Using decision trees to recognize visual events

Cedric Simon
Communication and Remote Sensing Lab
UCL, Louvain-La-Neuve, Belgium
cedric.simon@uclouvain.be

Jerome Meessen
Multitel ASBL
Mons, Belgium
jerome.meessen@multitel.be

Christophe De Vleeschouwer
Communication and Remote Sensing Lab
UCL, Louvain-La-Neuve, Belgium
devleeschouwer@uclouvain.be

ABSTRACT

This paper presents a classifier-based approach to recognize events in video surveillance sequences. The aim of this work is to propose a generic event recognition system that can be used without relying on a long-term tracking procedure. It is composed of three stages. The first one aims at defining and building a set of relevant features from the foreground objects. Second, a clustering tree-based method is used to handle the features and aggregate them locally in a set of coarse to fine activity patterns. Finally, events are modeled as a sequence of structured patterns with an ensemble of randomized trees. In particular, we want this classifier to discover the temporal and causal correlations between the most discriminative patterns. Our system is tested on simulated events and in a real world context with the CAVIAR video sequences dataset. Preliminary results demonstrate the effectiveness of the proposed framework for event recognition in automated visual surveillance applications. We also prove that more flexible algorithms (i.e. deterministic classifiers) rather than probabilistic graph models are conceivable for video events analysis.

Keywords

Randomized Decision Trees, Automated Visual Surveillance System, Activity Recognition

1. INTRODUCTION

Challenges The growing number of cameras in public and private areas increases the interest of the image processing community in automated visual surveillance system. Nonetheless, there is still a broad gap between what automated systems offer and the actual needs of the industry [8].

Visual surveillance events have some specific characteristics that makes them especially challenging to recognize:

- They are difficult to define, as they can be subjective and semantically complex. Therefore, a strong interaction of the expert with the automated system (e.g such as expert systems) is often necessary to model elaborated events.

- Sophisticated events are often variable in length, which means that they are non stationary over the time dimension. Therefore, the model of each event should be scale invariant.

- Visual surveillance events are divided into two subgroups in the literature: events that depend on the location and/or the actions of each individual objects, and events that depend on the relative displacement and interaction between the objects, often independently of their absolute position. Those groups are often analyzed separately while occurring most of the time in the same context.

- The events of interest occurs very rarely. Therefore, using a supervised learning technique can be knotty because of the few available training samples.

- Most of the time, the angle of view proposed by a surveillance camera is wide. As a result, the scene contains passers-by who do not take part in the events.

Most of the event recognition systems start with tracking and occlusion reasoning, then use some a priori information to model the scene, analyze the objects actions, and finally conclude with an event recognition algorithm [7]. Therefore, the efficiency of this latter algorithm mainly relies on the video processing algorithms up stream.

Long-term tracking, for instance, suffer from a lack of robustness in most realistic video surveillance scenarios. Indeed, objects are difficult to track over a long period of time in crowded scenes, because frequent overlapping motion patterns often result in discontinuous objects trajectories and inconsistent labeling. Those mistakes can be also due to surveillance camera contexts, and are usually caused by several artifacts such as variable luminosity, low resolution or non-stationary objects.

Related works Once the trajectories are computed, generative models like Hidden Markov Models (HMM) [13, 14] or Bayesian Network (BN)[16] are often used to represent the events. A recent review [6] describes the field almost entirely in terms of these methods. An HMM is essentially a quantification of a system's configuration space into a small number of states, together with probabilities of transitions between the states.

AREA '08, Vancouver, British Columbia, Canada

Though generative sequential models are reliable for simple events whose structure can be easily learned from training data, deterministic classifier-based inference seems more appropriate when dealing with more complex events [7]. Also, they are better in handling multi-dimensional and multi-class dataset. For instance, in [15], Perez compared a set of classifiers for human activity recognition. It showed that HMM gave a slightly better performance when using a few features, but it was not well adapted to the use of multiple features, as most of the sequential classifier. Also, some of the interesting activities to detect were often hardly recognizable due to the ill definition of the labels in the ground truth.

More recently, attempts have been made at identifying human interactions [14, 20]. In [14], the authors compare two state-based statistical learning architectures (HMMs and Coupled-HMMs) for modeling behaviors and interactions. Given the limited amount of training data available, they propose to train the learner with a synthetic agent training system that allows to create flexible and interpretable prior behavior models.

Instead of trying to recognize short actions, the framework presented by Xiang in [20] proposes to model more elaborated events. From the pixel changes distribution, he detects and classifies object-independent actions at each frame by using a clustering method. Then he uses the Bayesian Information Criterion in order to find a specific set of actions. Finally, Dynamic Probabilistic Networks are formulated to model an activity within a video sequence with the temporal and causal correlations among discrete action.

In a work on multivariate time-series classification (which is identical to trajectory features classification) , Waleed propose to create a set of generic metafeatures from the multivariate time-series data [10], where the temporal properties are converted into propositional attributes. Then, he compares the classification performance of several deterministic and generative learners (HMM, decisions trees, ensemble methods...). If it does not give a clear conclusion about the best classifier, it shows that they can all achieve high accuracy by using the appropriate metafeatures.

Our contribution Because of the peculiarities of the visual events, the proposed approaches tend to be very specific to one type of events, or to analyze only short actions in the video. The system we propose aims at recognizing a wide set of possibly sophisticated events. It is thus capable of modeling individual behavior as well as the interactions of people.

No long-term tracking procedure, intermediate reasoning (e.g. occlusions) nor a priori information are needed, which makes our system greatly flexible and generic. This is achievable by first creating a set of patterns describing people's activities in the foreground. Then, a deterministic classifier (i.e. in our case the ensemble of randomized trees) models the events by encoding the space and time arrangements of the patterns in the video sequences.

Outline This paper begins by presenting our proposed framework. Then, the preprocessing steps, used to extract the foreground objects features, are detailed. Our learning strategy is composed of two stages. They are explained in Section 4 and 5. Experiments are proposed in Section 6. We first analyze our system on simulated events. Then we try it on a real visual surveillance environment. Next, from our experimental results, we discuss about the benefits and drawbacks of our architecture for visual event recognition. Finally we conclude and propose some future works.

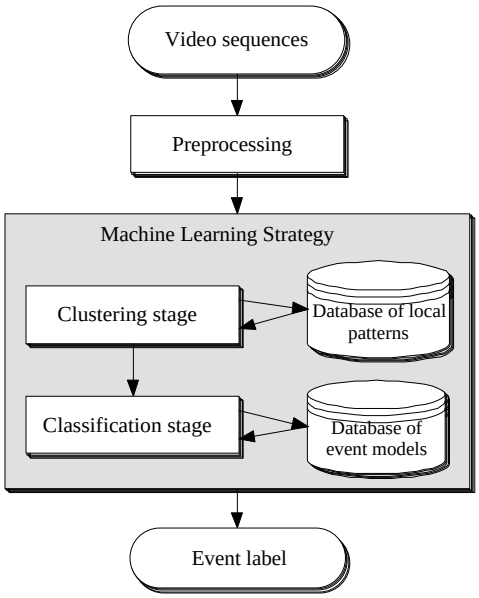

Figure 1: Framework

2. OVERALL ARCHITECTURE

We define a visual surveillance event as a combination of coarse human actions occurring at specific times and places in a monitored scene. In order to recognize those events, the proposed framework relies on three main stages (Figure 1).

In a first stage, a set of preprocessing methods are exploited to detect the foreground objects and extract useful information from them. This step is essential in automated visual surveillance system, for which the valuable information is generally composed of the moving objects (people, vehicles...). Once those objects (i.e. the blobs) are extracted, the appropriate set of blob's features are provided to a machine learning system. The events of interest can then be recognized by reasoning about how the objects look like, where and how they move, and their possible interaction.

In a second stage, we describe the blobs and their behavior in the sequence, in order to infer the events. Unsupervised learning is used to cluster the features in a set of patterns. A large amount of coarse to fine patterns (standing for the actions) are created using the clustering tree methodology [3]. In this process, each node of a clustering tree identifies a specific pattern characterizing a blob or a group of blobs. Thus, each node is labeled, and those labels are associated to the appropriate time-steps in the video sequences, according to the type of blobs existing in each time-step.

In the third stage, the events are classified, based on the temporal distribution of the patterns in the sequences. The main idea is thus to discriminate the events by investigating the temporal relationships between the patterns, for example by asking if there is a pattern X 'before' a pattern Y. Those relationships are specified by coarse constraints on the temporal arrangement of the patterns, and do not involve absolute temporal location or scale constraints. The ensemble of randomized trees methodology [9] is used in order to

find the discriminant relationships.

This machine learning framework is inspired on Geman works in [2]. He suggests to recognize patterns (in images) also by using a set of decision trees. In his work, each node of the classification trees choose a test that describe a spatial arrangement between local features. Like us, those local features are labels for the tree nodes, where each node is a successive partition of small sub-images. We propose in this paper to extend this concept for events recognition in video surveillance sequences. In our case, the challenge lies in identifying key objects and actions that are indicative of events of interest. If we use each blob's features independently, the risk is to use features that belong to uninteresting objects. By using higher level descriptors possibly representing the actions in the scene, this risk becomes greatly reduced.

3. BLOBS FEATURES EXTRACTION

Our machine learning framework takes as input the blobs features. In this work, a *blob* is defined as a group of local pixels belonging to the foreground, and linked spatially and temporally. Therefore, each blob is a spatio-temporal volume where the temporal window size can be set empirically. In section 6, we compute the blobs features based on 12 frames, with a 6 frames overlap for video sequences recorded at 25 frames per second. The idea of using spatio-temporal blobs for building more robust and efficient features is not new [11, 21]. In our case, it allows to integrate at the blob's level its motion features. Another alternatives for motion features construction (without tracking procedures) would be to use the Motion History Image [4] or the Pixel Change history [20]. Those motion features are often combined with the shape or the position of the blobs to describe coarse actions happening in the scene, which is our goal in section 4.

Blobs segmentation Our framework is based on a real-time statistical segmentation algorithm using a mixture of Gaussians modeling for the background luminance of each pixel [12]. The main advantage is that it automatically supports backgrounds having multiple states like blinking lights, grass and trees moving in the wind, acquisition noise etc. Furthermore, the background model update is done in an unsupervised way when the scene conditions are changing.

From the images derived from background subtraction, each frame is binarized and the morphological operations *shrink* and *expand* are applied to reduce the noise. Then, a fast connected-component operator is used to differentiate each blob, spatially but also temporally (on the 12 frames). Small regions are then eliminated and big regions are considered as the relevant objects, i.e. the blobs.

Features From the latter methods, we can define a set of features that, when combined together, may distinguish the different actions included in the events of interest. We consider, intuitively, three groups of features needed to describe properly the events.

The first group of features code the average size $\overline{s}$, the average position $\overline{x}$ and $\overline{y}$, and speed $\overline{v}$ of each blob. $\overline{s}$ refer to the number of pixels included in a spatio-temporal blob. $\overline{x}$ and $\overline{y}$ are the 2D centroid position of a blob being projected on the x-y coordinate system. The speed $\overline{v}$ is the norm of the velocity vector $\overrightarrow{v}$ obtained through differentiation of the average position estimate of the first 6 frames and the last 6 frames:

$$\overline{v} = \sqrt{(x_{fr_{7-12}} - x_{fr_{1-6}})^2 + (y_{fr_{7-12}} - y_{fr_{1-6}})^2} \qquad (1)$$

Since the blobs are computed over 12 frames, the temporal derivative is averaged and contains less noise than instantaneous velocity.

A second group of features describes the interaction between the blobs. We use the distance D^{2b} between the blobs and its difference of speed $\Delta\overrightarrow{v^{2b}}$ that are computed easily from the previous blob's features. We also compute the ratio R^{2b}_{dir} between the norm of the average velocity of the 2 blobs, and the average speed of the 2 blobs. It gives a good description of the relative direction between the two blobs:

$$R^{2b}_{dir} = \frac{||\overrightarrow{v^{2b}}||}{\overline{v^{2b}}} = \frac{||\overrightarrow{v_1} + \overrightarrow{v_2}||}{||\overrightarrow{v_1}|| + ||\overrightarrow{v_2}||} \qquad (2)$$

R^{2b}_{dir} approaches to 1 when the blobs have the same direction and tends to 0 when the directions disagree. Because only people that are close to each other are more likely to interact, we take into consideration only pairs of blobs having a relatively short distance between them (which can also be chosen empirically).

A third group of features, finally, is based on the motion inside each blobs. With our definition of a blob, one blob can represent one person, but also multiple objects being close or crossing each other. A set of blobs features are then created to discriminate what each blob characterizes and/or the way it moves. In [17], Ribeiro proposes to use one set of features to code the instantaneous position and velocity of an object, and a second set of features to represent the instantaneous pixel motion inside each tracked object. We proceed in a similar way, by building a set of features characterizing the intra-blobs motion. The proposed features are the variation of the size $\Delta\overline{s}$, speed $\Delta\overline{v}$ and direction $\Delta\overline{d}$ inside each blobs. $\Delta\overline{s}$ and $\Delta\overline{v}$ are obtained through differentiation of the average size and speed estimate of the first 6 frames and the last 6 frames, while for $\Delta\overline{d}$, we apply equation 2 by using the speed and velocity vectors given by the first 6 frames and the last 6 frames of the blobs.

The computed features are finally stored in a table that can be used as input by our machine learning algorithm. Each blob is thus represented by a 10-dimensional features vector:

$$B = [\overline{x}, \overline{y}, \overline{s}, \overline{v}, D^{2b}, \Delta v^{2b}, R^{2b}_{dir}, \Delta\overline{s}, \Delta\overline{v}, \Delta\overline{d}] \qquad (3)$$

4. CONSTRUCTION OF LOCAL PATTERNS

The previous section computes an ensemble of features for all foreground objects in the sequences. Individually, each single feature carries little information about any elaborated event. Discriminative power is expected to arise from the combination among the features, both spatially and temporally. The idea of the following section is to cluster the foreground objects (i.e. the spatio-temporal blobs) according to their features and their (spatial) neighbors. Those clusters are local patterns characterizing coarse actions at specific instants in the video sequences, and are represented by a specific location, shape, direction and motion of the blobs and pair of blobs.

From the blobs features, we adopt an information-theoretic approach and cluster similar blobs by using the classical unpruned decision tree methodology. At each node of the tree,

a question is selected by choosing one attribute, i.e by picking one blobs feature and one threshold. Each node of the tree can be seen as a weak classifier that organizes each blobs of the input sequences into two branches, based on the binary answer of a question.

Given a set of blobs in a tree node, the growing process selects the question in a vast database so as to provide a significant information gain, i.e. so as to reduce the impurity of the output variable within the local learning subset. In our case, we do not use the events classes, making this stage completely unsupervised. Instead, we select the attribute and cut-point at node n such that the intra-node distance (i.e. the Euclidean distance between the attribute values of the examples belonging to the same node) is as small as possible and the inter-node distance is as large as possible.

The family of blobs in the video sequences is thus recursively partitioned with binary splits. A stop splitting criteria can be applied in order to prevent small and useless clusters. In our experiment section, we stopped splitting when the number of blobs in a node reached less than one percent of the total number of nodes in the training video sequences.

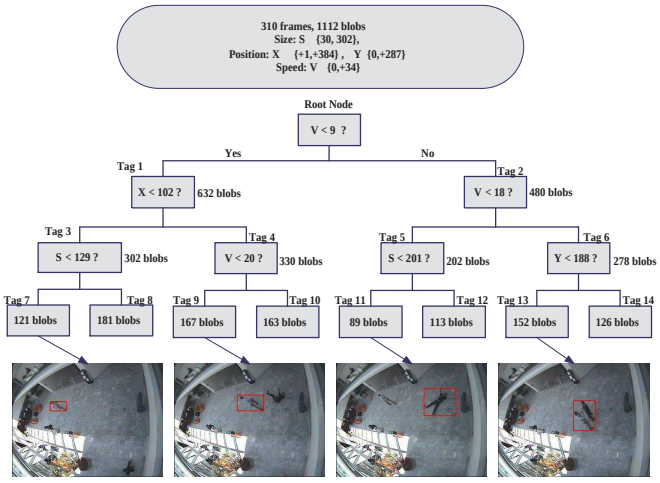

Figure 2: Depth three clustering tree with a video sequence from the CAVIAR dataset. A tag is assigned to each node, corresponding to a local pattern in the sequence. The temporal and causal correlations among the patterns are further analyzed by a classifier

The three groups of blobs features are considered separately to cluster the blobs, and we thus build three pools of clustering trees:

- In the first pool, attributes are based on the first set of features, characterizing each blob individually $(\overline{x}, \overline{y}, \overline{s}, \overline{v})$
- In the second pool, attributes are based on pair of blobs features $(D^{2b}, \Delta v^{2b}, R^{2b}_{dir})$.
- In the third pool, attributes are based on the intra-blob motion's features $(\Delta\overline{s}, \Delta\overline{v}, \Delta\overline{d})$.

An example of clustering tree for the first pool is proposed in Figure 2. It shows the first three levels along with some instances from the CAVIAR video sequences. Once the trees are built, each node represents a *pattern* and is tagged with a number (except for the root node), so that each time-step of the sequences can be labeled with the appropriate patterns. Therefore, those patterns are a data-driven combination of the blobs features which allows us to reduce the dimensionality of the data while keeping most of the information. The number of clustering trees should be set empirically, knowing that more trees means more diversity for the local patterns but also more redundancy and confusion for the classifier.

5. EVENTS CLASSIFICATION

We defined and created a list of potentially relevant patterns to represent the video sequences. We will now proceed to select which patterns prove most useful for event recognition. The idea of the following section is to design a classification system which is able to learn prior knowledge about the events of interest. We propose to use an ensemble of randomized trees (ERT) to model the video sequences according to the temporal and causal relationships between the local patterns.

5.1 Modeling the spatio-temporal events with decision trees

Decision trees are used to classify the video sequences, based on the spatio-temporal relations of those patterns. This is done by defining the two output branches of a tree node based on the presence/absence of a specific arrangement of patterns in the video sequences. In order for these arrangements to be completely scale invariant (and better fit the notion of the visual surveillance events), coarse binary relations like "before", "after" and "at the same time" are used. An example of arrangement would then be: pattern X exists "before" pattern Y which exists "at the same time" as pattern Z.

In order to build the decision trees, we use the following methodology. Like in section 4, one question is chosen at each node that allows to minimize the impurity with regard to the classes within the local learning subsets. At the root node N_r one pattern P_x is chosen from the full set of patterns $\mathcal{P}_{N_r}$ available in the training sequences in N_r. The question Q_{N_r} at node N_r is *"Does P_x exist in the training sequences?"*. Those training samples for which $Q_{N_r} = 0$ are in the "no" child node N_{no} and we search again through $\mathcal{P}_{N_{no}}$ (i.e. the set of available patterns in N_{no}). Those samples for which $Q_{N_r} = 1$ are in the "yes" child node N_{yes} and we then put P_x among the used patterns $\mathcal{P}^{used}_{N_{yes}}$. Then, all the nodes N_j for which $\mathcal{P}^{used}_{N_j}$ is not empty can use the following questions:

$$\left\{ \exists P_i \star P_j ? \quad with \begin{cases} P_i \in \mathcal{P}_{N_j} \\ P_j \in \mathcal{P}^{used}_{N_j} \\ \star \in [before, during, after] \end{cases} \right. \tag{4}$$

Still, we keep the patterns P_i and P_j and the binary relation that minimize the impurity in the nodes downstream. In our case, this is done by using the normalized version of the Shannon information gain, which was proposed by Wehenkel [19]:

$$Score = \frac{2.I^A_C(S)}{H_A(S) + H_C(S)} \tag{5}$$

where S is the set of sequences in the node, $H_C(S)$ is the entropy of each class, $H_A(S)$ is the entropy of the chosen attribute and $I^A_C(S)$ its mutual information. $Score$ can varie

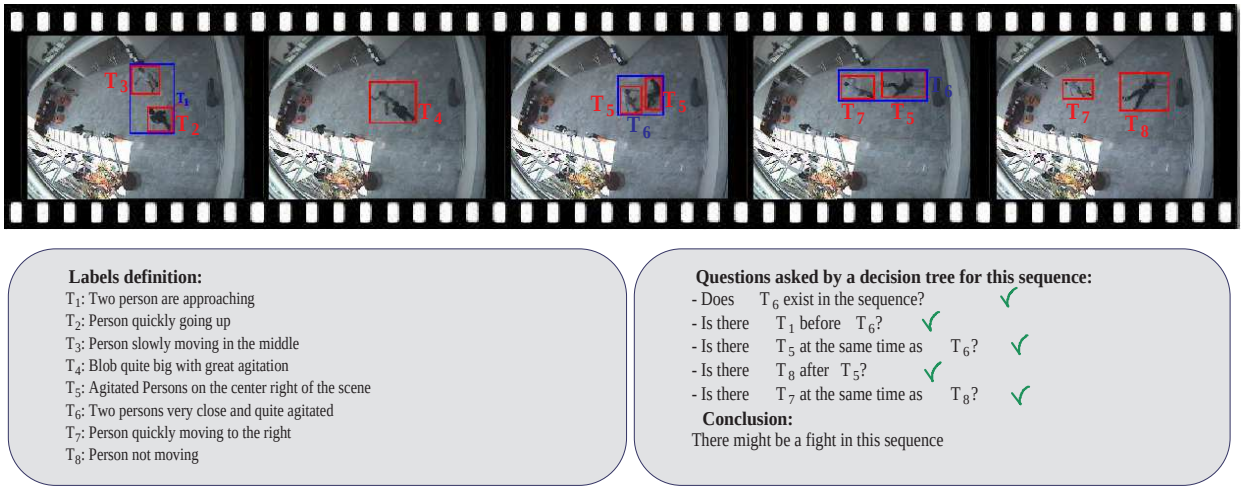

Figure 3: This example shows our system mechanism. It first creates a set of patterns ($T_1 ... T_8$) from the blobs features, and then evaluates their temporal relationships

between 0 and 1, 1 meaning that the classes are completely separated.

The criteria to stop splitting node N is when the training sequences in N belong to the same classes. This criteria is the most appropriate for applications where few training data are available. The example in figure 3 resumes the mechanism.

5.2 Ensemble of randomized trees

Similar to numerous previous works [2, 5, 9], randomization methods are considered to improve the tree decision accuracy. These methods explicitly randomize the tree growing algorithm, and perform multiple runs of the growing process, so as to produce an ensemble of more or less diversified tree models, whose predictions are aggregated by a simple majority vote. The purpose of multiple trees is to solve the approximation error and the estimation error problem at the same time.

In our case, we follow the Extra-Tree methodology [9], meaning that we select the attributes of the questions at each node partially at random, i.e. we choose the patterns and the binary relation maximizing the information gain among x randomly chosen patterns/relations. This contrasts with bagging approaches, which introduce randomization based on sub-sampling of the input training data. In our case, we do not want to grow the tree models based on a subset of the training samples because we are often dealing with scenarios for which the training set is already quite small. Further sub-sampling would thus dramatically impact the model accuracy.

6. EXPERIMENTAL EVALUATION

The described system was tested on two data sets. The first one consists of simulated video surveillance events, while the second is a real world context, occurring in the entrance lobby of a public company.

6.1 Simulated data

Building simulated events allow us to analyze the proposed event recognition system without being restricted by the number of training samples nor the quality of the preprocessing step. Moreover, it eases the study of robustness by adding in the scene in a considered way some noise and passers-by usually occurring in a real environment.

A Matlab code has been written to emulate four events of interest. Those events are: *pocket-picking, a fight, a bag left alone* and *someone staying in a forbidden zone*. With those events, we added 2 classes of 'non event': *passers-by* and *two persons meeting and discussing*.

The code together with a brief explanation of the assumptions underlying the random process generating the blobs trajectories are available in [1]. The output of the code gives a table containing the the usual blobs features and pair of blobs features (i.e $\overline{x}, \overline{y}, \overline{s}, \overline{v}$ and $D^{2b}, \Delta v^{2b}, R^{2b}_{dir}$) for each frame. We did not use in this first experiment the intra-blob motion's features, as it was more difficult to produce with a simulation.

In this experiment, we define the accuracy of the system as the quantity of correctly classified test samples over the total number of test samples. We use 100 test samples of each class for each trial. Figure 4 explains how our system's performance depends on:

- The number of *clustering trees* ($N_{ClustTr}$) envisioned to define the local patterns.
- The number of *classification trees* ($N_{ClassTr}$) used in the learning process.
- The number of training sequences per class used ($N_{TrainSeq}$).

It points out the fact that the number of clustering tree should be chosen with care according to the application, while the system's performance seems to increase with the number of trees and the number of training samples. Nevertheless, we notice a limit in the accuracy when $N_{ClassTr}$ and $N_{TrainSeq}$ reaches 50.

We now present the behavior of our system when some of its parts are perturbed. For this purpose, we use $N_{ClustTr} = 10$, $N_{ClassTr} = 30$ while $N_{TrainSeq}$ is set to 20 per class. We

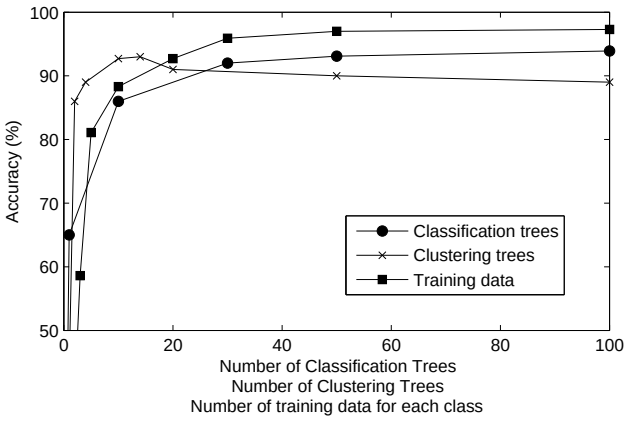

Figure 4: Behavior of our system with respect to the three main parameters. By default, those parameters are set up at $N_{ClassTr} = 30$, $N_{ClustTr} = 10$ and $N_{TrainSeq} = 20$

compare six different cases of alteration whose performances are depicted on Figure 5:

- **Case 1** : This is the initial set up of the system. The accuracy reaches 92.7%.
- **Case 2** : Only the blob's features from the first group $(\overline{x}, \overline{y}, \overline{s}, \overline{v})$ are used in the clustering process. Features coming from pairs of blob are not used here. The difference between the performance in case 1 (92.7%) and this case (85%) shows that using the appropriate features is essential.
- **Case 3** : Each of the 7 features are used independently to build the *clustering trees*. It means that the patterns are not a combination of intervals coming from different features but represent instead an interval of one specific feature. Figure 5 shows that it reduces the accuracy (86.3%).
- **Case 4** : In this case, the nodes of the classification trees are not using the temporal relation between the tags. Each node simply choose a pattern that is enough discriminant by itself. Results are reduced to 88.1%. Intuitively, the consequence of using the temporal arrangement between the local patterns should increase when the events get more complex, or can be seen as a succession of small actions.
- **Case 5** : Here, the clustering trees are not used. It means that the nodes of the randomized trees choose directly one blob's feature and one threshold, instead of the local patterns. Our system accuracy falls to 71%, which is the lowest result so far. Nevertheless, this case reaches an upper limit of 84% when adding 50 more training data for each classes.
- **Case 6** In this case, the ensemble of randomized trees are built without considering the event classes, in the unsupervised way. The event classes of the training samples are only used to assign the appropriate classes to the leaves of the trees. It means that the questions at each node are chosen completely at random (as long as it separates one sample from the training samples). The system remains relatively accurate with 88% of correct classification.

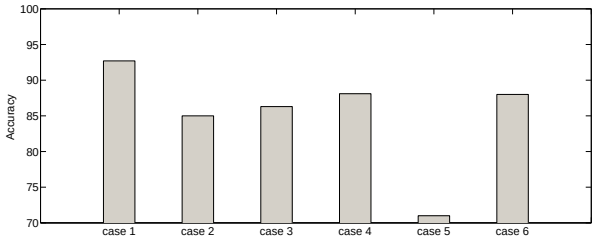

Figure 5: Accuracy of the system when the features construction or the classification process changes. Each case corresponds to one specific change in our framework.

Table 1: Accuracy of the system when the input data is disturbed. Column *TS and LS* corresponds to the accuracy when both the learning sample (LS) and the test samples (TS) are disturbed. Column *TS* corresponds to the accuracy when only the test samples are disturbed.

Input data	Overall Accuracy	
	TS and LS	TS
Only events	92.7 %	92.7 %
1 passer-by added	77.9 %	87.7 %
2 passers-by added	61.7 %	82.5 %
3 passers-by added	56.8 %	78.7 %
Random noise added	90.8 %	91.4 %
People split in multiple blobs	87 %	90.1 %

For these experiments, the dataset we used were built up from the trajectories of the only people performing the events of interest. We did not consider the people simply crossing the scene (i.e the passers-by), nor the possible noise and artifacts resulting from the preprocessing stage. This is what we analyze on Table 1. The first column gives the accuracy when both the learning samples (LS) and the test samples (TS) are disturbed. In the second column, only the testing sequences are disturbed.

The first row of the table gives the accuracy of the system when using a clean dataset. Then it shows its performance when one up to three passers-by are crossing the scene. Performances are decreasing quickly mainly when the algorithm is trained with the passers-by, while it gives better performance if we can manage to train the system without passers-by. Finally, the two last rows of Table 1 shows that our system is particularly robust with noise and artifacts coming from the preprocessing step. Random noise is simulated by adding in each frame of the sequence up to 3 blobs having a random position, size and speed. The last row of the table outlines a very common artifact in the blobs segmentation process. It gives at the output, instead of one blob characterizing one person, a set of one, two or three blobs very close to each other. Our system seems less affected by those disturbances rather than the passers-by.

Table 2: Events information

Events	Number of clips	Length
Walk	10	5s
Browse	10	7s to 15s
Left Bags	6	5s to 15s
Meet	6	5s to 10s
Fight	4	5s to 8s
Sign	15	8s

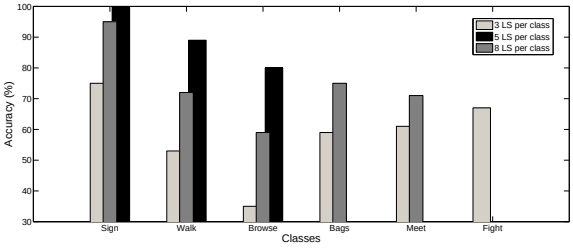

Figure 6: Accuracy of the system for each class, when the number of learning samples increases.

6.2 CAVIAR Data Set

Data Set The proposed video clips were captured by the CAVIAR project [1] with a wide angle camera lens in the entrance lobby of the INRIA Labs at Grenoble, France. The resolution is half-resolution PAL standard (384 x 288 pixels, 25 frames per second) and compressed using MPEG2. Six basic scenarios were acted out by the CAVIAR team members. Our goal is to use our framework to classify some parts of the available video sequences. Like in most real visual surveillance contexts, only a few training samples of the events of interest are available, which makes supervised learning especially challenging. The ground truth XML version of the sequences is not used in this work. The events we tried to recognize in this context are resumed in table 2. It corresponds to the most common events proposed by the CAVIAR project. We added the event *Sign* which corresponds to the person showing in body sign language the scene number in each original video clip. Those clips include several events. We removed some parts of the data where none of these events were happening, and we annotated the other parts with the appropriate classes. More detailed information about our experimental procedure is available in [1].

Preliminary results Each video clip is primarily segmented and filtered according to the proposed methods in section 3. The blobs features are computed over 12 frames with a 6 frames overlap, which gives us approximately 4 time-steps per second to represent each sequence. Only blobs larger than 50 pixels are considered. Size, appearance and speed of the people in the scene are influenced by the perspective given by the camera field of view. Unlike [17], we don't try to achieve a viewpoint invariance, which gives us some distorted features and noise.

We used a 10-fold cross validation to validate our results. We have 6 classes, and thus classifying them at random would give for each sequence 16.7 % probability to obtain the right class. The performance of the system are presented in Figure 6, when the number of training samples are increasing. It shows that our framework works reasonably well with a great variety of events, and the performance increases rapidly with the number of training sequences.

[1] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/

7. DISCUSSION

Reducing the dimensionality of the data is a very challenging and common problem in video analysis. From the huge set of available features in each data sample (i.e. a video sequence), it becomes barely impossible to find the appropriate combinations of specific features that can model each event. In order to reduce this input data, most of the recognition systems uses a priori information that makes strong assumptions to support the detection of the foreground objects and their trajectories. When using a Markov process, they also make strong assumptions while choosing their states definition. At the opposite, our system barely makes no assumption prior to the classification stage. We have a set of features characterizing the volumes of interest (the spatio-temporal blobs) that are organized and combined in a data-driven statistical way. Then the ERT procedure selects the most discriminant patterns and their temporal or causal relations. Therefore, our system accuracy mainly depends on the features extracted from the data and on the classifier, without being misled by previous hypothesis.

This difference makes the comparison with other event recognition systems difficult. Similar to us, Xiang proposes in [20] a framework that does not use any tracking algorithm. The main divergence is that its unsupervised clustering stage tries to define a set of consistent actions in terms of pixel changes distributions. Those actions define the states of a Markov model that is used to recognize events of interest. As a consequence, events that contain ambiguous actions can be hard to model, or trying to design a process for modeling multiple and dissimilar events seems difficult.

When dealing with blobs, one can question himself/herself about how noisy the extracted blobs can be and how sensitive is the algorithm to the input quality of the video. Four common noises generally affect the blob segmentation process, that tends to affect further tracking or learning algorithm. The first one is randomly distributed, and typically due to the low quality of the camera, the global illumination changes or other contextual elements. A second source of noise is caused by the fragmented blobs problem. Both have been experimented in section 6.1. Table 6.1 shows that our architecture is quite robust toward these types of noise. Furthermore, using spatio-temporal blobs helps to reduce these fragmented blobs by merging the blobs that are linked spatially and temporally. A third common source of error appears when the foreground objects is hardly differentiable from the background, or when it is too far or hidden from the camera. This kind of noise leads to excessive missed detections errors. By using coarse temporal arrangements of patterns (i.e *before, after*..), our learner is able to skip moment in the sequences where no valuable data are available, making our approach able to recognize events even in presence of partial observations. The fourth common error in the blobs analysis happens when two nearby people tend to get clustered as a single merged blob (for instance the pattern $T4$ in Figure 3.) We get over this merged detection error thanks to our blobs definition and representation. In our case, a blob does not have to represent one person specifically, but represents instead a volume where an action is happening. Then, a specific set of features $(\Delta\overline{s}, \Delta\overline{v}, \Delta\overline{d})$ is

created to understand what is happening inside this volume.

Finally, one could also questions the choice of a tree structure for classification regarding robustness to noise. Although a single decision tree often tends to overfit the data and thus lacks of robustness, ensemble methods applied to decision trees like ERT or Extra-trees [9] proved to be efficient and robust in many pattern recognition systems for image classification. It was also applied with some success on multivariate time series [18, 10]. Our main motivation in experimenting ERT on video event recognition is not only its computational efficiency which makes the system usable in real-time, but also its flexibility, which allows us to tend towards a greater genericity.

8. CONCLUSION AND PERSPECTIVES

This paper proposes a system that is able to recognize possibly sophisticated visual surveillance events. From the recorded video sequences, our system first extracts a set of features describing spatio-temporal blobs. These features are then clustered into patterns of features, which semantically refer to coarse actions in the scene. We finally use an ensemble of randomized trees to learn and model the events as a succession of those patterns. This architecture allows in a simple and generic way to deal with most of the challenges of visual event recognition.

Our experimental section proves the effectiveness of our framework. The results reported in section 6.2 are encouraging our conviction that it is possible to avoid the tracking procedure in order to gain in robustness. Nonetheless, one of the main drawbacks is given on table 6.1. It shows that the performance of the system decreases quite rapidly in crowded scenes. A solution was proposed by training the system with clean data, which means asking the user to indicate the objects actually taking part to the events. Further improvement could also analyze how to use histograms of similarity between blobs to enrich our model, along with features characterizing each frame more globally (i.e. the number of blobs in the frame, the frames number etc).

A great challenge with visual surveillance event is also the limited number of examples of those events for training the model. A major emphasis of our work, is on having a system that can already work with very few training samples. Future work will also look at adding more training data on line based on an active learning procedure.

9. REFERENCES

[1] http://www.tele.ucl.ac.be/∼csimon/trajectories.html.

[2] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997.

[3] H. Blockeel, L. D. Raedt, and J. Ramon. Top-down induction of clustering trees. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 55–63, San Francisco, CA, USA, 1998.

[4] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.

[5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] H. Buxton. Learning and understanding dynamic scene activity: a review. *Image and Vision Computing*, 21(1):125–136, 2003.

[7] H. Dee and S. Velastin. How close are we to solving the problem of automated visual surveillance? a review of real-world surveillance, scientific progress and evaluative mechanisms. In *Machine Vision and Applications, Special Issue on Video Surveillance Research in Industry and Academic Springer*, 2007.

[8] A. R. Dick and M. J. Brooks. Issues in automated visual surveillance. In *Proceeding of VIIth digital image computing: technique and applications*, pages 195–204, Santorini, 2003.

[9] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, April 2006.

[10] M. Kadous and C. Sammut. Classification of multivariate time series and structured data using constructive induction. *Machine Learning Journal*, 58:179–216, 2005.

[11] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2/3):107–123, 2005.

[12] D. Lee. Effective gaussian mixture learning for video background substraction. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(5):827–832, 2005.

[13] G. Medioni, I. Cohen, F. Bremond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(8):873–889, 2001.

[14] N. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):831–843, 2000.

[15] Ó. Pérez, M. Piccardi, J. García, and J. M. Molina. Comparison of classifiers for human activity recognition. In *IWINAC (2)*, pages 192–201, 2007.

[16] P. Remagnino, T. Tan, and K. Baker. Multi-agent visual surveillance of dynamic scenes. *Image Vision Computing*, 16(8):529–532, 1998.

[17] P. Ribeiro and J. Santos-Victor. Human activity recognition from video: modeling, feature selection and classification arquitecture. In *HAREM 2005: International Workshop on Human Activity Recognition and Modelling*, Oxford, UK, 2005.

[18] C. Simon, J. Meessen, D. Tzovaras, and C. De Vleeschouwer. Using decision trees for knowledge-assisted topologically structured data analysis. In *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, Santorini, June 2007.

[19] L. Wehenkel. *Automatic learning techniques in power systems*. Kluwer Academic, Boston, 1998.

[20] T. Xiang and G. Shaogang. Beyond tracking: Modelling activity and understanding behaviour. *International Journal of Computer Vision*, 67(1):21–51, 2006.

[21] A. Yilmaz and M. Shah. Actions as objects: A novel action representation. In *proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 05)*, pages 178–185, 2005.