GROUP BEHAVIOR RECOGNITION WITH MULTIPLE CAMERAS

Frédéric Cupillard, François Brémond and Monique Thonnat,
INRIA Sophia Antipolis
2004, route des Lucioles, BP93 - 06902 Sophia Antipolis Cedex, France
{Frederic.Cupillard, Francois.Bremond, Monique.Thonnat}@sophia.inria.fr

Abstract

We propose in this paper an approach for recognizing group of people behaviors using multiple cameras with overlapping FOVs (Field Of View). In this context, Behavior recognition first relies on low level motion detection and frame to frame tracking which generate a graph of mobile objects for each camera. Second, to take advantage of all cameras observing the same scene, a combination mechanism is performed to combine the graphs computed for each camera into a global one. This global graph is then used for long term tracking of groups of people evolving in the scene. Finally, the result of the group tracking is used by a higher level module which recognizes predefined scenarios corresponding to specific group behaviors. This article focuses on the graphs combination mechanism and on the recognition of group behaviors. At the end, results on these two algorithms are described.

1 Introduction

In this article, we present an approach for recognizing specific group behaviors such as violence or pickpocket in a cluttered scene (typically a metro scene) viewed by several cameras. Recent development in video acquisition hardware has made possible the acquisition of good quality video streams and increased the scientists interest in developing video surveillance systems [4, 5, 6]. The development of such systems present several difficulties and one of the most challenging is behavior analysis, since it requires the inference of a semantic description of the features (moving regions, trajectories,...) extracted from the video stream. The ambitious goal here is (1) to process real world video streams coming from metro stations taken for the European project ADVISOR (http://www-sop.inria.fr/orion/ADVISOR), (2) to combine information from multiple cameras, (3) to detect and track group of people and (4) to go up to the recognition of group behaviors. In our case, the cameras are calibrated and are overlapping each other and most of the time they are face to face.
The solution we propose is composed of four steps. First a motion detector and a frame to framer tracker generates a graph of mobile objects for each calibrated camera. Second, a combination mechanism is performed to combine the graphs computed for each camera into a global one. Third, this global graph is used for long term tracking of groups of people evolving in the scene (typically on hundreds of frames). Fourth, the results of group tracking are used for recognizing predefined scenarios corresponding to specific group behaviors.
In this article, we focus on the graph combination and the behavior recognition process. We describe the Behavior Recognition module through one example: "a group is Fighting".

2 Graph of moving regions for single camera

2.1 Motion detector

The goal of this step is to detect for each frame the mobile objects in the scene and classify them into a list of *moving regions* with labels corresponding to their type based on their 3D size, such as PERSON. This task can be divided into three sub-tasks: detection of *moving regions*, extraction of features, classification of *moving regions*. A list of *moving regions* is obtained at each frame. Each *moving region* is featured by 3D numerical parameters (center of gravity, position ,height,width,...) and by a semantic class (PERSON, OCCLUDED PERSON, GROUP, CROWD, METRO TRAIN, SCENE OBJECT, NOISE or UNKNOWN).

2.2 Frame to frame tracker

The goal of the frame to frame tracker (F2F Tracker) is to link from frame to frame the list of *moving regions* computed by the motion detector. The output of the frame to frame tracker is a graph of *moving regions*. This graph provides all the possible trajectories that a *moving region* may have. The link between a new *moving region* and an old one is computed depending on three criteria: their 2D (in the image) and 3D (in the real world) distance and the similitude between their bounding box size.

3 Combination of moving region graphs for multiple cameras

The goal of the combination phase is to combine all the graphs of *moving regions* computed by the F2F Tracker for each camera into a global one that we called the Combined Graph.

3.1 Data combination approach

As we mentioned in the previous section (F2F Tracker), a graph of *moving regions* is built frame by frame by computing links between the new *moving regions* detected and the old ones. To combine all these graphs, we combine together the *new moving regions* detected for all cameras. Each new *moving region* detected for each camera can be involved in one of the three following types of combination:

1. **Fusion**: the moving region is fused with one or several *moving regions* detected by other cameras. If the labels of all the *moving regions* are the same, we fuse them into one *moving region* by computing an average of their 3D features. By making such a fusion, the estimation of the 3D features (position, width, height) of the resulting moving region is improved (closer to the reality). If the label of the *moving regions* are different, we select for the Combined Graph the one which is the closest to the model of person.

2. **Selection**: the *moving region* is not fused with other *moving regions* and is selected for the Combined Graph. This *moving region* corresponds to a mobile object observed by one camera without any correspondence with other cameras.

3. **Removing**: the *moving region* is eliminated and is not present in the Combined Graph. This *moving region* corresponds to a noise just detected by one camera.

When the combination is done, the temporal links of the Combined Graph are inferred from the links computed for each individual graph. In the case of **Fusion**, if $\{q_1^t, .., q_n^t\}$ is the set of *moving regions* which are fused together into the *moving region* l_i^t at time t, l_i^t is linked with the set of previously combined *moving regions* $\{l_1^{t-1}, .., l_p^{t-1}\}$ which corresponds to the result of the combination of all the parents of the *moving regions* $\{q_1^t, .., q_n^t\}$ at time $t-1$. For the case of **Selection**, the process is similar with the difference that the *moving region* l_i^t results from the selection of only one *moving region* q_1^t. In the case of **Removing**, no link is computed because l_i^t has not been created. The main step in the graph combination phase is the combination of the *moving regions* newly detected by each camera. To manage this task, we use an iterative approach. If there are N cameras, we first combine the *moving regions* detected by the two first cameras, then the result is combined with the *moving regions* detected by the third camera and so on up to the last camera. Thus, we reduce the algorithm to a two dimension space in which we combine N-1 times information about 2 cameras.

3.2 Computation of the combination matrix

We compute a $n * m$ combination matrix to compare the correspondences between the *moving regions* $R = \{r_1 ... r_m\}$ detected by camera C1 and the *moving regions* $O = \{o_1 ... o_n\}$ detected by camera C2. We have 3 different criteria $\{k_1, k_2, k_3\}$ for doing this. For each criterion K_k, we construct a $n * m$ matrix N_k (n and m are respectively the number of *moving regions* detected for cameras C2 and C1). The element (i,j) of N_k is in the range $[0, 100]$ where zero indicates that there is no correspondence between the two *moving regions* r_j and o_i and 100 characterizes the strongest correspondence computed for criteria K_k. The final matrix F, called the Combination Matrix, is computed as the weighted sum of the matrices N_k

$$F = \sum_{k=1}^{3} w_k N_k$$

where the three global weights w_k are parameters of the combination algorithm.

The three criteria are :

1. **Position criteria** :
It estimates the spatial coherence between two *moving regions* o_i and r_j by computing their 3D proximity. This proximity is equal to 100 if both *moving regions* have exactly the same 3D position and decreases exponentially up to zero when the 3D distance between the two *moving regions* is above a threshold T (e.g. $5m$).

2. **Dimension criteria** :
For two *moving regions*, this criteria compares their label corresponding to their 3D dimension thanks to a class correspondence table. This comparison returns 100 if both *moving regions* have the same label and returns values which decrease up to zero when the labels indicates two classes of object very different (e.g. NOISE and GROUP).

3. **Temporal criteria** :
It reinforces the correspondence between two *moving regions* o_i and r_j when one parent of o_i and one parent of r_j have been fused together. This criteria returns 100 when the parents have been fused together, otherwise it returns 0.

We consider that two *moving regions* o_i and r_j have a high correspondence if the element (i, j) of the Combination Matrix F is higher than a threshold T, where T is also a parameter of the combination algorithm (we use $T = 70$). Then we use a set of rules to decide which type of combination to apply (see Section 3.4).

3.3 Extraction of sub combination matrices

We extract in the Combination Matrix F, disjoint sub matrices called F_s which correspond to a set of *moving regions* having correspondences together. Each sub matrix is constituted of two *moving regions* sub sets: one called R_s is a sub set of R and the other called O_s is a sub set of O. These sub sets are the maximum sub sets of neighbor *moving regions* for which each *moving region* of R_s has a high correspondence with at least one *moving region* of O_s. A *moving region* of R_s is a neighbor of another *moving region* of R_s if they have at least one high correspondence in common with one *moving region* of O_s.

3.4 Combination rules

This section describes the rules that we use for combining the *moving regions* $\{r_1 ... r_m\}$ and $\{o_1 ... o_n\}$ detected for two cameras C1 and respectively C2 by using the Combination Matrix F and the sub matrices F_s.

3.4.1 No correspondence

First, we process the *moving regions* which do not have any high correspondence in the Combination Matrix F.

In that case, a *moving region* detected for camera C1 or C2 does not have any high correspondences with a *moving region* detected for the other camera. Two possible situations in the real 3D scene can generate such a case.

1. A mobile object (in the real world) is in the field of view of only one camera, consequently, this mobile object is detected for this camera but not for the others (e.g. a person hidden by a pillar will be only detected by one of the two cameras).

2. A mobile object is in the field of view of both cameras, but it corresponds to a noise (e.g. a reflect on the floor). Consequently, it is possible that only one of the two cameras has suitable features (position, direction,...) to detect this mobile object.

When we process a *moving region* without high correspondences, we first check whether it corresponds to a mobile object which is in the field of view of both cameras. If it is the case and if the dimension of the *moving region* is too small (typically if its label is UNKNOWN or NOISE), this *moving region* is **removed**. Otherwise, the *moving region* is **selected** for the Combined Graph.

3.4.2 Fusion in case of good correspondence

Second, we process the combination of *moving regions* having good correspondences. In that case, the size of the sub matrix F_s is $1 * 1$. Such a dimension means that there are one or several mobile objects (in the real world) which are detected as only one *moving region* for each camera (one *moving region* r_j detected for camera C1 and one *moving region* o_i detected for camera C2). The two *moving regions* r_j and o_i are **fused** together (see the definition of fusion in Section 3.1).

3.4.3 Combination in case of ambiguity

Third, we process the combination of sub sets of *moving regions* R_s and O_s which have ambiguous correspondences.

- **Noise removing**

This corresponds to a prefiltering task which is applied on every rectangle $n * m$ ($n \neq m$) sub matrices in order to eliminate the *moving regions* that correspond to noise even if they have one or several high correspondences. This situation can occur when a person is detected by the two cameras C1 and C2 whereas a noise is detected close to this person only by one of the two cameras. In that case, the *moving region* corresponding to the noise could have a high correspondence with the one corresponding to the person. The approach used to solve this ambiguity is similar to the one used in the case of No Correspondence (see Section 3.4.1). We identify in the sub set (R_s or O_s), with the largest number of elements, the *moving regions* too small (labeled UNKNOWN or NOISE) that belong to the FOV of both cameras C1 and C2. As the probability that such *moving regions* correspond to a noise is big enough, these *moving regions* are **removed**. Thus, the size of the sub matrix is reduced and the sub matrix can be processed by the other combination rules.

- **Correct but noisy correspondence**.

This corresponds to square sub matrices $n * n$ ($n > 1$). Such a sub matrix implies that the number of mobile objects detected by each camera C1 and C2 is the same but their correspondence is ambiguous. We distinguish two cases:

The first case corresponds to a $2 * 2$ sub matrix F_s. Two fusion combinations between the two sub sets $R_s = \{r_{s,1}; r_{s,2}\}$ and $O_s = \{o_{s,1}; o_{s,2}\}$ are possible. Either we fuse the *moving regions* associated to the main diagonal of F_s ($r_{s,1}$ is fused with $o_{s,1}$ and $r_{s,2}$ with $o_{s,2}$) or we fuse the *moving regions* associated to the second diagonal of F_s ($r_{s,1}$ is fused with $o_{s,2}$ and $r_{s,2}$ with $o_{s,1}$). We perform the **fusion** of the *moving regions* associated to the diagonal (main diagonal or second diagonal) for which the sum of the *moving region* correspondences is the biggest and is higher than a threshold ($T_d = 90$). If the *moving regions* on the diagonals have not high enough correspondences, we **select** for the Combined Graph the *moving regions* of the camera which have the largest 3D size to keep a maximum of information.

The second case corresponds to a $n * n$ ($n > 2$) sub matrix F_s. In a theoretical point of view, we should compare all the possible fusion combinations ($= n!$). However, in order to ease the algorithm scheme and to save time processing, we do not explore and compare all the possible fusion combinations but we rather try to reduce the size of the sub matrix by achieving the fusion between the two *moving regions* that have the highest correspondence, if this correspondence is high enough (above a threshold $T_s = 90$). If such a correspondence exists, the two corresponding *moving regions* are **fused** together. The size of F_s is reduced and F_s is re-processed by the combination rules. On the other hand, if the highest correspondence is not high enough (< 90), we **select** for the Combined Graph the *moving regions* of the camera which have the largest 3D size to keep a maximum of information.

- **Incorrect and noisy correspondence**.

This last case corresponds to rectangle $n * m$ ($n \neq m$)

sub matrices F_s and is managed similarly to the previous case ($n * n, n > 2$ sub matrix F_s). We fuse together the two *moving regions* that have the highest correspondence if this correspondence is high enough (> 90). The size of F_s is reduced and F_s is re-processed by the combination rules. If the highest correspondence is not high enough, we **select** for the Combined Graph the set of moving regions of the camera which have the largest number of elements to maximize the number of isolated mobile objects (see figure 1).

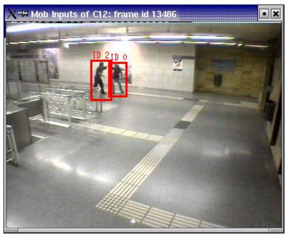
3.5 Graph combination results

This fusion module has been tested on several metro sequences. In this section we are showing images of the processed videos. In these images, a red box corresponds to a *moving region* classified as a PERSON and a green box corresponds to a *moving region* classified as a GROUP. On figure 1, the two top images show the *moving regions* detected for each camera and the two bottom images show the projection for each camera of the *moving regions* resulting from the combination process. This figure illustrates the case where two persons are detected as one moving region by the camera C1 (a1) whereas they are detected as two separated moving regions by the camera C2 (a2). A $2 * 1$ sub combination matrix F_s is constituted (case: Incorrect and noisy correspondence). As there is not high enough correspondences in F_s and as there are more moving regions detected by camera C2, the moving regions of camera C2 are selected. Thus the combination process has improved the detection of camera C1 (b1): the two persons are now detected as two separated moving regions. Currently, the main limitation of the combination module is an over estimation of the number of persons. This can occur when the same person is detected as one *moving region* (labeled as a PERSON) from two cameras and one of these *moving regions* is not detected in the field of view of the other camera due to 3D position errors. In that case, the combination process cannot make the correspondence between the *moving regions* and decide to keep both *moving regions* (corresponding to the same individual in different cameras) in the Combined Graph.

4 Long term group tracking

The goal of the Long Term Group Tracking (LTG Tracker) algorithm [3] is to track globally all the persons belonging to the same group over a long period of time and to compute the history of groups. It uses a 3D model of the scene, and is based on explicit models of group and of the trajectory of people inside a group.
The LTG Tracker detects and follows the real groups of people in the scene by computing then selecting, inside the global graph, the trajectories of *moving regions* that can correspond to the persons inside the real group. While a group is tracked, an history containing relevant information

a1: mono camera (C1) detection



a2: mono camera (C2) detection

b1: detection improved for camera C1

b2: detection selected for camera C2

Figure 1. The combination algorithm uses one view where people are isolated to separate them on the other view.

such as position or trajectory about the group is computed. Contrary to traditional trackers [1, 2] that track each person individually, this tracker tracks globally all the persons belonging to the same group and tracks these persons with a time delay to keep coherent the structure of the group during the entire sequence. As the LTG Tracker takes as input the global graph combining all the information coming from the cameras observing the scene, the results of the group tracking are more robust.

5 Behavior recognition process

The goal of this task is to recognize specific behaviors and specially to recognize group behaviors occurring in a metro scene. A main problem in behavior recognition is the ability to define and reuse methods to recognize specific behaviors, knowing that the perception of behaviors is strongly dependent on the site, the camera view point and the individuals involved in the behaviors. Our approach consists in defining a formalism allowing us to write and easily reuse all methods needed for the recognition of behaviors. This formalism is based on three main ideas. First the formalism should be flexible enough to allow various types of operators to be defined (e.g. a temporal filter or an automaton). Second, all the needed knowledge for an operator should be explained within the operator so that it can be easily reused. Finally, the description of the operators should be declarative in order to built an extensible library of operators.

5.1 Behavior representation

The entities needed to recognize behaviors represent different types of concepts which are:

- **The states**: A state is a situation describing an actor at time *t* (e.g. a group is agitated) or a characteristic of an actor constant during two consecutive instants (e.g. a group carries on being agitated). In the case of group behaviors, an actor is a group tracked by the LTG Tracker (see Section 4) or a static scene object (equipment, zone of interest).
- **The events**: An event is a change of states at two consecutive times (e.g. a group starts to be agitated).
- **The scenarios**: A scenario is a combination of states and events. Behaviors are specific scenarios (dependent of the application) defined by the users.

5.2 Behavior recognition

To compute all the needed entities for the recognition of behaviors, we use a generic framework based on the definition of **Operators** which are composed of four attributes:

- **Operator name**: indicates the entity to be computed.
- **Operator input**: Description of input data. There are two types of input data. First, data relative to the actors involved in the entity such as the trajectory of the actors. Second, data computed by other **Operators**.
- **Operator body**: set of methods to compute the entity.
- **Operator output**: result of the entity computation stored in a specific zone (accessible by all the other **Operators**). This result corresponds to the value of the entity at the current time.

Methods used to compute the entities are the following:

- **State methods**: we use numerical methods which include the computation of: (1) 3D distance for states dealing with spatial relations, (2) the evolution of temporal features for states dealing with temporal relations and (3) the global speed and internal speed for states dealing with spatio-temporal relations. We also use symbolic comparison to check that a symbolic characteristic of an actor is constant during two consecutive instants. The output of these numerical or symbolic methods is then classified to obtain a symbolic value.
- **Event methods**: we compare the status of states at two consecutive instants. The output of an event method is boolean: the event is either detected or not detected.
- **Scenario methods**: for simple scenarios (composed of only 1 state), we verify that a state has been detected during a predefined time period using a temporal filter. For more complex scenarios (composed of several states and events), we use finite state automatons. The automaton alphabet is composed of states or events.

This generic framework based on **Operators** gives two advantages: It first enables us to test a set of methods to compute an entity, independently of other entities. So we can locally modify the system (the methods to compute an entity) while keeping it globally consistent (without modifying the meaning of the entity). Second, the network of **Operators** to recognize one scenario is organized as a hierarchy. The bottom of the hierarchy is composed of states and the top corresponds to the scenario to be recognized. Several intermediate levels, composed of state(s) or event(s) can be defined. For example, the scenario "A group is fighting" is recognized with an **Operator** network composed of four levels (see figure 2):

- **level 1**: two states: "Global Speed of Group" $(St1)$ and "Internal Agitation of Group" $(St2)$ are defined. $(St1)$ computes the speed of the group. The speed of an agitated group is generally quite big. $(St2)$ computes the lateral speed between the persons inside the group. For doing this, we compute for each frame the lateral distance between the *moving regions* inside the group. The lateral speed is then computed as the maximal difference of the lateral distances between two frames on a temporal window of N frames (we use $N = 5$). The lateral speed of a group increases significantly if the *moving regions* inside the group go away or go close to each other quickly and decreases when the relative motions of the *moving regions* inside the group are stable.
- **level 2**: one state: "Agitation of Group" $(St3)$ is defined. It combines the global speed and the lateral speed of a group and computes the symbolic value "agitated" or "not agitated".
- **level 3**: two events: "A group starts to be agitated" $(Ev1)$ and "A group stops to be agitated" $(Ev2)$ are defined and enable to detect a change of the status of the state $(St3)$ at two consecutive instants. These two events generate the boolean value at each frame: "detected" or "not detected". One state: "Agitation continuity of a Group" $(St4)$ is defined and enables to detect a $(St3)$ state continuity at two consecutive instants. It generates the symbolic value at each frame: "continuous agitation" if the $(St3)$ state status is "agitated" for the last two frames, "continuous non-agitation" if the $(St3)$ state status is "not agitated" for the last two frames or "not continuous" if the $(St3)$ state status changes in the last two frames. At each frame, a group can be involved in only of the four following state or event detection: (A) "A group starts to be agitated", (B) "continuous agitation", (C) "A group stops to be agitated" or (D) "continuous non-agitation".
- **level 4**: one scenario: "A group is fighting" $(Sc1)$ is defined. It takes as input the detection result (A), (B), (C) or (D) of the third level.

The scenario on the fourth level is recognized with an automaton described on figure 3 and composed of:

- **An alphabet**: $\{(A), (B), (C), (D)\}$ corresponds to the detection of the state and events of third level at each frame.
- **Three automaton states**: $\{Q0, Q1, Q2\}$. Q0 is the initialization state and enables to compute the number of consecutive detections of the (B) state. Q1 enables to compute the number of consecutive detections of the (D) state after a (A) or (C) event have been detected. Q2 is the final state which corresponds to the recognition of the scenario.
- **two state variables**: $\{Bsucc, Dsucc\}$. $Bsucc$ counts the number of consecutive detections of the (B) state. $Dsucc$ counts the number of consecutive detection of the (D) state. $\{ThB, ThD, ThR\}$ are the thresholds on the state variables to control the transitions.When the number of consecutive detections of the (B) state is high enough ($> ThB$) or when two consecutive (A) event, separated by no more than ThD consecutive detections of the (D) state, is detected, so the scenario: "A group is fighting" is recognized and it will be still recognized while the number of consecutive detections of (D) state or (C) event is lower than ThR.

These states variables are necessary to take into account the erratic detection of the state and events of the third level (detection based on two consecutive frames). These variables enable the automaton to keep an history of the recognition process. For example, in a typical case of a group fighting, the group will be detected as agitated on different frames but between these frames, the group will be detected as not agitated. The state variables will enable to keep the scenario recognized on the whole interval. On the opposite, when someone quickly joins the group, the group will be detected as agitated but thanks to the state variables, the scenario will not be recognized.

5.3 Group behavior recognition results

In our **Operator** hierarchy, we have defined several states, events and scenarios given by the following lists:

- **states**: "Global Speed of Group", "Internal Agitation of Group", "Agitation of Group", "Presence of Group In Forbidden Area" (e.g. the track), "Presence of Group In Blocking Zone" (e.g. entrance zone of the escalators), "Agitation continuity of a Group" and "A Group is Stationary".
- **events**: For each state defined above, the two events: "start to be in the state" and "stop to be in the state" have been defined (e.g. for the state: "Internal Agitation of Group", the two following events have been defined: "a group starts to be agitated" and "a group stops to be agitated").
- **Scenarios**: two scenarios have been defined: "A group is fighting" (see figure 2) and "A group is blocking the way of other individuals".

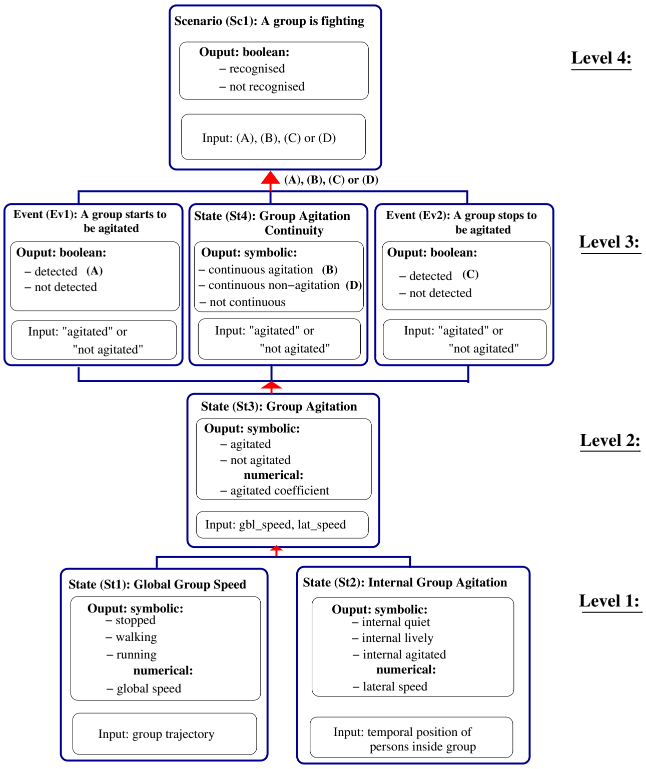


Figure 2. Operators network for the scenario: "A group is fighting"

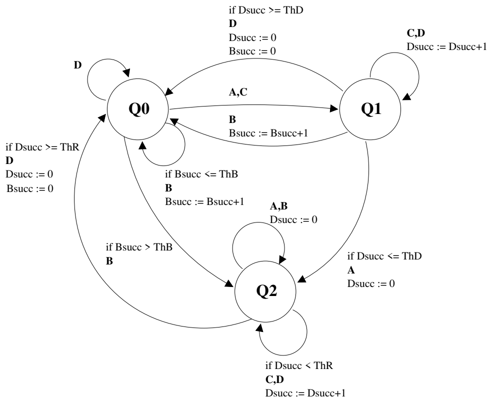


Figure 3. Finite state automaton used to recognize the scenario: "A group is fighting"

The states, events and scenarios we have defined have been tested on 20 metro sequences. The longest lasts 44 minutes (more than 13000 frames). The percentage by frame of good detection for states and events is about 70% for the states and events relative to group agitation (e.g. "Internal Agitation of Group") and about 95% for the others.

Consequently, the percentage of good recognition for the scenario "A group is blocking" (composed of the three states: "Global Speed of Group", "Presence of Group In Blocking Zone" and "A Group is Stationary") is about 95% whereas the percentage of good recognition for the scenario

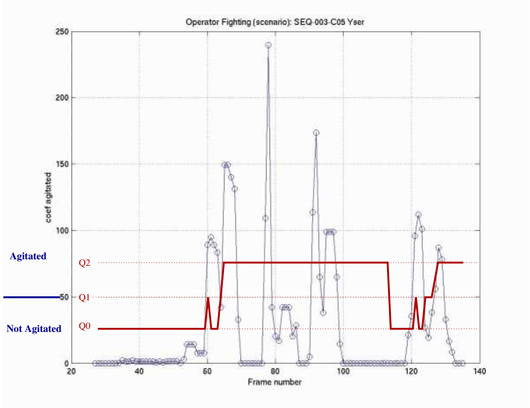


Figure 4. Resulting graph of the scenario: "A group is fighting".

"A group is fighting" is about 70%. The graph of Figure 4 describes the result of the recognition of the scenario: "A group is fighting" on a fighting sequence in a Hall of a Brussels metro station. The X axis corresponds to each processed frame (from frame 20 to 140) and the Y axis corresponds to the level of global agitation (output of the state "Agitation of Group"). We can see that the evolution of the level of agitation is very irregular. As an overlay of this diagram, we had the value of the three automaton states (Q0, Q1 and Q2) of the scenario "A group is fighting". These values show that the automaton is able to correctly recognize the scenario most of the time where the persons are fighting. On figure 5, we can see typical images of the sequence.

frame 36: the group is not agitated   frame 66: the group is fighting

Figure 5. Images illustrating the recognition of the scenario: "A group is fighting"

Currently, the main limitation of the behavior recognition is how we combine states and events to recognize scenarios. We are using an heuristic to maximize the recognition of scenarios on a set of given sequences. For example for the scenario "A group is fighting", we are using a combination of four states and two events. This combination gives good results (70% of good detection) but there are still some false alarms (3% of incorrect detection) on other sequences where people are not fighting.

6 Conclusion

We introduced a new approach for recognizing group of people behaviors using multiple cameras. This paper focuses on two characteristics of this approach. First, it combines the mobile objects detected by several cameras with overlapping FOVs to improve the tracking of groups of people. From our experience, we find out three ways for combining mobile objects: (1) when cameras are close to each other and the detection is good, the combination can be based on the 2D appearence of mobile objects (image feature), (2) when the detection is average, the combination should be based on the 3D position and 3D size of the mobile objects and (3) when the cameras are far from each other and the detection is bad, the combination can only be based on the 3D trajectory of the mobile objects. As in our application, the cameras are located in a metreo station (case (2)), we propose an algorithm optimizing the correspondances between mobile objects based on their 3D position and 3D size. Second, the good quality of the tracking process allows us to perform a recognition of high level scenarios. This recognition process relies on a hierarchy of operators, each operator corresponds to a method to recognize behavior entities (states, events and scenarios). This hierarchy of operators enables the end user to built his own methods to recognize application dependent behaviors. This approach has been validated on 20 metro scenes and we have been able to correctly recognize two types of scenario of group of people: "A group is fighting" with 70% of good detection and "A group is blocking the way of other individuals" with 95% of good detection. However, for the fighting scenario, the rate of false alarms is still too important (3% of incorrect detection). One way to reduce these false alarms is to to use learning techniques (Dynamic Bayesian Network) to optimize the combination of states and events leading to the recognition of the fighting scenario.

References

[1] F. Brémond and M. Thonnat. Tracking multiple non-rigid objects in a cluttered scene. In *proc. of the 10th Scandinavian Conference on Image Analysis (SCIA)*, Lappeenranta (Finland), June 1997.

[2] I. Cox and S. Hingorani. An efficient implementation of reid's Multiple Hypothesis Tracking algorithm and its evaluation for the purpose of visual tracking. In *IEEE Transactions on pattern analysis and machine intelligence*, volume 18, Sept. 1996.

[3] F. Cupillard, F. Brémond, and M. Thonnat. Tracking group of people for video surveillance. In *proc. of the 2nd European Workshop on Advanced Video-Based Surveillance System*, University of Kingston (London), Sept. 2001.

[4] S. Hongeng, F. Brémond, and R. Nevatia. Bayesian framework for video surveillance application. In *Proc. of the 15th International Conference on Pattern Recognition*, Barcelona (Spain), Sept. 2000.

[5] R. Howarth and H. Buxton. Conceptual descriptions from monitoring and watching image sequences. In *Image and Vision Computing*, pages 105–135, 2000.

[6] A. Pentland and A. Liu. Modeling and prediction of human behaviour. In *Neural Computation*, pages 229–242, 1999.