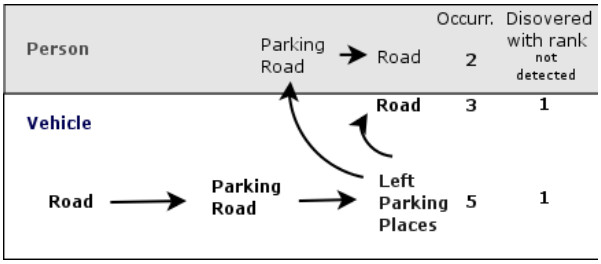
We test our approach on two types of sets containing frequent composite events: synthetic data and data from the parking lot monitoring domain. In both cases we describe the data manually and compare the most frequent patterns found by the algorithm with this description. We assess (i) which subpatterns of the expected event patterns were recovered and (ii) what is their frequency compared with the frequency of the other detected patterns. The latter aspect is quantified in form of a rank: a pattern has rank k if it is the k^{th} most frequent.

The synthetic data was generated from two manually created sample patterns of length 6. For each sample pattern 6 test sets were created as follows. 5 copies of each of these patterns were perturbed and randomly positioned in a time interval of 15000 time frames. Precisely, the start/end times were perturbed with a gaussian noise with variance equal to 5 and 10 time frames and mean 0 and thus resulting in

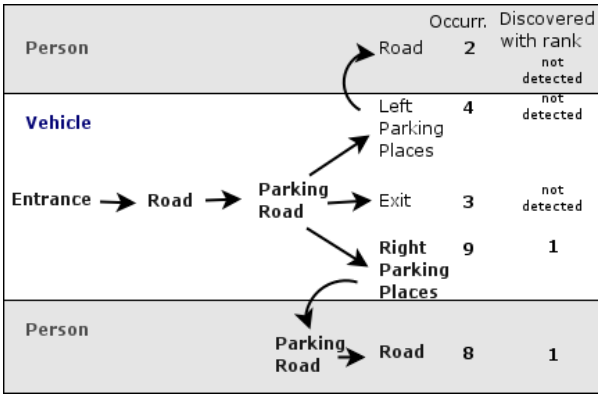
two sets for each sample pattern. Additionally, noisy events were added to each set whose portion of all events equals to 0%, 25% or 50%. The resulting sets and the results of the experiments for each set are displayed in table (1) and show that in all test runs at least a 5-subpattern of the optimal 6-pattern was recovered. This pattern was in 65% of the cases the most frequent and in the remaining cases the second frequent. In the case when the expected pattern was the second frequent, the most frequent pattern was caused by events which coincidentally form patterns. The performance of the algorithm was stable even in the worst case of high perturbation and 50% noise.

In order to evaluate the technique in a real situation we apply it in the parking lot domain. We process appr. 4 hours video from two days resulting in appr. 200 hundred primitive events representing appr. 20 composite events divided into two sets, one set for each day. These sets are presented in fig. (2) together with the results. In both cases the most frequent complex events were detected and they had in both cases rank 1. These events correspond to the manoeuvre parking and thus the most natural activity in the domain was detected.

The reasons for not obtaining a pattern of full length are strong perturbations in some cases and imperfect tracking which splits sometimes one primitive event into several due to lost objects. The computational cost reduces rapidly with each iteration of the APRIORI algorithm: in each step beyond the 3^{rd} one less than 1% of all possible patterns were taken into account. This shows the effectiveness of the WEAK-APRIORI property in the case of the attribute-similarity measure: the bound becomes more restrictive with increasing pattern length for which the number of possible patterns increases. On a machine with a 3.4 GHz Intel Pentium® CPU the algorithm needed between 30 and 90 minutes for each run.



(a) BOREL_PARKING_11_03



(b) BOREL_PARKING_21_03

Figure 2. Manually created description of the data and results. Each flow displays a sequence of primitive events of 'vehicle or person in an zone' with the zone name and object type given. The occurrence refers to data descriptions. The discovered complex events are marked green with their rank to the right.

8 Conclusion and Future Work

In this paper we present an approach for detecting frequent composite events using the APRIORI algorithm from the data mining field. We were able to adapt this algorithm to handle uncertainty without losing its computational attractiveness. It discovers clear composite events and structures them hierarchically. The proposed method is built as a general framework for which context knowledge in form of a similarity measure and a generic library of primitive events must be specified.

In the future we would like to investigate other similarity measures based, for example, on probabilities. In a similarity we can incorporate not only uncertainty of the temporal attributes but also of the remaining attributes such as labels, for examples. Another topic is the analysis of the detected frequent patterns in order to create a compact and expressive model of the whole data. A different topic is to improve the performance of the method. This can be achieved through better implementation but also through integrating the operation from the line 3 in Algorithm (1) as another merge step in the clustering from line 4: we can create longer patterns by combining classes whose patterns have large number of overlapping states. In this way the whole algorithm can be represented as a clustering.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant: *Mining Sequential Patterns.* Proc. of the 11^{th} International Conference on Data Engineering, 1995, pp. 3 - 14.

[2] Alberto Avanzi, Francois Bremond, Christophe Tornieri and Monique Thonnat: *Design and Assessment of an Intelligent Activity Monitoring*

Platform. EURASIP Journal on Applied Signal Processing, 2005 (in press).

[3] Matthew Brand and Vera Kettnaker: *Discovery and Segmentation of Activities in Video.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 22:8 (2000), pp. 844 - 851.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork: *Pattern Classification.* John Wiley & Sons, Inc., New York, 2001.

[5] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner: *The Structure-Mapping Engine: Algorithm and Examples.* Artificial Intelligence, 41, (1989), pp. 1 - 62.

[6] Aphrodite Galata, Anthony Cohn, Derek Magee, and David Hogg: *Modeling Interaction Using Learnt Qualitative Spatio-Temoral Relations and Variable Length Markov Models.* Proc. of the 15^{th} European Conference on Artificial Intelligence, 2002, pp. 741 - 745

[7] Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank: *A Survey on Visual Surveillance of Object Motion and Behaviors.* IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 34:3 (2004), pp. 334 - 352.

[8] Weiming Hu, Dan Xie, Tieniu Tan, Liang Wang, and Steve Maybank: *Learning Activity Patterns Using Self-Organazing Neural Networks.* IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics, 34:3 (2004), pp. 1618 - 1626.

[9] Tao Li, Sheng Ma, Mitsunori Ogihara: *Entropy-Based Criterion in Categorical Clustering.* Proc. of The 21^{st} International Conference on Machine Learning, 2004, p. 68

[10] D. R. Magee, C. J. Needham, P. Santos, A. G. Cohn, D. C. Hogg: *Autonomous learning for a cognitive agent using continuous models and inductive logic programming from audio-visual input.* Anchoring Symbols to Sensor Data, 2004 AAAI Workshop, 2004, pp. 17-24.

[11] Xingzhi Sun, Maria E. Orlowska, and Xue Li: *Introducing Uncertainty into Pattern Discovery in Temporal Event Sequences.* Proc. of the 3^{th} IEEE International Conference on Data Mining, 2003, pp. 299 - 306.

[12] Van-Thinh Vu, François Brémond, and Monique Thonnat: *Temporal Constraints for Video Interpretation.* Proc. of the 15^{th} European Conference on Artificial Intelligence, 2002.

[13] Alexander Toshev: *Unsupervised Learning of Scenario Models in the Context of Video Surveillance.* Diploma thesis, Institut für Algorithmen und Kognitive Systeme, Fakultät für Informatik, Universität Karlsruhe (TH), Karlsruhe, Germany, July 2005.

[14] Jiong Yang, Wei Wang, Philip S. Yu, and Jiawei Han: *Mining Long Sequential Patterns in a Noisy Environment.* Proc of the 2002 ACM SIGMOD International Conference on Management of Data, 2002, pp. 406 - 417.

[15] Lexing Xie, Shih-Fu Chang, Ajay Divakaran, and Huifang Sun: *Unsupervised Mining of Statistical Temoral Structures in Video.* Video Mining; A. Rosenfeld, D. Doermann, D. Dementhon (Eds.), 2003, pp. 279 - 307.

[16] Michael Walter, Alexandra Psarrou, and Shaogang Gong: *Data Driven Model Acquisition using Minimum Description Length.* Proc. of the British Machine Vision Conference, 2001, pp. 673 - 683.