Video-based detection of specific events in public transport networks

L. Khoudour*, D. Aubert*, S. Velastin**, V. Leung** and J. Orwell**

* Institut National De Recherche Sur Les Transports Et Leur Sécurité (INRETS)
  (louahdi.khoudour@inrets.fr)
** Digital Imaging Research Centre, Kingston University, United Kingdom
  (sergio.velastin@kingston.ac.uk)

RÉSUMÉ.
ABSTRACT. In recent years, image-processing solutions have been to automatically detect incidents and make measurements on video images from CCTV cameras, relieving the staff in control rooms of much of the difficulty in finding out where events that require attention are occurring. In this paper we illustrate some of the results obtained with on-site experiments carried out as part of projects funded by the European Commission and some additional experimentation carried out outside the project. We present here details on the systems, algorithms and their performance evaluation. Four different specific events in transport networks are described: abnormal stationarity, queuing, intrusion detection and loitering. Promising results are obtained from onsite experimentation.

MOTS-CLÉS : KEYWORDS: Image processing, events, public transport, stationarity, queuing, loitering, intrusion.

1. Introduction

Public transport operators are facing increasing demands for more efficiency and better security from the general public as well as from the
governments. An important part of the efforts deployed to meet these demands is the ever-increasing use of video surveillance cameras throughout their networks, in order to monitor passenger flows, enable staff to be informed of possible congestion, and detect incidents without delay. A major difficulty of this approach, however, is the very large number of cameras required to effectively monitor even a comparatively small network. Added to the cost of the cameras themselves, the cost and complexity of the required wiring and maintenance, plus the sheer impossibility of watching all the images at the same time, make such a system increasingly ineffective as the number of cameras increases.

In recent years, image-processing solutions have been proposed to automatically detect incidents and make measurements on video images from CCTV cameras, relieving the staff in control rooms of much of the difficulty in finding out where interesting events are happening. However, the need remains to bring the latter to a centralized computer located in a technical room, which implies the need to place significant amounts of video cabling thus increasing cost, complexity, and decreasing the adaptability of the video system.

In the type of environments considered here, the term surveillance is normally associated with visual monitoring. Thus, a significant amount of technical work took place to investigate and develop robust vision systems able to detect situations of interest for public transport operators. These situations include overcrowding, congestion (too many people who are not moving), stationary people (e.g. indicative of loitering, drug dealing, buskers, etc.), abandoned packages, long queues, etc. The overall approach has been to design systems where video processing takes place at or near the camera so that only when an event of interest is detected (locally), an alarm is raised by sending a message with event information to a control room, for instance. Such an approach is sometimes referred to as “intelligent cameras” or “smart cameras”. A significant challenge has been that the vision algorithms need to work on what are mostly cluttered environments.

INRETS and Kingston University have devised and tested in real-life situations an architecture to address the problems mentioned above [1, 2, 3]. The general idea is to deal with the need of making full-resolution images at real-time frames available for processing (to im-
prove detection) by deploying the processing power close to the cameras themselves, and sending only meaningful images through a general-purpose network to the control room. Until recently, computers and video grabbers were much too expensive to even dream of having multiple computers spread all over the network, but costs are decreasing at a steady pace, and it is becoming realistic to believe that this will be commonplace soon. Existing technologies already allow, although still at a relatively high cost, to realize such a working network.

In this paper we illustrate some of the results obtained with on-site experiments carried out as part of our research projects. Four different specific event detections in a public transport context are proposed. Theory and main results are presented.

2. Video-based detection of events

INRETS and Kingston University have developed a multi-camera vision system specified to meet key requirements of security and monitoring tasks in a transport network. The diversity of the tasks at hand implies significant processing power and highly versatile configuration options. These requirements are best met by a modular architecture based on distributed processing, considering in particular the following points:

- Low-level information processing must be local to the cameras.

- The cameras must be capable of operating either alone or interconnected in a network including an array of sensors and high-level processing stations (PC workstations for instance) for the interpretation, display and archiving of the collected data.

- The architecture chosen must allow the integration in the same network of different types of sensors fitted with a standard network interface [3].

- The cameras should be configurable remotely, and should be able to transmit images across the network on request, as well as high-level data resulting from local processing.
These requirements are best served by using an “intelligent camera” module as the main building block of the system. In this context, the low-level image processing is performed directly at the camera end. High-level information resulting from the on-board processing is then sent across a network interface to a local substation or a central processing computer, depending on the specifics of the site. This reduces the cost and complexity of the infrastructure, by reducing drastically the amount of data to be sent through the cabling. Since most of the time, only low bandwidth information is transmitted across the network and most of the processing power is distributed, image retrieval and analysis is performed locally to each camera. The system nevertheless allows the transmission of individual images to the control station for the operators to look at and, for instance, evaluate the actual reasons why an incident has been raised by a camera (Figure 1).

![Image Processor](image.png)

**Figure 1** – An integrated image processor similar to those used in the local camera network.

The whole system consists of several camera-processor heads, linked via an Ethernet network to a central Data Display and Analysis Station.

In this paper we illustrate some of the results obtained with on-site experiments carried out as part of two European projects and some additional experimentations carried out outside the projects.

### 3. Detection of abnormal stationarity

The automatic detection of what we call “abnormal stationarity” is very difficult to achieve. This specific event corresponds to either people
or to objects. A stationary person in a transport network is something that might be common and not necessarily associated to a potentially dangerous situation [4, 5]. However, in some places (e.g. in narrow and/or isolated corridors) the presence of stationary people could be due to drug dealers, a thief waiting for a potential victim or a person requiring assistance. In the case of objects, the most common event is that of objects that have been left unattended. In the public transport network context, this type of detection is required to react quickly to events potentially impacting user security. It must be emphasized here that the abnormal stationarity detector has one critical parameter, which is the time during which stationarity remains “normal”. For instance, it is considered safe for air passengers to stop and check their plane ticket. A passenger doing so should not raise an alarm. On the other hand, stopping for several minutes is most likely to indicate an unattended luggage, someone who has collapsed, or other situation requiring a response. This time parameter is supposed to be set once and for all, and not changed during normal operation.

In our context, “stationarity” for a person does not always mean to be “completely motionless”. A person standing still does move their legs, arms, etc. Thus, in the following sections “stationarity” means a person or an object located in a small area in the image during a given minimum period of time.

The fact that a person may move adds complexity to the detection process. Other difficulties depend on the site and the network. For instance, the camera view angle may be low (which is almost always the case in metro stations), causing occlusion. If the site is crowded, the “stationary” person or object will be partially or completely occluded from time to time, making its detection and the duration of its stationarity more difficult. The lighting conditions may affect the perception and the detection. In fact, the method needs to be robust to contrast changes and to shadows. Its needs to deal with occlusion and to take into account the motion of a “stationary” person.

Broadly speaking, the algorithm involves averaging the motions in the passenger flow, and detects whether some part of the image, while not being part of the background, remains motionless for more than a user-defined time threshold (usually set to two or three minutes de-
pending on the context as defined by experienced operators). Then, additional modules are used to detect excessively long stationarities. In the following section, we provide some details on the strategy we have adopted to detect the stationary people or objects.

3.1. Characterisation of stationarity

By stationarity, we imply somebody or something present in the same area for a given period of time and probably moving arms and legs (Figure 2).

![Figure 2](image.jpg)

Figure 2 – Characterisation of a stationary person in a corridor.

A first idea to achieve detection would be to perform shape recognition and to detect if this shape remains in the same area for a given period. The benefit of the recognition process is the possibility to discriminate object classes. However, the recognition is quite difficult to perform under frequent occlusions, when the view angle or the object aspect changes. This approach does not seem realistic, at least in the short term and during crowded conditions.

The selected solution consists in detecting changes at the level of individual pixels, determining those “motionless” and finally grouping together connected “motionless” pixels to get stationary shapes. In this case, it is not possible to determine the object class (person or bag, or something else) [6, 7, 8, 9]. However, the result is obtained regardless of changes in the aspect of the object and of the view angle. Another
advantage is the possibility of detecting a motionless object, even if only a small part of this object is seen (because of multiple occlusions).

3.2. Change detection algorithm

The “change detector” constitutes the key element of the overall algorithm proposed for detecting stationary objects. We assume that the camera is observing a fixed scene, with the presence of objects moving, appearing and disappearing. Hereafter, the term “background” designates the fixed components of the scene, whereas the other components are considered as “events” in the scene. The purpose of the “change detector” is to decide whether each pixel belongs to the background or to the events.

In order to avoid disturbances caused by global illumination changes, one of the original ideas of our change detector is to separate geometrical information from contrast information.

Local measures based on level lines.

A local measurement based on the level-lines geometry preserves the robustness of image representation. We compute the direction of the half-tangent of each level line passing through each pixel. As a consequence, several orientations may be simultaneously present for a given pixel. All of these orientations are then quantified for easier storage and use.

3.3. Reference data building and updating process

The orientations of the local level lines are used to build and update an image of the background, used as the reference. First, we compute, for each pixel, the occurrence frequency of each orientation over a temporal sliding window:

\[
F_{t \leq T} (P, \theta_k) = mF_{t \leq T-1} (P, \theta_k) + (1 - m) f_t (P, \theta_k)
\]

(1)

with \( f_t (P, \theta_k) \) a binary value indicating whether the direction \( \theta_k \) exists at pixel \( P \) at time \( t \), \( F_{t \leq T} (P, \theta_k) \) a cumulative function which contains
the number of times the direction $\theta_k$ exists for $P$ over a time window $T$, and $m = T / (T + 1)$.

A direction with a large number ($T_o$) of occurrences is considered to belong to the reference $R$.

$$R_{t,T,T_0} = 1 \text{ if } F_{t \leq T} (P, \theta_k) \geq T_0 \text{ and } R_{t,T,T_0} = 0 \text{ otherwise}$$

(2)

3.4. Change detection

Given a direction $\theta$ at a pixel $P$, we verify if this direction occurs in the reference $R_{t,T,T_0}$, up to its accuracy. If not, the pixel belongs to an event:

If $\exists R_{t,T,T_0} (P, \theta_k) = 1$ and $C (P) = 1$

otherwise $C (P) = 0$

(3)

The time window ($T$) and the occurrence threshold ($T_o$) are chosen experimentally, depending on the context (average speed of motion in the site of experimentation, frame rate used, etc.). However, given a window time ($T$), a lower and upper bounds exist for ($T_o$) to ensure that no orientation, in the reference, is caused by noise. By adjusting $T$, we can detect changes that occur over various time ranges.

3.5. Stationarity detection algorithm

The extracted level-lines could be classified into one of the following categories: those that belong to the scene background or those that correspond to moving objects, or to stationary objects.

The system uses the “presence duration” to discriminate between these two categories. The background is naturally assumed to remain unchanged for a long period of time. Conversely, the moving objects will not yield stable configurations even over short periods of time. Between these two extremes, stationarity is characterized by objects that remain at approximately the same place over an intermediate period of time (the threshold). This set-up then involves the use of the change detector with two time periods: a short one to detect moving objects...
Detection of short-term change ("moving parts")

By applying the change detector with a short-term reference $R_{\text{short}}$, areas in the image containing moving objects are detected. The result is a binary image representing the short-term changes, referred to as the “motion map” ($M_t$):

$$
\text{If } \exists \theta_k / f_t (P, \theta_k) = 1 \text{ and } R_{\text{short}} (P, \theta_k) = 0 \text{ then } M_t (P) = 1,
\text{otherwise } M_t (P) = 0
$$

(4)

Detection of long-term change ("events")

By using the same process with a long-term reference $R_{\text{long}}$, “events” areas are highlighted ($N_t$):

$$
\text{If } \exists \theta_k / f_t (P, \theta_k) = 1 \text{ and } R_{\text{long}} (P, \theta_k) = 0 \text{ then } N_t (P) = 1,
\text{otherwise } N_t (P) = 0
$$

(5)

Detection of the stationary areas

By removing the moving parts from the events, only the people and objects remaining stationary for at least a short term duration are kept ($S_t$):

$$
S_t (P) = N_t (P) - M_t (P)
$$

(6)

Estimation of the stationary duration.

Adding the occurrences of the stationary state at each pixel over time enables us to estimate the “Stop Duration”, called here $SD$. So, the $IOD$ (Index Of Duration) can be expressed as follows:

$$
\begin{array}{ll}
\{ & IOD (P)_t = (1 - \alpha) \times IOD (P)_{t-1} + \alpha & \text{if } S_t (P) = 1 \\
& IOD (P)_t = (1 - \alpha) \times IOD (P)_{t-1} & \text{otherwise}
\end{array}
$$

(7)

So, for a motionless pixel $P$

$$
IOD (P)_t = 1 - (1 - \alpha)^t, 0 \leq IOD \leq 1
$$

(8)
where \( t \) is the number of previously observed images, and \( \alpha \) the value controlling the evolution speed of “average”. \( \alpha \) is determined so that “IOD” is equal to a threshold \( T_d \) when the target maximal stop duration is reached. For instance, \( \alpha = 1 - (1 - 0.7)^{1/(25 \times 360)} = 0.00026 \) considering a target stop duration of 3 minutes, \( T_d = 70\% \) and a 25 frames per second processing frame rate.

\( T_d \) is chosen so that any detection is stable (the “IOD” varies from \( T_d \) to 1 without any loss of detection) and so that a detection disappears as quickly as possible when the stationary person/object leaves the scene (the detection ends when “IOD” becomes lower than \( T_d \) again).

In fact, to get a stable “IOD”, its estimation is frozen for a while when the area is occluded. In the system, the occlusion is characterized by the motion map \( (M_t) \), since it is generally caused by people passing in front of the stationary person/object. However, due to this freezing process, the estimated stop duration may be shorter than the real duration. Thus, to get the right “IOD”, \( \alpha \) may be increased according to the number of images for which the computation was not done \( (t_{occlusion}) \).

Therefore, \( \alpha \) varies in time and space:

\[
\begin{align*}
    t_{occlusion}(t, P) &= t_{occlusion}(t - 1, P) + M_t(P) \\
    \alpha(p) t &= 1 - (1 - T_d)^{1/t'}
\end{align*}
\]

(9)

where \( t' = t - \min(t_{occlusion}(t, P), 50\%t) \).

This definition includes a threshold on the occlusion rate up to 50\% of the time. Indeed, for a larger occlusion duration, the number of images over which the stationarity is observed would be too low, which may yield false detection. As a consequence, there is a delay when the occlusion rate is higher than 50\%.

**Detection of the target stationarities.**

Each pixel with a stationarity duration which exceeds a fixed stop duration is marked. Connected pixels are then grouped together to form a bounding box around the stationary object. Thus, the described system is able to detect stationarity regardless of the changes in the person/object position inside a small area.

To summarise, in this algorithm there are the main steps: from the initial images, a difference between the short-term change detections
and the long-term change detections provides the stationary areas. Then, the stationary duration updating is carried out. After that, the stationary analysis consists in comparing the duration to a given threshold and permits the system to send an alarm or not. We must take care here regarding the stop duration threshold. It depends significantly on the place where the detection is made. It should be at least equal to 2 or 3 minutes to correspond to a valid event.

3.6. Some Results

Our algorithm has been tested on several real-life situation data sets in the context of subways and airports. The first data sets correspond to stationary cases collated in the Gare de Lyon metro station and Newcastle airport, in the framework of the EU PRISMATICA project [1]. In order to assess the system, we manually recorded all stationarity events. Results obtained by the system were then compared with the recorded data. Table 1 below summarises the comparison results:

Tableau 1 – NUMERICAL RESULTS OBTAINED ON STATIONARITY IN GARE DE LYON AND NEWCASTLE AIRPORT

<table>
<thead>
<tr>
<th>Number of stationary situations</th>
<th>True positives</th>
<th>False negatives</th>
<th>False positives</th>
<th>Detection delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>436</td>
<td>427 (98%)</td>
<td>9 (2%)</td>
<td>0</td>
<td>+/- 10 sec</td>
</tr>
</tbody>
</table>

Figure 3 provides illustrations of stationary cases at Newcastle airport: the first case represents a standing person in a corridor; the second case is an object left with very low contrast between it and the background; it is also correctly detected. The second image shows a black bag left by its owner two minutes ago (a user-selectable threshold). At this point, the function has gradually accumulated the stationary time of this object, until the threshold was reached to raise an alarm. This alarm remains on for about 5 minutes or more (another selectable threshold), after which the software considers the bag to be part of the background, and the alarm stops. This is, however, not a disadvantage since the alarm
has been raised, and for a normally sufficient time for the operation staff to notice and process it.

Figure 3 – Illustration of specific cases of stationarities.

The third case is a person sitting just beside big windows which results in difficult illumination conditions to deal with. This trial showed that very poor imaging conditions caused, for instance, by heavy backlighting, do not prevent the system from working correctly.

Other trials were carried out with people, staying motionless while other passengers were passing in front of them. The behaviour of the system remained the same as in the previous case, although the occlusions of the “offender” by other people did somewhat increase the time before raising an alarm. The stationarity is, however, kept in memory even in this case, so an alarm was still raised in every case.

As observed from these results, the system is able to deal efficiently with this detection problem. The non-detections can be explained by very low contrast, a stationarity duration just above the threshold, or a very high occlusion rate (> 90%).
The second dataset on which we have tested our system comes from the Magenta station in Paris in the SNCF (French railways) network. Parameters of the algorithm (such as stationarity duration, the occlusion rate, etc,) were modified slightly to adapt them to the new environment.

Table 2 provides the overall results on this new data set.

Tableau 2 – NUMERICAL RESULTS OBTAINED ON STATIONARITY IN MAGENTA STATION AT PARIS

<table>
<thead>
<tr>
<th>Number of stationary situations</th>
<th>True positives</th>
<th>False negatives</th>
<th>False positives</th>
<th>Detection delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>120</td>
<td>115 (96%)</td>
<td>4 (3%)</td>
<td>1 (0.08%)</td>
<td>+/-5 sec</td>
</tr>
</tbody>
</table>

Figure 4 represents examples of detections of stationary persons in this dataset. These three examples are chosen to show: the low contrast for the first one (upper-left), the illumination conditions which are very difficult for the second case in which the person to detect is quite far in the image (upper-right), the illumination and the presence of shadows for the third case (lower). For each case, a display of a red arrow indicates the location of the stationary area. For the first case, a person is waiting in front of escalators. The second case corresponds to a woman making a phone call. The third case corresponds to a person waiting for their train. These results are very promising.

4. Queue-length measurement

4.1. Introduction

This event detection functionality uses a camera overlooking the counters with a very wide angle lens, and can measure simultaneously the length of several queues, using a user-defined position of the start of each queue (this is done once and for all). It can then send these lengths to the supervising PC, for immediate display and/or to a log file for later use. Optionally, a length threshold can be defined by the user to raise an alarm when one or more queues become too long. This function was
developed and evaluated in the framework of the PRISMATICA project and also on additional data provided by the French Railways (SNCF).

Information to be qualified is the estimation of both the queue-length and the time spent in a queue. This information is useful for at least three purposes:

- To increase the comfort of passengers by providing them with information about the average waiting time.

- To obtain statistics and, thus, an estimate of crowd management efficiency. Such statistics may be used, \textit{a posteriori}, to improve the site design and layout, to plan employee working shifts in a more suitable way, to raise the efficiency of ticket offices, cash dispensers, check-in desks and customs desks.

- To check continuously the effectiveness of crowd management, and react immediately to any disturbances.

Two types of queue may be encountered [4, 5]:

Figure 4 – Examples of stationary detections in Magenta station at Paris.
One type corresponds to a moving flow of people, all walking in the same direction at roughly the same speed. It is the case, for instance, at stadium entrances. This type of queue may be characterized by a motion direction and the compactness of people.

The second type is characterized by a group of people, standing generally in line (but not always), motionless for a few seconds, then walking in the same direction for a small distance, waiting again and so on until they leave the queue. It starts in specific areas, such as places close to ticket offices, cash dispensers, registrations desks and boarding gates.

The developed system deals only with the second type. It uses the measurement of stationarity time to detect when people in the queue are motionless.

Detection of queues is more or less difficult depending on the environment [10, 11]. First of all, the lighting conditions may change over time, making it difficult to build and maintain a “background” image. It is the case for modern airports, where the light might come from the outside through large windows. However, as has been seen in the earlier discussion of the “stationarity” detection, our change detector is robust to such problems. A much more difficult and specific problem is induced by the camera location: when the camera is low and not in front of the queues, a part of a queue may be occluded by a crowd and/or merged visually with another queue.

4.2. Characterisation of a queue

To characterize a queue in the image, we postulate that people in queues are motionless at several periods and that a queue always starts at a specific location. This enables us to discriminate it from motionless people who do not belong to a queue. We will not use any hypothesis on the direction of the queue since it may evolve in time depending on several events (the opening of a check-in desk, for instance).
4.3. Initialisation process

To detect a queue, we need to detect regions in the image where people are motionless for a few seconds or minutes, and locate those intersecting specific locations such as ticket offices, cash dispensers, customs or boarding gates. These locations enable the system to distinguish between queues (stationary regions intersecting a selected area) and isolated motionless people in the scene.

Thus, during the installation, an operator needs to tell the system where queues may start. For this purpose, we developed a Human/Machine Interface (HMI). When there is no big perspective as shown in Figure 6, a rectangular shape is sufficient for the initialization of a queue. The two cases will be shown in the next sections.

4.4. Measurement process

The detection and the measurements of the queues are performed in two steps: first from the initialization, the queues are extracted from the stop duration of the people forming these queues; then an evolution in space and time of the queues is performed.

Queue location

As seen previously in the stationarity algorithm, for each “stationary” pixel, the system determines its stop duration. All the connected sets of
pixels classified as motionless for at least Stopduration are gathered to define a region called REG:

\[
REG_t(P) = \text{Label}_{\text{Reg}} \quad \text{if the pixel } P \text{ at image } t \text{ belongs to the region with a } \text{Label}_{\text{Reg}} \\
REG_t(P) = 0 \quad \text{if it does not belong to any region.}
\]

We can reasonably assume that the regions that overlap one of the selected areas (ticket office, for instance) are the queues.

To check the intersection, when the algorithm starts, an image corresponding to the selected areas is created. In this image, each pixel in a given selected area has its value set to the corresponding area label. The other pixels are set to zero (black).

The intersection computation is then a binary test:

\[
\text{IF } REG_t(P) \neq 0 \text{ and } AREA_t(P) \neq 0 \text{ THEN } \\
\quad \text{queue } (REG_t(P)) = AREA_t(P)
\]

At this point, we may face a problem. Indeed, a region may intersect several selected areas (queue(Label_{Reg}) receives various area labels). In the worst case, when the view angle of the camera is low and, when the camera is not in front of the queues, two different queues may appear as one due to the perspective effect. The developed system is not able to deal with such a configuration. Queues need, at least, to look separated.
as in the images shot at Newcastle airport (Figs 5 and 6). However, even in this situation, queue regions may, from time to time, merge due to people standing between them. We have also developed a procedure to cope with this situation (Fig. 7).

When a region covers several selected areas, an algorithm determines, through temporal differencing, parts (binary function A) that appear to split it into its various components.

\[
\text{if } REG_t(P) \neq 0 \text{ and } REG_{t-1}(P) = 0 \text{ then } A(P) = 1 \\
\text{otherwise}\hspace{1cm} A(P) = 0
\]

Each new part linked with either two previously separated queues, or with a queue and an “old” (region existing for several images) stationary region, is discarded (Figure 7).

![Diagram of queue split](image)

**Figure 7 – Process to split two merged queues.**

Conversely, a queue may be split into several regions due to low contrast or a gap between people.

If these regions are close to each other, and if they were merged on the previous image, the disappearing (binary function D) parts are reinserted to get the real queue region (Figure 8).

\[
\text{if } REG_t(P) \neq 0 \text{ and } REG_{t-1}(P) = 0 \text{ then } D(P) = 1 \\
\text{otherwise}\hspace{1cm} D(P) = 0
\]

**Shape analysis**

The first version of the software consisted in extracting the main axis for each queue region. Since the border of the queue region is not
smooth and sometimes with holes inside the region due to low contrast, for instance, the axis of the queue deviates. To smooth it, we use both a median filter and a morphological process. Finally, the axis, as well as the queue regions, is projected on the scene floor through calibration parameters to get an estimate of their real length.

The second version of the software consists in marking some pixels which represent more or less the beginning, the middle and the end of the queue and to draw a line. Then, segments are drawn to join these pixels and to constitute the axis of the queue.

4.5. Results

To assess the system, the difference between the real queue length and the estimated one is measured. For each tape recorded, we manually keep a cursor at the tail end of a queue. Every tenth of a second, its position (in pixels) is recorded. In parallel, the system determines, at the same rate, its own estimate of the position. The absolute average distance between the “true” location versus the estimated one is then measured. An average of all distances gives the global length error.

Images of queues were recorded at different moments of the day and of the year, to estimate both the robustness of the function to the lighting conditions (uneven lighting, direct sunlight, poor lighting, etc.), and to
various queue configurations (short or long, straight or curved, mixed with the passing passengers in the background, etc.). Although we did not go so far as to measure the actual length of the queues in the hall, we did check that the lengths “seen” by the system (shown as colored solid lines in the processed images) did match the queues as we saw them on the original video sequences.

The system was tested on more than 12 hours of videos covering airport scenes during off-peak and peak hours and on a second data set provided by SNCF. Thus, we obtained various types of queues, ranging from short ones to quite long ones.

Tables 3 and 4 show the global performances of the algorithm on the two datasets.

Tableau 3 – NUMERICAL RESULTS OBTAINED ON QUEUE LENGTH MEASUREMENT IN NEWCASTLE AIRPORT

<table>
<thead>
<tr>
<th>Number of measurements</th>
<th>Average queue length Error</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>1.68%</td>
<td>3.21%</td>
</tr>
</tbody>
</table>

Tableau 4 – NUMERICAL RESULTS OBTAINED ON QUEUE LENGTH MEASUREMENT IN MAGENTA STATION

<table>
<thead>
<tr>
<th>Number of measurements</th>
<th>Average queue length Error</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>115</td>
<td>1%</td>
<td>2.21%</td>
</tr>
</tbody>
</table>

Figure 9 provides some illustrations on the queues detection at Newcastle airport. The two upper illustrations show the queues are well detected and are not merged, in spite of their proximity. The next sequence shows a possible error caused by passengers staring at the information
desk. These are standing close to the end of a queue, but have not been mistakenly merged even if they are motionless.

The third queue, on the right, has not been detected because the user has not defined a “starting pad” for it. So, the system does not look for a queue at this place.

The lower case has been elaborated with the new version of the algorithm (a new HMI and a different way of calculating the axis of the queue).

Figure 9 – Examples of detected queue axes at Newcastle Airport.

There are two main explanations to the errors:

There is a delay between the creation of a queue and its detection, due to the time to detect a stationarity ($Stop_{duration}$ in the worst case, off-peak period).

The discrimination between queues and other stationary areas is not always perfect.

5. Detection of people loitering

Modern public transport systems are often characterized by regularities in the pattern of movements of passengers, vehicles and staff. Assu-
when this is the case, it may be useful to identify behavior that does not conform to the regularity, since it may be symptomatic of an anomalous event that is worthy of the operators’ attention.

Examples of such irregular activity include 'loitering’ in a place where normally passengers are constantly moving, or a delay in the departure or arrival of an otherwise punctual service. In the special case where this regularity can be described as a periodic activity, the anomalous events, such as loitering, can be detected using the deviation from the expected periodic variation. Here, the proposed approach is a reduction of the multi-dimensional data (i.e. color video data) to a one-dimensional signal, from which the periodicity can be automatically modeled and anomalies consequently detected. This approach is explained as follows.

First, a model which explains the underlying process of the scene is constructed based on the method in [12] using normal data. This model consists of a number of states; therefore the posterior probability of any data having arisen from each state of the model can be calculated. This, in turn, is then used to calculate the entropy of the system, producing a 1D signal. In this application, the periodic behavior exhibited by the signal is exploited where any deviation from the expected periodic trend is classified as anomalous. This process was carried out on a dataset captured at a Turin underground station, and the 1D entropy signal is shown in Figure 10. Normal events, as well as anomalous events, are annotated.

Two anomalies are present here: loitering, and the late arrival of a train.

Next, the 1D signal is processed to allow anomalies to be detected automatically. The problem has been addressed in fields such as medical signal processing [13, 14] condition-based monitoring [15]. Here, we use an unsupervised approach. Taking a Fourier transform of the time series in Figure 11, one period of the signal was found to be of a duration of 177 seconds. This allows the extraction of a pattern from the time series, which is used as a “template” for comparison with the remainder of the data. In an online application, this process is performed using a fixed window of samples, the duration of which should include at
least 5 periods of the signal. The template can be updated online, where the samples can be selected using a sliding window. Here, the entropy time series shown has been processed in its entirety to demonstrate the approach.

The metric used to detect the anomalies is based on the Kullback-Leibler (KL) divergence:

$$D_{KL}(f_0 \parallel f_1) \times D_{KL}(f_1 \parallel f_0)$$  \hspace{1cm} (10)$$

where $f_0$ is the template, and $f_1$ is the sub-sequence in the time series. Figure 11 shows the “degree of anomaly” of the signal, where a threshold of -100 has been used to produce a binary classification of the presence of anomalies (a value below the threshold is anomalous). The points which drop below the threshold correspond to the two anomalous events of interest: loitering, and the delay in train arrival. Experiments on normal data detected no anomalies, as expected.

6. Edge following and polygon-based method for intrusion detection

The aim of this application is to improve safety by detecting automatically people or objects falling on or crossing the tracks and people entering or exiting from the tunnels.
Figure 11 – The “degree of anomaly” based on the KL divergence.

Through computer vision processing, the system determines and sends, in due time and via a satisfactory Human Machine Interface (HMI), alarms and the corresponding video image to the system operators. Detection speed is a major improvement for safety and the minimization of operation disruption. The system is built and installed on a site to inform in real time on the existence of incidents.

In the current version of the system, to detect falls on the tracks, we propose to fix the cameras on the ceiling, aimed vertically so that the framing delineates clearly what is standing on the platform from what is above the tracks, so as to reduce the complexity of the software, and reduce the computing time accordingly.

To detect falls or intrusions, only the moving edges in the “tracks” part of the picture will trigger a warning. With this particular camera location, passengers bending down across the edge are detected as possible incidents, though other camera placements do not detect such incident scenarios and are being considered with the French operator RATP.

6.1. Extraction of Moving Edges (ME)

In order to cope with the very difficult imaging conditions (uncontrolled lighting, changing background, unknown moving objects), a specific robust algorithm has been designed to perform the detection of the moving edges. This algorithm is based on the analysis of the grey level differences between successive frames of the video sequence. Edge extraction, performed on the difference of successive images, allows re-
taining only the edges of moving objects in the frame (the word object is used here in a very wide sense, for example it can apply to a person).

The pre-processed data is then filtered to eliminate false incidents, in particular objects located on the platform, or too small to cause concern. This filtering process is detailed later in this paper.

A particularly interesting feature of this algorithm is the ease to implement it to allow real-time detection. This can lead to a substantial reduction in overall processing time. Moving edges are available as a sequence of binary frames in which the pixels belonging to moving edges are coded as white and all the others remain black (see Figure 12).

![Initial frame](image1.png)  ![Moving edges](image2.png)

Figure 12 – Result of the detection of moving edges in a video sequence representing a person entering a tunnel.

### 6.2. Reconstruction and modelling of the ME

The extraction of moving edges provides an image containing the edges of the objects in motion in the scene. In order to use this image for scene analysis, more processing is necessary, yielding a list of moving objects along with their characteristics. This processing is composed of 3 successive steps:

- Find and label the moving edges in the image, trying to represent them in the most convenient way for the next and last step:
Video-base Detection

- Model the edges as convex polygonal shapes,
- Merge the overlapping polygonal models (assumed to belong to the same object), and calculate their geometrical characteristics.

The moving edges carry very important information regarding the moving objects in the scene. However, this information is somewhat distorted by its very nature of representing only those parts of the objects that are in motion. For instance, a circular shape such as a rolling ball usually yields two unconnected edges, corresponding to the parts with high visual motion. On the other hand, the parts of the circular shape tangential to the movement are not visually moving, and do not yield detection of moving edges. The next step is therefore to recombine these pieces together to reconstitute the original object, while avoiding mistakenly merging separate moving objects.

The basic idea is to model the moving edges by simple, polygonal shapes, so it is easy and fast to measure their main characteristics: size, surface, orientation, etc. Several methods have been proposed to perform such modelling: for instance, Pavlidis searches a polygonal bounding box of any shape, concave or convex, without limitation on the number of sides. The results, in our application, are disappointing.

**Edge labelling**

The algorithm starts by looking for each moving edge in the image (moving edges are all the parts of the image that are not black, appearing as grey or white spots). This is straightforward procedure, repeated when an edge has been fully processed: the software scans the image line by line until it encounters a pixel that belongs to an edge that has not been processed yet (the pixels belonging to processed edges receive a specific “pseudo-grey level” indicating that they have already been processed). The software then starts modelling the new edge it has just found.

The next step of the algorithm is to recognize the edges as such, and label them to prepare the modelling. This is done as follows:

Firstly, the spurious edges (edges reduced to one pixel) are rejected. These are caused by noise from the camera, for instance, and do not
usually correspond to actual objects. Then, edges constituted of at least 2 pixels are followed.

**Modeling by rectangles**

A simple, fast way to model a shape is by finding the smallest bounding rectangle. The computation is particularly fast if the rectangle is upright, i.e. composed of horizontal and vertical edges. It is also easy to merge overlapping models to reconstitute the integrity of objects. However, this kind of model is sensitive to the orientation of the object, in that its perimeter and surface can be grossly exaggerated in the case of a thin object with a 45 orientation (cf. Figure 13). This approach has been used in the past, for instance to detect falls on the tracks in subways within the PRISMATICA project [1], an application in which the reaction time is critical. However, the computing power achieved by even small PC’s has increased considerably with time, and it is now possible to use a less crude approximation of the real objects, while staying in the real-time domain.

![Figure 13 – Modelling by a rectangle. Note the widely overestimated size caused by the simplistic definition of the model.](image)

Note that it is also possible to find the best bounding rectangle by removing the constraint on its orientation. However, its determination is lengthy (computing the inertia tensor, inertia axes, etc.), so this approach is not as interesting as it seems, due to its comparatively poor computation time / precision trade-off.

**Modeling by convex octagons**

In order to improve the precision without the drawback of a grossly increased computational burden, we chose to refine the model by adding
more sides to the bounding polygon, whilst keeping the constraint that the sides must be “upright”, i.e. they have specific orientations so we do not need to calculate them, resulting in a less accurate, but faster to compute, approximation of the actual shape. In the case of octagons, for example, these orientations are 0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°.

We use an algorithm designed to model the shapes by octagons, with the constraint that the shape is supposed to be purely convex, with no concavities. This is the case with basket balls, but also with people, animals and many objects of the real world that are likely to fall on the tracks of train or subway stations.

The result obtained on a ball is considerably closer to the reality than the Pavlidis model (cf. Figure 14). The ball is now well modelled as a single object, with geometric characteristics close to the actual ball (it is not exactly circular, though, but this is not really annoying in the intended application, in which it is mostly the size that matters). Another case of persons walking on the tracks is illustrated in figure 15.

![Initial frame](image1) ![Moving edges](image2) ![Modeling result](image3)

**Figure 14** – People and objects modelled by convex octagons.

7. CONCLUSION

A system for multiple-sensor surveillance in key public transport networks has been described. The design was closely inspired by what pu-
public transport operators are familiar with, i.e. the concepts of distributed sensors, distributed monitoring and decision-making in a control room environment. Results have been presented that demonstrate the capability of automatic systems to detect specific events and of a distributed system that can promptly alert operators providing multiple sensor views of the events. Real-world demonstrators on key sites have demonstrated the feasibility of the approach.

In terms of functionalities or events, we have presented four different examples of events recognition: stationarity, queuing intrusion detection and loitering.

The results of the trials show that the output of the algorithms implemented on the local camera network meet the expectations, with, in particular a good robustness to lighting conditions and lack of contrast.

Further trials and developments are currently underway e.g. in London for a more intensive evaluation of the loitering function.

ACKNOWLEDGMENT

We particularly thank the public transport operators who supported us in defining, developing and testing the distributed camera network and the applications: Régie Autonome des Transports Parisiens (RATP), Azienda Trasporti Milanesi (ATM), London Underground Ltd, British Airports Authority, Newcastle International Airport Ltd, Société Nationale des Chemins de Fer (SNCF) and others whose help was invaluable.
References


