Group interaction and group tracking for video-surveillance in underground railway stations

Sofia Zaidenberg[1], Bernard Boulay[1], Carolina Garate[1], Duc-Phu Chau[1], Etienne Corvée[1], and François Brémond[1]

INRIA Sophia-Antipolis – Mediterannée,
2004, route des Lucioles, 06902 Sophia-Antipolis, France
Sofia.Zaidenberg@inria.fr

Abstract. In this paper we propose an approach to recognize behaviors of groups of people in the subway. Violent behavior or vandalism performed by a group can be detected in order to alert subway security. The proposed system is composed of 3 main layers: the detection of people in the video, the detection and tracking of groups among the detected individuals and the detection of events and scenarios of interest based on tracked actors (groups). The main focus of this paper are the group tracking and event detection layers.

Keywords: events detection, behavior recognition, automatic video understanding, tracking

1 Introduction

In this paper, we propose a method for detecting events regarding the behavior of groups of people in a subway station. Events of interest are encoded into scenario models using a descriptive language and can be used at different levels. Non-urgent events enable the analysis of global behavior in the subway and the usage of equipment (for instance, queues at vending machines can be detected). Urgent events such as violence and vandalism increase subway security since they allow alerting the staff. This method can be divided in the following steps: 1) Human detection. 2) Frame-to-frame tracking. 3) Group detection and tracking. 4) Event detection. In the following sections we will describe the steps of our method, and show experimental results.

Our approach is able to detect people in an image based on (Local Binary Pattern) features trained with Adaboost. The frame-to-frame tracking associates detected people from one frame to another, allowing their tracking. The group tracking algorithm analyzes the trajectories of detected people and associates them in groups based on their properties. Group dynamics can vary. Group members can temporarily move away from each other and then gather closely again. To robustly detect and track real groups of people, even in such cases, the proposed method uses a time delay (a temporal window of analyze).

Tracking people, and especially groups of people in relatively unconstrained, cluttered environments is a difficult task for various reasons. McKenna et al. in [3] propose a method to robustly track people as they form groups using an adaptive background subtraction method and relying on color information to disambiguate occlusions. But their goal is to track individuals before and after they have been in a group, not to keep track of people evolving as a group. In [2], Ge et al. propose a method to discover small groups of people in a crowd based on a bottom-up hierarchical clustering approach. Trajectories of pedestrians are clustered into groups based on their closeness in terms of distance and velocity. The experiments of this work have been made on videos taken from a very elevated viewpoint, providing less occlusions than in our testing video set.

2 Methodology

2.1 Human Detector and F2F tracking

Human Detector People are detected using LBP (Local Binary Pattern) [5] features trained with Adaboost. For training the NICTA database [6] is used, containing 10,000 positive samples. For negative examples, various background images from the NICTA database are used.

Fig. 1. Example of human detection.

The input image is scanned at multiple resolutions. The minimum size of the scanning window is usually 48x96 pixels, and is increased by 10% until reaching the input image size. A filtering is then performed on the candidates to extract one output object from many overlapping candidates following the rule: one output object is generated from a minimum number (*e.g.* 4) of candidates having a sufficient overlapping ratio (*e.g.* $> 50\%$).

The output of the human detector is a set of regions, each of them containing one or several persons. Several people will be detected as one if they are too close and overlap each other (one is occluded by the other). Each region (bounding box) is characterized by its 2D (in the image) and 3D (in the calibrated scene) coordinates. We will use the term *mobile* to designate such regions. Figure 1 shows an example of a detected human (blue regions are candidates and the green region is the output object).

F2F tracking The goal of Frame-to-Frame tracking is to compute the similarity between two any detected objects appearing in a given temporal window to establish possible links.

For each detected object pair in a given temporal window (of a predefined size), the system computes their similarity using a feature pool including 2D and 3D displacement distance, 2D area and shape ratio, RGB color histogram, histogram of oriented gradients (HOG), color covariance and dominant color. A temporal link is established between two detected objects when their similarity is greater than a predefined threshold. At the end of this stage, we can obtain a

weighted graph whose vertices are detected objects in the considered temporal window and edges are the temporal established links with the object similarities (see figure 2(a)). We decide to consider all possible links in a temporal window so that if a mobile object cannot be detected in few frames, it can still be tracked. In order to decrease the algorithm complexity, for each object, the system looks for its matched candidate objects in a spatial neighborhood with a predefined radius.

In the rest of this paper we will use the term of *father* and *son* to designate the mobile resp. in the oldest and in the most recent frame of a link.

2.2 Group Tracking

Definition of a group of people To detect that people are in a group, we need a formal definition of a group in terms of constraints on people's properties. The human definition of a group is *people that know each other or interact with each other*. In fact, McPhail gives the following definition for the sociological sciences field in [4]: *Two defining criteria of a group:* proximity *and/or* conversation *between two or more persons*.

It is quite difficult to directly detect knowledge that people have of each other, their interactions or conversation in a video. We rather work with a derived definition on observable properties: *two or more people who are spatially and temporally close to each other and have similar direction and speed of movement*. In order to get more robust results and eliminate false positives, we introduce a time delay for detecting groups. Those constraints have to be met not instantly but for a certain amount of frames to detect a group. Once that several people have been associated in a group, the algorithm will try to keep them associated for as long as possible even if they move temporarily away from each other. For instance, if one person from a group goes to use the vending machines while the others wait for him, this person should be kept as part of the group.

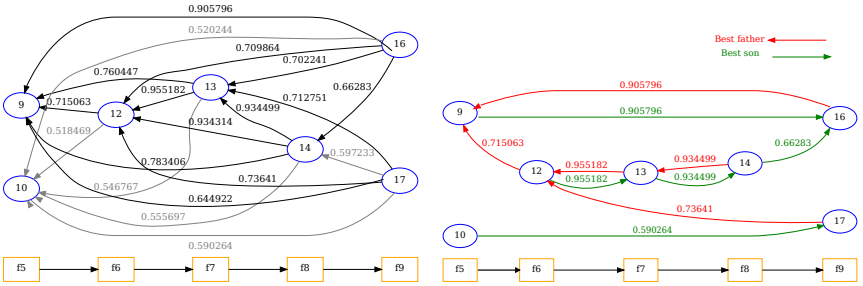
Temporal window of analyze For a better robustness to situations where several people meet the conditions to form a group randomly for a short time, we introduce a temporal window of analyze. Social conventions dictate people to keep a polite distance even when having similar trajectories. For instance, when getting off the train together people might be close but will move away while walking along the platform to the same exit. Thus we can modify the definition of a group to: *two or more people who are spatially and temporally close to each other and have similar direction and speed of movement for a minimum duration*.

In the rest of this paper we will denote by t_c the current frame of the stream and by $t_c - T$ the frame being currently processed by the group tracking algorithm, T being the delay (in quantity of frames).

Trajectory of a person The F2F tracker provides a set of father links for each detected mobile (see figure 2(a)). To simplify the rest of the algorithm, we

extract mobile's trajectories by choosing for each mobile one *best father* among its father links and one *best son* among links it is the father of.

As a consequence, from the output graph of the F2F tracker (figure 2(a)), we keep the simplified graph, including only best fathers and best sons, shown in figure 2(b). One can notice that the best-father/best-son relationship is not symmetric. A mobile can be the best father or best son of zero, one or more mobiles.



(a) Detected mobiles and their links, *e.g.* mobile 13 in frame 7 is the father of mobile 16 in frame 9 with reliability 0.7.

(b) Simplified graph obtained from (a) by keeping for each mobile one exiting (best father) and incoming (best son) link

Fig. 2. An example of F2F tracker output: mobiles (ellipses) and their links. Squares represent frames.

Group properties According to our definition of a group (section 2.2), a group is a set of mobiles having similar locations, directions and speed for a noticeable duration. The properties of a group are thus the average distances between mobiles in the same frame and the average over frames of standard deviations of speed and direction.

The average of distances between mobiles is computed using the following equation: $distanceAvg = \frac{1}{nFrames} \sum_{k=1}^{nFrames} \frac{1}{nPairs_k} \sum_{i=1}^{n_k} \sum_{j=i+1}^{n_k} dist(i,j)$, where $nFrames$ is the number of frames containing mobiles that belong to the group, $nPairs_k$ is the number of pairs of mobiles in frame k, n_k is the number of mobiles in frame k, $i, j \in [1; n_k]$ are mobiles in frame k and $dist(i,j)$ is the norm of the motion vector[1] between i and j.

The average over frames of standard deviation of speed follows the equation: $speedStdDev = \frac{1}{nFrames} \sum_{k=1}^{nFrames} \sqrt{\frac{1}{n_k} \sum_{i=1}^{n_k} (s_i - m_k)^2}$, where s_i is the average speed of mobile i, *i.e.* the average over frames of the speeds of i's ancestors along its trajectory. The instantaneous speed of a mobile is defined as the norm of the motion vector between the mobile's best father and the mobile itself. m_k is the average of mobiles' speed at frame k.

[1] The motion vector is a vector in the 3D space from the bottom center of one bounding box to the same point of another.

The average over frames of standard deviation of direction is computed as follows: $directionStdDev = \frac{1}{nFrames}\sum_{k=1}^{nFrames} dirSD_k$ with $dirSD_k = \sqrt{\frac{1}{n_k}\sum_{i=1}^{n_k}\frac{1}{3}((m_{i_x}-d_{k_x})^2+(m_{i_y}-d_{k_y})^2+(m_{i_z}-d_{k_z})^2)}$, where m_i is the motion vector of mobile i (the vector between i's best father and i) and d_k is the average direction vector of mobiles at frame k.

Finally, we define the global group coherence as the weighted sum of these three quantities: $groupCoherence = \omega_1 \cdot distanceAvg + \omega_2 \cdot speedStdDev + \omega_3 \cdot directionStdDev$, where the weights ω_1, ω_2 and ω_3 are parameters. In the experiments described section 3 we used $\omega_1 = 7$ and $\omega_2 = \omega_3 = 5$ to slightly favor distance over speed and direction similarity which are quite noisy. One should notice that with this definition, a low value of *groupCoherence* is significative of a group.

The group tracking algorithm The group tracking algorithm is divided into 4 parts: creation of groups, update of groups, split/merge of groups and termination of groups, although the first step of the algorithm is the update step.

The currently processed frame is frame $t_c - T$ and the input are mobiles detected at frame $t_c - T$ and existing groups at frame $t_c - T - 1$. Updating a group is adding mobiles of the current frame that correspond to the members of the group in the previous frame – *i.e.* a mobile having its best father in the group. A member should only be added if they will stay in the group, or move away temporarily. To take a decision about adding a mobile to a group at frame $t_c - T$, we consider the relative evolution of the mobile and the group through the time window (from frame $t_c - T$ to t_c) using the *groupCoherence* defined above (section 2.2).

To reduce computation cost, we introduce the notion of *best* mobile of a group. The best mobile is defined as the closest to the center of gravity of the group. Instead of analyzing the evolution of all group members through the window, we only consider the future trajectory of the best mobile and of the mobile considered for update. This algorithm is summarized below (algorithm 1).

The $futureCoherence$ function combines the *groupCoherence* values at each frame (c_f) to obtain a value representative of how well the mobile is integrated into the group in the future: $futureCoherence = (1/\sum_{i=1}^{n} w_i)\sum_{i=1}^{n} w_i c_{i+t_c-T-1}$, where $n \in [1;T]$ is the number of frame coherence values (c_f) obtained after the loop through the time window. If the chain of sons of either m or m_{best} is interrupted before frame t_c, the number of coherence values is lower ($n < T$).

The weights w_i favor the most immediate future. The number of c_f values can vary. The mobile m or m_{best} may not be visible through the whole window of analyze. The resulting value of $futureCoherence$ should be comparable to the same threshold whatever the value of n. The following equation ensures the first weights to be higher in case of a low value of n than in case of a higher value of n: $w_i = \frac{1}{n}e^{-\frac{1}{n}}$.

The update step assigns all mobiles that should belong to an existing group to their group. The remaining mobiles are eligible for group creation.

Algorithm 1: Update of groups.

```
input  : groups_{t_c−T−1}, mobiles_{t_c−T}
output : updated groups_{t_c−T}
for m ∈ mobiles_{t_c−T} do
    g = groupOf(bestFather(m));
    if g ≠ ∅ then
        m_best ← bestMobile(g);
        m_tmp ← m; g_tmp ← g;
        for f = t_c − T to t_c do
            g_tmp.add(m_tmp);
            g_tmp.add(m_best);
            c_f ← groupCoherence(g_tmp);
            m_tmp ← bestSon(m_tmp);
            m_best ← bestSon(m_best);
            if m_tmp = ∅ or m_best = ∅ then
                break;
        c ← futureCoherence({c_f});
        if c < threshold then
            g.add(m);
```

The group creation step computes the *groupCoherence* of every pair of available mobiles and creates a group if the resulting value is low enough. Similarly to the update step, this evaluation is performed through the window of analyze. A group is created only from mobiles that form a group for a noticeable amount of time. This way, we avoid including randomly formed groups by people that pass by each other. If pairs (a, b) and (b, c) are eligible for group creation, the group (a, b, c) is created.

The termination of groups step erases old groups. Mobiles that were detected at a largely outdated frame $(t_c - 5T)$ are deleted at frame $t_c - T$ and empty groups are erased. Groups having no new mobiles for $5T$ frames are erased. All existing groups, even currently empty ones, can potentially be updated.

The split of groups will operate naturally. When a mobile moved away for too many frames from its group, it is no more added to the group (the *groupCoherence* raises above the threshold) and so it splits from the group. If several mobiles split from a group, they will become candidates for group creation and this way a bigger group can naturally split into smaller ones.

Two groups g_1 and g_2 can be merged if two mobiles, one in each group at frame $t_c - T + k$ ($k \in [0; T-1]$), have the same best son at frame $t_c - T + l$ ($l \in [k+1; T-1]$). The oldest group among g_1 and g_2 is kept and all mobiles of the disappearing group are added into the remaining group.

2.3 Event Detection

Event detection is a key task in automatic understanding of video sequences. This detection is usually based on the detection of objects of interest (with video algorithms) and on *a priori* knowledge of the scene (the context).

The main categories of approaches used to recognize events are the probabilistic/neural network based techniques and the symbolic network based techniques. The probabilistic techniques mainly use HMMs (Hidden Markov Models) and their variants or Petri nets [1]. These techniques are appropriate to model uncertainty during recognition because of the properties of HMMs. But, since they are time-sliced, their modeling of temporal relationships is limited. This leads to difficulties to model complex activities. In this work, we propose to use the *ScReK* tool (Scenario Recognition based on Knowledge) for modeling the knowledge and recognition of events based on a symbolic technique (see figure 3). The recognition approach solves spatio-temporal constraints between objects of interest [7].

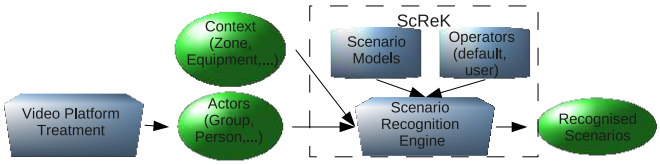


Fig. 3. Overview of the event detection with ScReK tool. Users define knowledge of their application: objects of interest, scenario models and specific operators if needed. The scenario recognition takes as input this information and the actors detected by the video platform, to recognize scenarios by solving spatio-temporal constraints of the models.

Description of the ontology One of the main advantages of using the *ScReK* tool is to easily model the knowledge of the studied domain: the *ontology*. This knowledge is composed of the objects of interest on which scenario models depend, and the scenario models themselves.

The *ScReK* tool proposes a simple language to describe the objects of interest. The main idea is to describe each object by declaring its attributes with help of basic types (*e.g.* Integer, String, 3D point, ...). Moreover, the user can also describe a hierarchy in the objects (*e.g.* a *group* is a *mobile* object).

```
class Group:Mobile
{
   const false;
   CSInt NumberOfMobiles;
   CSDouble AverageDistMobiles;
}
```

```
class Zone:ContextualObject
{
    const true;
    CS3DDPoints Vertices;
    CSDouble Height;
}
```

A *Group* is a *Mobile*. A *Group* is not constant (dynamic, *i.e.* its attributes can change their values with time). One of its attributes is *AverageDistMobiles* which is the average distance between mobiles of the group. A *Zone* is a *ContextualObject*. It is constant in time, and it is composed of *Vertices* (the list of points defining the zone) and a *Height*.

The constraints used in the scenario models are based on operator (e.g. *in* takes as input a point p and a set of points S to check if p is inside S). The ScReK tool allows the user to model their own operators to define constraints specific to their domain.

A scenario model is composed of 5 parts:

- **PhysicalObjects**: the list of the physical objects involved in the model. The types of these objects are defined by the user in the ontology.
- **Components**: the list of sub-events composing the model.
- **Constraints**: the list of the constraints for the physical objects and/or the components. The constraints can be temporal (between the components), spatial or symbolic (for physical objects).
- **Alarm**: describes the importance of the scenario model.
- **Action**: is the list of specific treatments which can be launched when the model is recognized (*e.g.* to provide feedback to low level vision algorithm).

```
CompositeEvent(Queue_at_vending_machine,
 PhysicalObjects((g:Group),(z:Zone))
 Components((c:Group_Stays_Inside_Zone(g, z)))
 Constraints((g->NumberOfMobiles > MIN_NUM_QUEUE)
     (z->Name = "Vending_Machine_Zone"))
 Alarm ((Level : NOTURGENT)))
```

The scenario `Queue_at_vending_machine` is recognized if the primitive event `Group_Stays_Inside_Zone` is activated for the zone in front of the vending machine and if the group has a minimum number of mobiles.

Event Recognition The usual algorithms to recognize events can be time consuming. The *ScReK* tool proposes a solution to reduce the computation time. The user has to define optimal scenario models: at most two components, at most one temporal constraint between these components (the ScReK tool proposes the Allen's interval algebra). This property is not restrictive since all scenario models can be optimized in this format. From the properties of the optimal scenario model, the scenario model tree is computed. The tree defines which sub-scenario (component) triggers the recognition of which one: the sub-scenario which happens last in time triggers the recognition of the scenario. For instance, the scenario A has two components B and C. The temporal constraint is B *before* C. Then the recognition of C triggers the recognition of A. The tree triggers the recognition of the only scenarios that can happen.

The first step of the event recognition process is to recognize all the possible simple scenarios (primitive states) by instantiating all the models with the detected objects (*e.g.* instantiating the model `Group_Stays_Inside_Zone` (takes as input one group and one zone) for all the detected groups and all the zones of the context).

The second step consists in recognizing complex events according to the scenario model tree and the simple events previously recognized.

The third step checks if the recognized event at time t has been already recognized previously to update the event (end time) or create a new one.

3 Results

We used the framework of ViSEvAl (visualization and evaluation tool) provided by the Pulsar team at INRIA under AGPL license and available on request[2] for evaluating group tracking. This framework provides metrics and tools for evaluating and visualizing detection, classification, tracking and event recognition.

For evaluating the detection, we used 3 annotated sequences: Sequence 1 is a short sequence of 128 frames with just one ground truth object (one group), Sequence 2 has 1373 frames and 9 ground truth objects, and Sequence 3 is 17992 frames long and 25 ground truth objects were annotated.

The criteria used for evaluating the tracking computes the confusion between the detection and the ground truth objects according the formula: $confusion = \frac{1}{\sharp\{DmR\}} \sum_{i \in \{DmR\}} \frac{Max_i}{Total_i}$ with $\{DmR\}$, the set of the detected objects which match the reference objects, Max_i, the maximal number of frames in common between the detected object i and all the objects of the ground truth, and $Total_i$, the number of frames where the detected object i matches a ground truth object. A confusion value close to 1 is significative of a good tracking.

Detection and tracking results are shown in table 1.

	Sequence 1		Sequence 2		Sequence 3	
	S	HD	S	HD	S	HD
True Positives (TP)	72	67	1395	1079	5635	3679
False Positives (FP)	0	0	11	111	1213	642
False Negatives (FN)	6	11	269	585	3686	5642
Precision (global)	1	1	0.99	0.90	0.82	0.85
Sensitivity (global)	0.92	0.84	0.83	0.65	0.60	0.40
Tracking confusion	1	1	1	0.99	0.92	0.96

Table 1. Segmentation (S) and Human Detector (HD) Results

We defined and modeled an ontology of events that can not be shown here due to a lack of space. Figure 4 shows two examples of event detection from this ontology: a group getting off the train based on their trajectory and the context (zones in front of doors). The second example is a group detected as "lively" based on the variation of its size.

4 Conclusion

We presented a complete method for detecting and tracking groups of people in video-surveillance videos and for detecting group behavior or events relative to groups. The proposed method detects people in images using a LBP-based people detector. The detected targets are linked by a Frame-to-Frame tracker before the groups are detected and tracked themselves. Events of interest are encoded in scenarios using a descriptive language and recognized based on properties of

[2] `http://www-sop.inria.fr/teams/pulsar/EvaluationTool/ViSEvAl_Description.html`

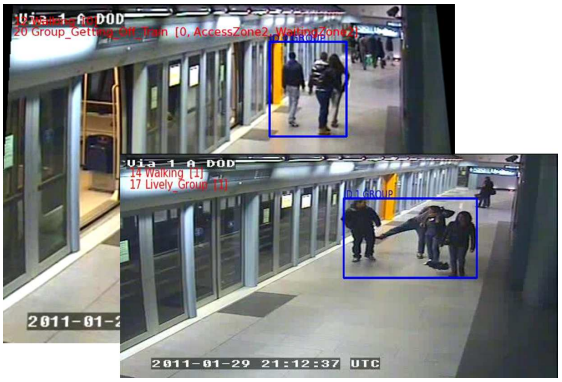


Fig. 4. Example of detected groups and events: a group getting off the train (above) and a group having a lively behavior (below).

groups. Table 1 compares results using two methods for detecting people in the video. The segmentation method (S) uses background subtraction and blob detection. Persons are detected by merging blobs of pixels. The HD method uses the human detector described section 2.1. One can notice that the human detector has less true positives. This is due to the fact that this method only detects persons entirely visible in the view. In case of sequence 3, this method also detects less false positives, it is more accurate in certain cases. As future work, we will combine both methods to detect people and use sensor fusion to merge both results.

Acknowledgment This work was supported partly by the Video-Id, ViCoMo, Vanaheim, and Support projects. However, the views and opinions expressed herein do not necessarily reflect those of the financing institutions.

References

1. M. Albanese, R. Chellappa, V. Moscato, A. Picariello, V.S. Subrahmanian, P. Turaga, and O. Udrea. A constrained probabilistic petri net framework for human activity detection in video. *IEEE Trans. on Multimedia*, 10(6):982–996, 2008.
2. W. Ge, R. T. Collins, and B. Ruback. Automatically detecting the small group structure of a crowd. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8, December 2009.
3. S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80(1):42–56, 2000.
4. C. McPhail and R. T. Wohlstein. Using film to analyze pedestrian behavior. *Sociological Methods & Research*, 10(3):347–375, 1982.
5. T. Ojala, M. Pietikäinen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. ICPR 1994, Jerusalem, Israel. Vol I, 582-585, 1994.
6. G. Overett, L. Petersson, N. Brewer, L. Andersson, and N. Pettersson. A new pedestrian dataset for supervised learning. Intelligent Vehicles Symposium, 2008.
7. V. T. Vu, F. Bremond, and M. Thonnat. Automatic video interpretation: a novel algorithm for temporal scenario recognition. In *IJCAI'03*, 2003.