

Automatic detection of anomalous events for advanced real-time video surveillance

A. Mecocci & M. Pannozzo

Department of Information Engineering, Siena University, Italy

Abstract

This paper introduces the architecture and the implementation details of an automatic real-time video surveillance system, capable of autonomously detecting anomalous events. Nowadays video surveillance systems are not flexible in adapting to different operative scenarios (they only work in well known and structured environments) and generally need human assistance to recognize and tag specific visual events. A new approach is described to overcome these limitations. The proposed system can automatically adapt to different scenarios without human intervention (but the placement of the TV sensors), and automatically learns the "typical" targets behavior in each specific operative environment by means of robust self-learning techniques. The learned knowledge is used by the system to give alerts in an automatic way by analyzing the trajectories of visual targets in the controlled scene. Robustness is obtained through an improved version of the Altruistic Vector Quantization algorithm (AVQ). The modified AVQ autonomously establishes the number of trajectory prototypes, and improves the representativeness of the prototypes themselves. Anomalies are detected if visual trajectories deviate from the "typical" learned prototypes. Standard PCs and TV cameras have been used for the actual implementation, that has been tested in many real indoor and outdoor environments. Currently real time performance has been obtained (15 fps for each camera). A preliminary learning period (about 20 minutes to grant a suitable time interval to learn all the "typical" visual trajectories) is necessary. After that period the system gives automatic alerts about events that do not conform to typical behaviors. The field of view can be changed (by panning the TV camera, for example), then the system relearns the new scenario without any human intervention (no thresholds or other settings) and accurately detects anomalous events.

Keywords: feature analysis, vector quantization, event recognition, anomalies detection.



1 Introduction

Today the increase of terrorist attacks, accidents, robberies, and crimes, urges the development of innovative surveillance systems capable of operating in different contexts, like: parking lots, stores, galleries, buildings, roads. It is evident that the possibility of preventing crimes by taking immediate counter-actions depends on the availability of just-in-time alarms and this is an important development aspect of innovative surveillance systems. Nevertheless, even if the use of video surveillance is already broadly diffused, its basic use is as a "post factum" device i.e., as a legal tool to track possible transgressors and to reconstruct the chain of actions [1].

Moreover, in general, digital technologies are mainly used to transmit compressed streams towards central control rooms, where human operators are in charge of performing vigilance. Unfortunately this approach is not effective and it is error prone due to the human component involved (it is very difficult to maintain a high level of attention for long time periods, particularly when many videos must be controlled in parallel). This is why a suitable video surveillance system must be capable of autonomously detecting anomalous visual events in real-time to immediately attract the attention of the human operators in case of danger.

In this paper we introduce an approach to monitor live video, to raise automatic alarms, and to store digital video records of the ongoing actions. Due to the adopted procedures, the system does not need specific tailoring even if the inspection scenario changes and, due to the self-learning capabilities, it is possible to adapt it to a vast range of operational conditions without relevant human-based settings. The possibility of automatically selecting the appropriate descriptive features and of automatically detecting visual events of interest, is still an open research theme, even if an extensive literature has been devoted to the interpretation of human movements (see [2] for a detailed review of Human Motion Analysis) and many approaches have been proposed for the detection and tracking of target movements. In [3] the authors describe a system that generates a database of visual events detected in real-time. Unfortunately such a system deeply relies on humans that must define sophisticated queries to describe complex events and target interactions. In [4], event recognition can be performed only about known scenes where the targets move in predefined directions; this fact limits its practical applications. In [5], Hidden Markov Models are used to describe the visual targets behaviour. Nevertheless, a suitable state-space needs to be superimposed to the scene. This makes it difficult to adapt the system when different scenes must be controlled (a different state-space must be manually defined for each different scenario). Recently the extension of the CONDENSED algorithm proposed in [6], has been used to reconstruct trajectories by analyzing the history of past object states (that describe the motion path of each target). Also in this case, highly specialized personnel must setup the system in particular to suitably define the number of hidden states necessary to describe the events. Another approach, [7], uses target silhouettes to track and cluster pedestrians in outdoor environments. The system



runs in real-time, but cannot suitably manage temporal occlusions that often occur in real applications.

The previous systems cannot easily deal with unknown scenes and they need the presence of an operator, at least in an initial training phase, to setup the parameters needed for the correct recognition of anomalous events. To get over these problems, we re-worked the original idea of Johnson [8], that used the Altruistic Vector Quantization algorithm to acquire and model the targets motion in the scene. Our system basically uses the same algorithm, but two important improvements have been added. The first one improves the accuracy during the data learning phase; the second one solves a crucial problem of the original algorithm, namely: it allows the automatic definition of the number of trajectory prototypes to be used.

2 The system architecture

In general, a visual surveillance system must perform the tasks illustrated in the following Figure 1. In this paper we concentrate on the implementation of the *Anomaly Detection System* (ADS) that plays a fundamental role as regards the performance and robustness of the whole chain.

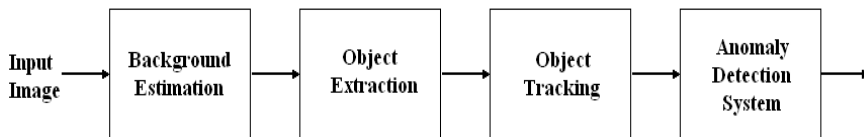


Figure 1: Architecture of a generic video surveillance system.

The proposed Anomaly Detection System takes as input the trajectories of the moving objects in the field of view. Such trajectories are described by a series of Instant Positions (each Instant Position is described through a vector that codes the Centroid Position, the Speed and the Dispersiveness of each moving Object in the scene). The accumulated Instant Positions (*IPs*) are analyzed through the AVQ algorithm and a set of Prototypes of State (*PSs*) is obtained that approximates the spatial distribution of the positions of the ensemble of moving objects observed in the scene. Subsequently the same AVQ algorithm is used to learn the so called Generalized Trajectory (*GT*) of each moving object (the *GT* is the set of the *IPs* related to the same target) and to construct a set of Prototype Trajectories (*PTs*) that are used to detect the anomalies. The following three major modules are involved:

Feature Extraction. This module receives the data about the objects extracted by the acquired sequence and computes the centroid, the displacement, and the dispersivity of each object in each frame. The corresponding information is stored in a set of *IPs*, while subsequent *IPs*, corresponding to the same moving

target, are stored into a *GT*. During the Learning phase, all the trajectories are temporarily stored in a Buffer.

Prototypes Learning. This module operates in two steps. The modified AVQ algorithm (see below) is initially applied to approximate the spatial pdf of the *IPs* by means of a certain set of *PSs*. The *PSs* are then used to define the so called Normalized Trajectories (*NTs*, see below). During the second step another AVQ algorithm generates the *PTs* that approximate the pdf of the *NTs*.

Anomaly Detection. This module decides when to raise an alarm. When a new object appears in the scene, a new trajectory is generated and the extracted features are used to obtain the corresponding *NT*. An alarm is raised in two cases: first when one *IP* exceeds a predefined distance from the whole set of *PSs*; second when the *NT* exceeds a predefined distance from the whole set of *PTs* (in both cases the Euclidean metrics is used).

2.1 Feature extraction

Making reference to Figure 1, the Object Extraction module detects all moving objects in the scene by extracting the corresponding blobs. Each Blob is described by a vector of the following type:

$$Blob_i^j = \{(x_i^j, y_i^j), a_{i_1}^j, a_{i_2}^j, \dots, a_{i_p}^j, (C_{x_i}^j, C_{y_i}^j)\} \quad (1)$$

where $Blob_i^j$ is the *i*-th blob at frame *j*, (x_i^j, y_i^j) is the start point of its border, $a_{i_1}^j, a_{i_2}^j, \dots, a_{i_p}^j$ is the Freeman's code of its border, and $(C_{x_i}^j, C_{y_i}^j)$ are the centroid coordinates.

All the blobs related to a certain moving target are grouped and stored into another structure called an Agent. The Agent is the basic information used to describe visual events in the scenes:

$$Agent_j = (Blob_0^j, Blob_1^j, \dots, Blob_{d_j-1}^j) \quad (2)$$

where the index d_j is the temporal duration (in frames), of the object *j*, $Blob_0^j$ is the first Blob of the object *j*, while $Blob_{d_j-1}^j$ is the last Blob of the object (before the object *j* leaves the scene). The set of centroids of each Agent is used to define the trajectory of the corresponding target.

Even if centroid tracking is very used in practical applications, the resulting description is very poor and unable to capture the nature of each visual event. This is why the Feature Extraction modules use the data of each Agent to evaluate: the shape, the centroid position, the speed changes, and the directional information of the corresponding target. These data are used to define the *GT* of the target itself. Be $\tilde{T}_j$ the *GT* corresponding to the *j*-th Agent, its formal definition is given by the following formula:

$$\tilde{T}_j = \{\vec{f}_0^j, \vec{f}_1^j, \dots, \vec{f}_{d_j-1}^j\} \quad (3)$$

where $\vec{f}_i^j$ denotes a feature vector comprising: the centroid position ($\vec{f}_i^j(0)$ is the x coordinate while $\vec{f}_i^j(1)$ is the y coordinate), the instant speed ($\vec{f}_i^j(2)$ is the horizontal component while $\vec{f}_i^j(3)$ is the vertical component), and the dispersiveness (that is the Perimeter/Area ratio, $\vec{f}_i^j(4)$) of the i-th Blob of the Agent j. The feature vectors in $\tilde{T}_j$ are normalized so that $f_i^j \in [0,1]^5$. The normalization avoids eventual bias due to the different range of values of the vector components.

2.2 Instant positions and prototypes learning

The ADS module uses the modified AVQ algorithm to approximate the spatial distribution of the trajectory points by clustering each group of nearby trajectories. During the clustering a suitable set of hyper-spheres is defined whose parameters (centre and radius) depend on the local density and arrangement of trajectory points. The centre of each hyper-sphere is called *PS* and the radius is called “Radius of Convergence”. When the trajectory of a certain target lies within the convergence region of some *PSs*, it can be considered typical. On the contrary, when the trajectory points are outside of the convergence region of all *PSs*, it can be considered anomalous.

3 The AVQ algorithm

To describe the typical behaviour of the objects that are moving in the scene, it is necessary to observe the scene for a certain time, and to extract a suitable description of their motion. To this end, the proposed system learns, as previously described, some *PSs* that approximate the pdf both of the *IPs*, and of the *GTs*. The two sets of calculated prototypes will be called, respectively: *PSs* and *PTs*.

3.1 Normalization of the generalized trajectories

The *GTs*, typically have different durations, since each target takes its own time to cross the field of view. This fact is a big problem for the original AVQ algorithm. In fact, the AVQ needs input vectors of the same length. To solve such problem a normalization step is introduced to grant the same length for each *GT*. The normalization is performed as follows: the whole *GT* is analyzed and for each component $\vec{f}_k^j$ the nearest *PS* is estimated by means of the following equation:

$$\|\vec{f}_k^j - \vec{p}_{S_r}\| = \min_{1 \leq i \leq NPS} \|\vec{f}_k^j - \vec{p}_{S_i}\| \quad (4)$$



where $\vec{p}_{S_i}$ represents the i -h PS . An Activation Function $a_i(t)$ is associated to each $\vec{p}_{S_i}$, then the NT is defined by concatenating the different values of the Activation Function of each PS , as follows:

$$\vec{T}_j = \{a_0(t), a_1(t), \dots, a_{NPS-1}(t)\} \quad (5)$$

The function $a_i(t)$ is updated, each time a new component of the GT is analyzed, according to the following formula:

$$\begin{aligned} a_i(t+1) &= O_i(t) & \text{if } O_i(t) > a_i(t) \\ a_i(t+1) &= \gamma \cdot a_i(t) & \text{if } O_i(t) \leq a_i(t) \end{aligned} \quad (6)$$

The index $\gamma \in [0, 1]$ is said Decadence Constant and represents the memory of each activation. The exit value $O_i(t)$ is computed in the following way:

$$O_i(t) = 1 - \frac{\|\vec{f}_k^j - \vec{p}_{S_v}\|}{d_{MAX}} \quad (7)$$

The distance d_{MAX} is the maximum distance observed during the learning phase.

The exit factor, $O_i(t)$, guarantees that each vector component is in the interval $[0, 1]$, so each NT belongs to the Cartesian product of NPS real intervals $[0, 1]$, i.e. $\vec{T}_j \in [0, 1]^{NPS}$. The activation level of each PS preserves memory of the temporal evolution of each GT .

3.2 Improvements to the AVQ algorithm

The AVQ algorithm has been claimed to be effective and robust, nevertheless major operative difficulties have been noticed during practical experiments in real operative environments. In particular, in many cases the prototypes obtained by the classic AVQ algorithm, tend to lie in completely wrong places with respect to true trajectory points.

3.2.1 Modified updating rule

To solve the problem, we propose a modification of the original AVQ algorithm that changes the strategy used to update the decay factors. The original AVQ moves the Winning Prototype ($\vec{m}_v(t)$) towards the Observed Vector ($\vec{X}_j(t)$) according to the following formula:

$$\vec{m}_v(t+1) = \vec{m}_v(t) + \alpha(t)[\vec{X}_j(t) - \vec{m}_v(t)] \quad (8)$$

The decay factor $\alpha(t)$ of the winning prototype has the following form:

$$\alpha(t) = 1 - 0.99 \cdot \left(\frac{2 \cdot t}{N}\right) \quad \text{if } 0 \leq t \leq \frac{N}{2}$$

$$\alpha(t) = 0.01 \quad \text{if } \frac{N}{2} \leq t \leq N \quad (9)$$

where N is the maximum number of allowed iterations, and t is the current iteration number. By referring to a certain NT it is easy to verify that non null components are only present in those positions interested by movements. All the other positions (static areas in the scene) give rise to zero values. If the original AVQ formulation is used, all the components of the winning prototype are updated even if they relate to areas not affected by any target movement (components with zero value). As a consequence, the winning prototype tends to move towards erroneous zones. This is a major drawback of the AVQ algorithm when applied to target tracking in general environments. In the revised updating method we propose, the components of the PTs are updated only if they relate to non null components of the current trajectory. With this modification the results dramatically improve.

3.2.2 Automatic definition of the number of prototypes

Another problem intrinsic to the original AVQ algorithm, is the fixed number of Prototypes that must be specified *a priori* during the learning phase. Even if this is not a big problem in the case of PSs (the number is proportional to the image resolution), it turns out to be a major problem in the case of PTs . In fact, the number of PTs depends on the targets behaviour and cannot be predetermined. Again, this parameter is very important because if it is too large or too small, the learning process gives poor performances. The degradation is due to the very nature of the AVQ algorithm. When a too small number of prototypes are specified, they are not capable of distinguishing between trajectories when they intersect each other. This happens even if the targets follow different directions. The common span will be described by the same prototype without any discrimination. This is not acceptable for practical visual inspection applications, where we want to discriminate the spatio-temporal behaviour (that also involves the motion direction). To solve this problem it is necessary to automatically define the correct number of PTs . The proposed approach extends the original AVQ formulation by adding directional information obtained during the learning phase. For each NT , say $\bar{T}_j \in [0,1]^{NPS}$, we define a set of Generalized Directions, D_G^j , as follows:

$$D_G^j = \{R_I^j, D_M^j, R_U^j\} \quad (10)$$

where:

- R_I^j : represents the start region in the scene, where Trajectory j begins;
- D_M^j : is the mean direction of the Trajectory j in the scene;
- R_U^j : is the exit region in the scene where Trajectory j ends.

The Generalized Directions are pre-analyzed to define the number of different directions that can be detected in the environment currently monitored. This number is used to fix the amount of prototypes to be used by the AVQ algorithm during the subsequent learning steps. In this way the number of PTs automatically follows from the conformation of the currently analyzed scene.

4 Anomaly detection

The Anomaly Detection is performed by associating a Convergence Region (see paragraph 2.2) to each *PS* and *PT* during the learning phase. The Convergence Region is based on the Mean distance and Variance of the Prototype from the true target positions. For each component of the Observed Vectors $\vec{V}_i$, the nearest Prototype $\vec{p}_v$ is computed (during the learning phase) by an Euclidean metric:

$$\|\vec{V}_i - \vec{p}_v\| = \min_{1 \leq j \leq NP} \|\vec{V}_i - \vec{p}_j\| \quad (11)$$

the Mean Distance from the winning prototype is updated by:

$$\mu_v = \frac{\sum_{j=1}^{N_v} \|\vec{V}_i - \vec{p}_v\|_j}{N_v} \quad (12)$$

The index N_v represents how many times the considered prototype is the winner. The Variance of the Mean Distance is computed by using the following formula:

$$\sigma_v = \sqrt{\frac{\sum_{j=1}^{N_v} \|\vec{V}_i - \vec{p}_v\|_j^2}{N_v} - (\mu_v)^2} \quad (13)$$

The Mean Distance and the Variance of the Mean Distance are used to define the corresponding Convergence Region for each Prototype. The Convergence Region is a hyper-sphere centred in $\vec{p}_v$, with a radius of $\mu_v + 3 \cdot \sigma_v$. After each Convergence Region has been defined, the Anomaly Detection module analyzes the *GT* of each Agent and its *NT*. For each Object the *IPs* and the *NT* are analyzed. If the *IP* lie outside the Convergence Regions associated to the corresponding nearest prototypes, an Anomaly State is triggered. A similar process is used to verify the *NT* to decide if a Trajectory Anomaly State must be raised. The *IPs* are considered of primary importance, so if an Anomaly State is triggered, the *NT* analysis is skipped and a Global Alarm is given.

5 Experimental results

Currently the system runs in real-time (15 fps, standard CIF frames) under Windows 2000 operating system installed in a standard PC (AMD Duron 1.3Ghz, 256Mb Ddr). The performance has been extensively analyzed in



different operating condition (both inside and outside environments). Figure 2, for example, shows the anomaly detection results related to an outdoor and to an indoor environment. The outdoor environment involves moving cars at the exit gate of an highway. A total number of 161 trajectories have been used during the learning phase. After the algorithm convergence, 10 *PTs* have been automatically identified. In the figure the car trajectories in green are considered as safe. In the bottom left image the car trajectory remains green until it reaches the gate (the trajectory is normal, in fact), but turns red as soon as the car moves backward due to unknown reasons (the trajectory is now completely forbidden and very dangerous). The indoor environment relates to part of a shopping gallery. Typical trajectories are represented in green, while anomalous trajectories are represented in yellow or in red according to the gravity of the anomaly itself (evaluated by the system). 200 trajectories have been used during the Learning Phase by the modified AVQ Algorithm (1.000.000 iterations) giving 8 different *PTs* in a fully automatic way (no parameter adjustment).

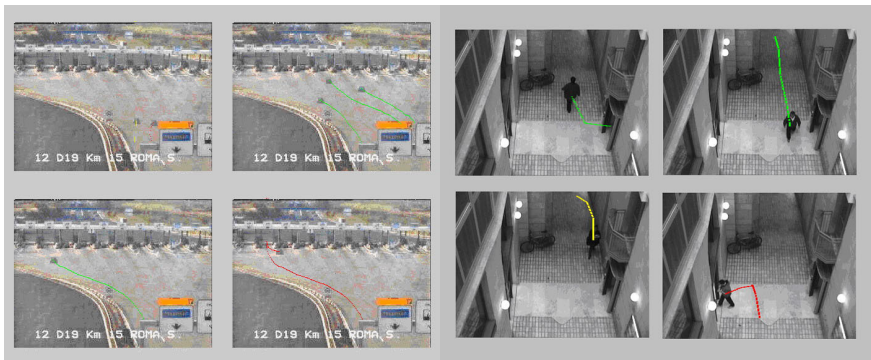


Figure 2: Real-time results obtained in indoor and outdoor environments.

It must be emphasized that the detection system works in complete autonomy: just decide the scene, start the learning and then begin the anomaly detection and logging. No parameters must be set.

6 Conclusions

The system presented in this paper faces the problem of real-time automatic detection of anomalous visual behaviours in complex, unstructured scenes. A low-level statistic approach has been used to automatically acquire the models of moving targets by simply watching at the scene for a certain time. The Learning uses a modification of the AVQ algorithm to make it applicable to real operative scenarios. A robust and stable behaviour has been obtained during the Detection Phase, due to the statistical basis of the proposed approach. Very good classifications have been produced without regards to the operative environment. The system has been implemented by means of standard PCs and TV cameras,

and has been tested in many real outdoor contexts at different times (night and day). It is capable of running in real-time (15 fps for each camera) on general purpose HW.

References

- [1] IEEE Transactions on Pattern Analysis and Machine Intelligence “*Special Section on Video Surveillance*”: August 2000.
- [2] J. K. Aggarwal and Q. Cai, “*Human Motion Analysis: A Review*”, Journal of Computer Vision and Image Understanding, Vol. 73, No. 3, pp. 428-440, March 1999.
- [3] D. Greenhill, P. Remagnino, and G. A. Jones, “*Vigilant. Content-Querying of Video Surveillance Streams*”. Video-Based Surveillance Systems, Computer Vision and Distributed Processing, February 2002.
- [4] Howarth R. and Buxton H. “Analogical representation of spatial events for understanding traffic behaviour”. In Neumann B., editor, 10th European Conference on Artificial Intelligence, pages 785–789. John Wiley & Sons, 1992.
- [5] Ch. Bregler, “*Learning and Recognizing Human Dynamics in Video Sequences*”, Computer Vision and Pattern Recognition, pp. 568-574, 1997.
- [6] E. B. Koller-Meier and L. V. Gool. “*Modelling and Recognition of Human Actions Using a Stochastic Approach*”. Communication Technology Lab, Image science Swiss Federal Institut of Technology.
- [7] J. Segen and S. Pingali, “*A Camera-Based System for Tracking People in Real Time*”. 13th International Conference on Pattern Recognition, Vol. 3, Track C, pp. 63-67, 1996.
- [8] Johnson N, “*Learning Object Behaviour Models*”, PhD Thesis, Leeds, England, September 1998.