ISCTE IUL
Instituto Universitário de Lisboa

Lisbon University Institute

Department of Information Sciences and Technologies

INTELLIGENT VIDEO OBJECT TRACKING IN LARGE PUBLIC ENVIRONMENTS

A Dissertation presented in partial fulfilment of the Requirements for the Degree of Master in Telecommunications and Informatics Engineering

By

Marco Paulo Fernandes Ferreira

Supervisor:
Doctor, Paulo Jorge Lourenço Nunes, Assistant Professor

October, 2009

Abstract

This Dissertation addresses the problem of video object tracking in large public environments, and was developed within the context of a partnership between ISCTE-IUL and THALES[1]. This partnership aimed at developing a new approach to video object tracking, based on a simple tracking algorithm aided by object position estimations to deal with the harder cases of video object tracking. This proposed approach has been applied successfully in the TRAPLE[2] project developed at THALES where the main focus is the real-time monitoring of public spaces and the tracking of moving objects (i.e., persons).

The proposed low-processing tracking solution woks as follows: after the detection step, the various objects in the visual scene are tracked through their centres of mass (centroids) that, typically, exhibit little variations along close apart video frames. After this step, some heuristics are applied to the results to maintain coherent the identification of the video objects and estimate their positions in cases of uncertainties, e.g., occlusions, which is one of the major novelties proposed in this Dissertation.

The proposed approach was tested with relevant test video sequences representing real video monitoring scenes and the obtained results showed that this approach is able to track multiple persons in real-time with reasonable computational power.

KEYWORDS

Multiple video object tracking, occlusions handling, people tracking, real-time video surveillance.

[1] THALES Security Solutions and Services, SA
[2] **TRA**cking **P**eople in **L**arge public **E**nvironments

Resumo

Esta dissertação aborda o problema do seguimento de objectos vídeo em ambientes públicos de grande dimensão e foi desenvolvida no contexto de uma parceria entre o ISCTE-IUL e a THALES. Esta parceria visou o desenvolvimento de uma nova abordagem ao seguimento de objectos de vídeo baseada num processamento de vídeo simples em conjunto com a estimação da posição dos objectos nos casos mais difíceis de efectuar o seguimento. Esta abordagem foi aplicada com sucesso no âmbito do projecto TRAPLE desenvolvido pela THALES onde um dos principais enfoques é o seguimento de múltiplos objectos de vídeo em tempo real em espaços públicos, tendo como objectivo o seguimento de pessoas que se movam ao longo desse espaço.

A solução de baixo nível de processamento proposta funciona do seguinte modo: após o passo de detecção de objectos, os diversos objectos detectados na cena são seguidos através dos seus centros de massa que, normalmente, apresentam poucas variações ao longo de imagens consecutivas de vídeo. Após este passo, algumas heurísticas são aplicadas aos resultados mantendo a identificação dos objectos de vídeo coerente e estimando as suas posições em casos de incertezas (e.g., oclusões) que é uma das principais novidades propostas nesta dissertação.

A abordagem proposta foi testada com várias sequências de vídeo de teste representando cenas reais de videovigilância e os resultados obtidos mostraram que esta abordagem é capaz de seguir várias pessoas em tempo real com um nível de processamento moderado.

PALAVRAS-CHAVE

Seguimento de múltiplos objectos de vídeo, processamento de oclusões, seguimento de pessoas, videovigilância em tempo real.

Contents

List of Figures

List of Tables

List of Acronyms

CD	Colour Difference
DB	DataBase
DD	Distance Difference
EKF	Extended Kalman Filter
fps	Frames per second
GUI	Graphical User Interface
ID	Identifier
IP	Individual Person
KBOT	Kernel-Based Object Tracking
KF	Kalman Filter
MAP	Maximum A Posteriori
MDE	Mean Detection Errors
ME	Metadata Engine
MF	Media Fusion
MIC	Mean Identification Changes
MMDB	MultiMedia DataBase
MMI	Man-Machine Interface
MMSE	Minimum Mean Square Error
MPEG	Moving Picture Experts Group
MSD	Mean Successful Detections
PHD	Probability Hypothesis Density
RFID	Radio Frequency Identification
RFS	Random Finite Set
TRAPLE	TRAcking People in Large public Environments
VO	Video Object
VO_ID	Video Object Identifier

Chapter 1

Introduction

Nowadays, surveillance cameras are widely present in many large public places. This extensive coverage was fostered, not only by the hardware developments, e.g., video cameras, communication networks, and processing power, but also by the scientific developments in the fields of machine vision and machine learning, leading to automated surveillance systems supported by computer-based image processing methods.

These automated surveillance systems have received a growing attention in the last years as an important technology to achieve improved security, namely in large public environments (e.g., shopping malls, railway stations, airports, etc.). These systems in some cases may be crucial in the detection of abnormal situations that a human would fail to detect with the intensive monitoring of surveillance system screens. Therefore, the limitations of human perception have to be overcome, concerning the simultaneously collection, processing and storage of large amounts of data [1]. The automatic analysis can provide real-time alerts or data enhancement to focus the attention of a human operator to an important event.

This Dissertation has been developed under the scope of the THALES[3] TRAPLE[4] project in collaboration with ISCTE-IUL, aiming at the development of an automated surveillance system for large public environments.

The TRAPLE system consists of various building blocks like video processing, artificial intelligence, multimedia databases and Radio Frequency Identification (RFID) detectors. This system aims to detect and track a large number of moving objects (in this case persons) in real-time, exploiting the characteristic object behaviours and trying to infer common and abnormal behaviours, such as the most used paths and violations of restricted areas. If, for example, an object has some unexpected behaviour, the system should be able to trigger an alarm in order that a human operator could devote special attention to such event.

The aim of this Dissertation is to develop and study video object tracking techniques for efficient and accurate people tracking using images from a surveillance camera that have been pre-processed through a moving object detection pre-processing module that outputs a set of candidate objects to track in the form of binary masks.

Although video object tracking can be seen as a relatively easy task for a human, at least for short periods of times and a reduced number of objects, for a computer, this is far from being a trivial task, notably, to understand how the objects move and who is who through the time, i.e., through the various adjacent video frames. For the computer the video images can be processed in two different levels. The first processing level is based on "low-level vision algorithms" that deal with numerical operations on the image pixels as will be shown in object detection in Section 3.1.1. These algorithms only deal with groups of pixels that change of colour over time. Then belongs to the man to make the machine understand what is happening behind these pixels changes. Therefore, a second processing level based on "high-level vision algorithms" can interpret the "low-level" results in a similar way that the human eye does, associating different spatial and temporal results. "Low-level algorithms" are usually very complex and the amount of information extracted from the scene, in most of the cases, is small but very knowledgeable. In the other hand, the

[3] THALES Security Solutions and Services, SA
[4] **TRA**cking **P**eople in **L**arge public **E**nvironments

"high-level algorithms" are usually less complex and produce large amounts of information, but that information can have inaccuracies.

Besides the computer-vision difficulties to recognize the objects, other factors related to the objects motion and the environment make video object tracking a difficult task. Factors like the mixture of people walking around the monitored scene, which means that some persons walking in front of others can create occluded areas between objects. Besides the mixture of people, that generates occluded areas to the camera, environment obstacles are also a problem for tracking, notably because, when a given object moves to the back of these obstacles, the camera is not able to perform a continuous tracking. In both occlusion cases, it is fundamental to make identification correspondence of the objects when the objects become visible to the camera again.

Figure 1.1 shows an example of multiple video objects tracking where three objects that are being tracked maintain its identifications over time, represented by the same bounding box colour over two temporal adjacent video frames. Figure 1.2 shows two different types of occlusions: while in Figure 1.2a the occlusion is between two moving objects, where one object moves in front of another (non-static occlusion), in Figure 1.2b the occlusion is originated by an object moving behind a static object in the scene, i.e., an obstacle (static occlusion); the areas with possible occlusions are marked with red bounding boxes in both cases.

In these cases it is difficult to know the actual object position precisely. Therefore, some estimations or verifications can be done to overcome this problem. The non-static occlusion can happen in any place of the image, including the entrance areas where it is impossible to know information from the past of the objects. Other factor that makes the estimation difficult during these occlusions is because persons interact between them modifying their trajectories while they are occluded. In the static occlusions it is easier to estimate the trajectory of a person, because the occluded area is always the same, and the person's movements are more regular.

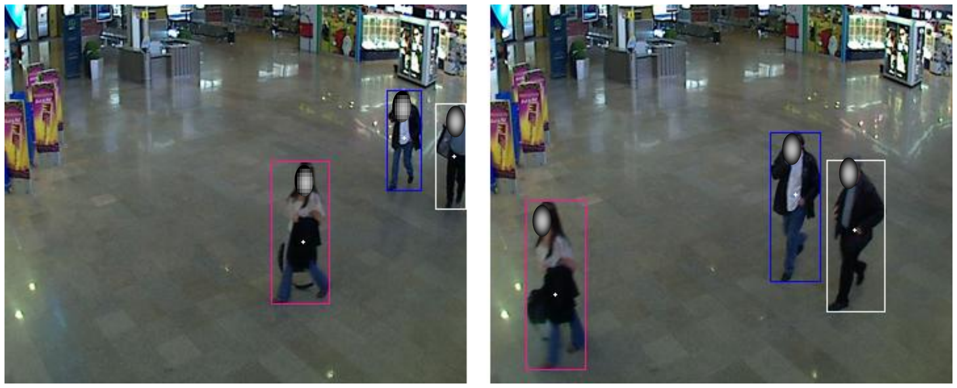
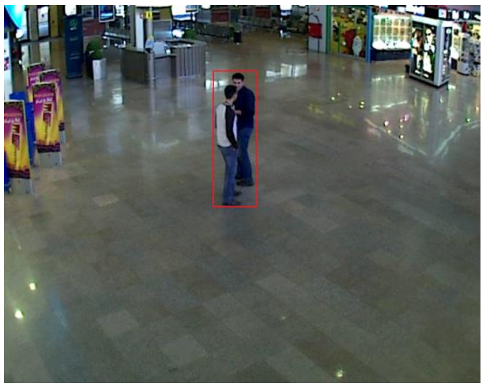
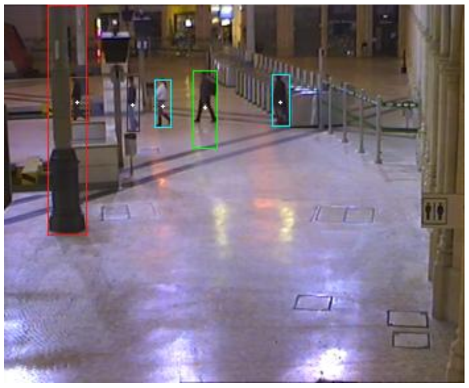


Figure 1.1 – Object identification correspondence



(a)



(b)

Figure 1.2 – Occluded areas: (a) non-static; (b) static

A proposal solution is presented in this Dissertation concerning the problems of video object tracking and the computational power that are available nowadays.

This proposal tackles the problem of video object identification management with some inherent difficulties, such as:

- Video objects size variation in space;
- Video object occlusion (non-static and static);
- Entering and exiting regions of the scene;
- Large densities of people in the scene.

Consequently, a solution composed by various algorithms organised into a set of modules was designed in order to solve such problems efficiently, based on two simple concepts:

- Track of video objects through their centroids;
- Estimation of the video object positions when not enough information about the objects position is available.

This proposal was tested and proved that it can be used in real environments with very good accuracy as can be seen in Chapter 5.

After this brief introduction, this Dissertation presents with more detail the TRAPLE project in Chapter 2, briefly presenting the structure of the system and unmasking some of the principles and objectives of the system; afterwards, Chapter 3 presents some of the main problems of the object tracking and a review of the literature concerning the main techniques and algorithms used in video object tracking; in Chapter 4 an intelligent video object tracking solution proposal for TRAPLE is presented describing the major problems addressed and explaining the choices made to solve them; after presenting the proposed video object tracking solution, Chapter 5 presents the set of tests designed to evaluate and to validate the proposed solution as well as the a discussion of the achieved results highlighting the main strengths and possible limitations of this solution; finally Chapter 6 presents some conclusions based on the major outcomes of this Dissertation and provides some possible future directions for this work.

Chapter 2

Overview of the TRAPLE Project

2.1 Introduction

This Dissertation was developed under the scope of the TRAPLE project, an internal project from THALES Security Solutions and Services, SA. Therefore, before presenting the major addressed problems and solutions proposed in the context of this Dissertation, this chapter gives an overview of the TRAPLE system.

TRAPLE proposes a new concept for tracking people in large environments. This concept lays on the usage of multiple sensor technologies to capture the presence of people, track them, follow their predicted behaviour, and provide alerts to the security operators when those behaviours are not the ones expected or when typical undesired behaviours are detected.

The proposed technologies for the presence sensors were networked video cameras and RFID sensors. With these sensors the information that can be extracted is obviously different, but with a mix of simple information of both sensors that can be rapidly processed it is possible to provide more reliable and rich information about each individual person. It is obvious that the information captured by each sensing technology needs some intelligent applications to analyse it and maintain it coherent, exploiting the relations between both types of information. The individual detection results along the scene define the corresponding object path in the monitored environment and based on these paths, discrepancies from "typical or desirable behaviours" can be detected in the form of system warnings or alarms.

For this purpose the video analysis methods have to be optimized for performance, enabling the tracking of a large number of individuals in real-time, which constitutes the major problem that this Dissertation aims to tackle.

The TRAPLE conceptual solution may be used in various public environments; an obvious segment where the system is applicable with possible benefits in efficiency is the Airport segment.

Other environments such as shopping malls, rail or metro stations, fairs or expo sites and large cultural or sports events may also make use of some or all of the proposed techniques. In summary, the major TRAPLE objectives are the following:

- Track people only with video images (continuous tracking)
- Track people only with RFIDs (persons identification)
- Cross information between the two kind of sensors
- Analyse the persons behaviours
- Analyse groups behaviours
- Analyse typical paths of persons
- See videos with the entire path of one person or group
- Monitoring in real-time

To handle all this objectives a system was designed with an overview of all the functions it needs, as seen on next section.

2.2 TRAPLE Architecture

Figure 2.1 shows a "high-level" architecture of the TRAPLE system. This architecture is composed of six main modules and two databases: one to store the video content (Multimedia database) and another to store the tracking positions as well as the events associated to the tracked persons and their relation to the correspondent video content (General database). All these components work together to address the main objectives defined for the system. The various modules are interconnected through a network structure where the various modules exchange information among each other to achieve their particular goals.

In this architecture, two of these modules are responsible for processing the data generated by the various sensors, i.e., video cameras and RFID's, and are entitled, respectively Image Processing and Tag Processing modules. They receive the raw data from sensors and must be able to process it in real-time as efficient as possible in the available time between sampling instants.

The information generated by these modules will feed the Metadata Engine module (ME) where all data generated by the sensor interpreters are routed to the system databases, to the Man-Machine Interface (MMI) and to an intelligent analysis module called Media Fusion (MF). The MF module compiles the captured data allowing to draw some conclusions from these data, such as typical routes for people with similar profiles or generating alarms for dangerous situations. The simulator module has the objective to generate and feed object location information to the MF through the ME. This information is only provided when the processing modules do not generate real information. The last module is the MMI, which constitutes the interface module where information requests are presented to human operators.

All these modules will be detailed in the next sections.

2.3 Tag Processing

The objective of this module is to obtain the positions of the persons with RFID Tags. These positions are received from the RFID sensors and when one tag is associated to one person it should be possible to know where this person is along time. This module returns to the system the positions of the persons holding tags, which have been previously associated to some personal user information.

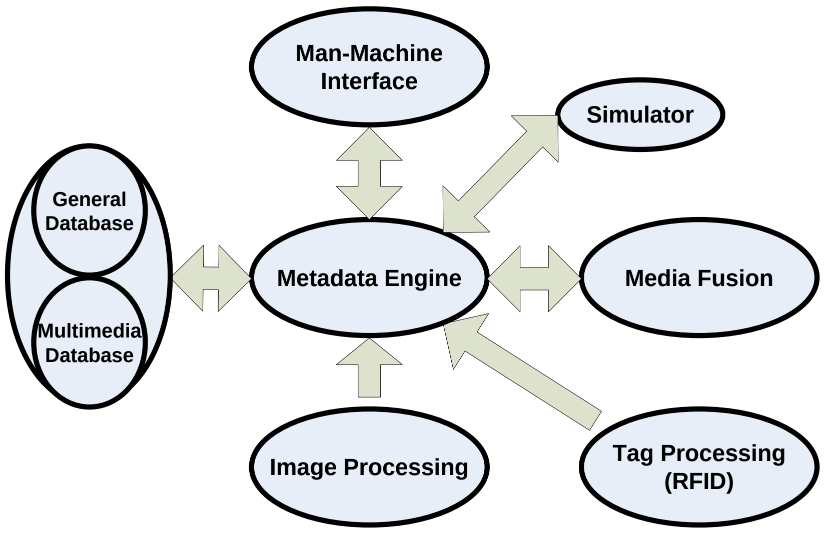


Figure 2.1 – High-level TRAPLE system architecture

2.4 Image Processing

The main objective of this module is to track objects in the monitored scene along time using video images only. To accomplish its objectives, this module has two complementary sub-modules: 1) video object detection and 2) video object tracking. The first sub-module is responsible for detecting the moving areas in the scene, i.e., the areas where the persons to be tracked are located in the video images (foreground areas). The second sub-module, developed in this Dissertation, is responsible for tracking the detected moving objects, associating an identification to these video objects (persons) and tracking them along the scene always with the same identification, i.e., one object is tracked along the scene with its centroid and associated characteristics. This module is responsible for the positioning of the objects in the real space and for communicating with the other modules using standardized information (MPEG-7 compliant messages [2]).

2.5 Metadata Engine

This module is the "heart" of the TRAPLE system; its main objective is the management of all the data exchanged with the other modules. This module receives information from the sensor modules saving their information on the corresponding databases and forwarding this information for the modules that have requested it. This module includes some intelligent searches that make it possible to ask for the path of an object or for persons that have been on a given location. These searches will allow the construction of various videos from different cameras, tracking a person or only information about his behaviour in the monitored space. The entire flow of information is based on the exchange of MPEG-7 compliant messages [2] facilitating the interaction between the various modules and the addition of new modules in future.

2.6 Simulator

The main objective of this module is to generate data that will substitute the tag and image processing data. The simulator will be useful to the media fusion module (described in the next section) to have some information about paths of persons on scene on the start-up of the entire system. These data will be used for testing the media fusion module before the tag and image processing modules are finished. After these modules are finished the simulator will initially simulate all these situations that could occur in a real scenario and this data will be the background of media fusion when the video processing and the tag processing start the tracking of people on scene.

2.7 Media Fusion

The media fusion module aims essentially at performing behaviour analysis, i.e., to identify the typical behaviour(s) or deviations to expected behaviour(s) exhibited by people present in the scene under surveillance. This analysis will be mainly based on the people's motion while in the scene. One of the functionalities of this module is to classify each person in the scene according to its speed using four different states: 1) stopped, 2) walking, 3) running, and 4) waiting. If the person has sudden and repeatedly changes of state in the scene, this situation is reported through event messages to the metadata engine. Another situation that has to be reported is if some people are approaching to a prohibited area.

Another important functionality of media fusion is the typical paths construction. These typical paths, initially constructed with simulated data, will be used to be compared with real data that comes from video. These data will be used to be compared with every location of a person and know if the person trajectory matches with some of the typical paths already constructed with these data. The real persons trajectories seen will influence the typical paths defined, updating it when the persons leave the scene. If some trajectory does not match with one of the typical paths, then a new path is constructed based on this trajectory.

2.8 Man-Machine Interface

The Man-Machine Interface (MMI) is the “face” of the TRAPLE system. This is the user interface where the maps of the monitored places can be visualized together with the video objects positions or the tag positions superimposed on the maps through an extra visual information layer. The states assigned by the Media Fusion to the moving persons in the scene can always be seen as well as some alerts, e.g., people running, walking or people in a restricted area. The MMI has also interfaces to perform queries to the Metadata Engine about some individual, group or local on scene. This module has interfaces which allow seeing the positions of the persons in the map and has the possibility to see the live stream cameras to monitor some strange behaviour. Another functionality of the MMI is the possibility to see constructed videos previously queried, tracking someone along the monitored scene.

2.9 Final Remarks

This section has presented the TRAPLE system composed by their six main modules: Tag processing, Image processing, Metadata engine, Simulator, Media fusion and Man-machine interface. All these modules work together to compute information about the people present in a monitored space having their location in real-time and storing the past movements of persons allowing to search the past. For this are used multiple sensor technologies to capture the presence of people, and track them, analyzing their behaviours.

In summary, this project could be a good tool for a lot of enterprises that want to monitor public spaces and is an innovative system that has a different approach on the “video processing world” which is actually patented [3].

Chapter 3

Video Object Tracking: State-of-the-Art

3.1 Introduction

This Dissertation addresses the video processing module, specifically tackling the problem of video object tracking.

Video object tracking is intimately connected to the video object detection: to have accurate tracking of video objects the system must also provide adequate video object detection as a pre-processing step to the tracking system. Therefore, a brief description of the main objectives and functionalities of the video object detection module is first provided followed by the main objectives of the tracking system and an overview of the most popular video object tracking techniques already proposed by other researchers in the literature.

3.1.1 Video Object Detection

Before tracking the video objects, the system has to know what it needs to track, so the video objects have to be detected previously. They can be detected through motion detection, on the video images or by comparing each frame to a background model. A video object detection algorithm has to classify the pixels in a video sequence as belonging to the background or to a foreground object; the technique for this purpose is called of background removal or background subtraction. This classification allows localizing the video objects in images. In Figure 3.1 can be seen four of the video processing steps to reach the tracking: 1) the background model used to subtract the images (top left) that represents the empty space under surveillance; 2) an example of a thresholded background subtraction (top right); 3) the thresholded background subtraction after applying the morphological filters (bottom left); 4) the current image with the video objects detected labelled with coloured bounding boxes (bottom right). White pixels represent the background and black pixels represent the foreground (in this case, the moving objects).

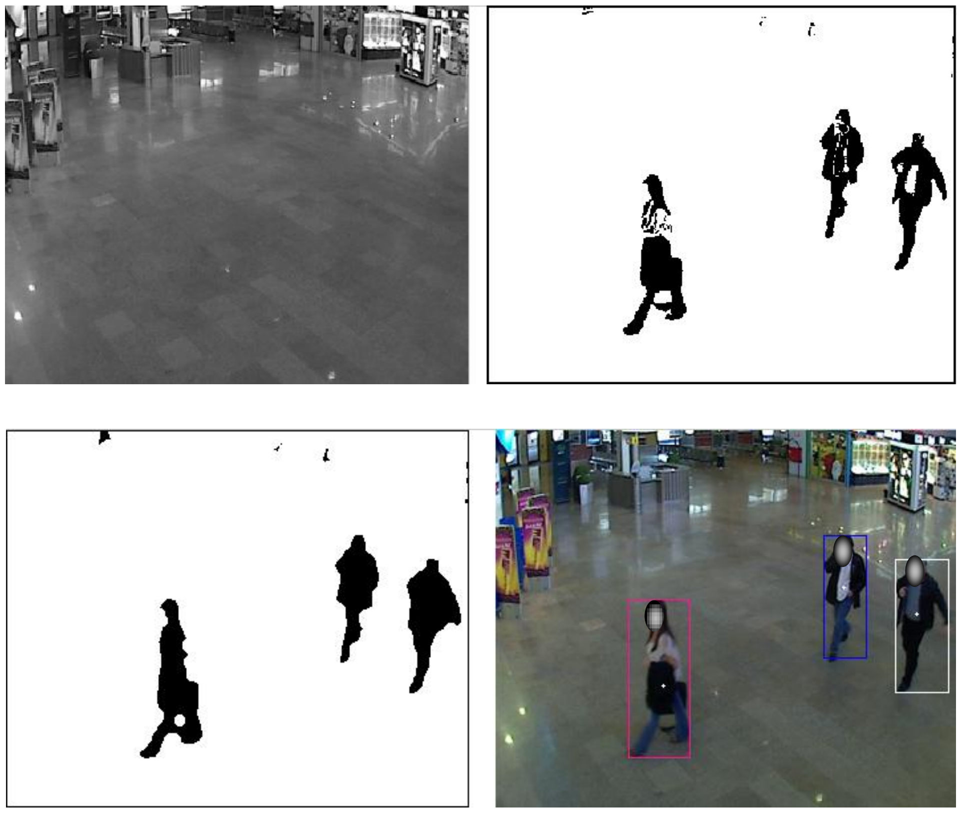


Figure 3.1 – Background subtraction example

In some cases such algorithms must be precise in object contour detection (spatial accuracy) and in temporal stability of the detection (temporal coherency). In this case the spatial accuracy is not a critical point because the major objective of the developed tracking system is to provide information about where the objects are and not about what people are doing; consequently perfect object contours are not required. Nevertheless, with accurate object contours, better object models can be extracted from foreground data and consequently better object discrimination (i.e., between objects and between each object and the background) can be achieved.

Video object detection algorithms should also be able to detect changes of small magnitude (sensitivity) and providing good accuracy under varying scene conditions, such as illumination changes, shadows or reflections (robustness). The real difficulty is to classify a foreground object if it has similar characteristics to the background (camouflage [4]) or if the object becomes motionless (sleeping person) [5].

3.1.2 Video Object Tracking

After detecting the foreground areas, the system will track the corresponding video objects. Tracking can be simply defined as the estimation of the trajectory of an object as the object moves around de scene; an algorithm analyses the video frames and outputs the location of moving targets within the video frame. If the object is well defined, it can be continuously observed because its shape, size or motion does not vary too much along the time. But the objects, specially the persons (the main focus of this Dissertation), do not usually exhibit constant motion, and even worst, an object can be occluded in the scene by a scene structural element or another moving object, which is a very common situation because people usually move in groups (see Figure 1.2a and Figure 1.2b) [6].

Another relevant problem is the number of targets that the system wants to be able to track. Tracking problems are usually modelled as a dynamic system, where a fixed number of targets are established. However, the problem becomes more challenging when the number of targets to track is not known or if this number changes along time [7]; the computational cost typically increases with the number of targets. This fact could compromise all the tracking processing to be done in real-time.

So the main difficulty in video tracking is the target localization in consecutive video frames, especially in the following situations:

- Fast object movements (when the objects are moving fast relatively to the processed video frame rate);
- Crowded scenes (when many objects are moving in the scene);

- Occlusions (when the scene has obstacles or between moving objects).

In the following sections will be presented an overview of some of the most widely used algorithms in the development of automated surveillance systems with respect to tracking objects in real-time, which is a key to the smooth operation of these systems.

3.2 Bayesian Sequential Estimation and Particle Filtering

One of the most used tracking algorithms is based in particle filtering, whose objective is to track a variable of interest, typically with non-*Gaussian* and potentially multi-modal probability distribution function *pdf*. Multiple copies (particles) of the variable of interest are used (set of pixels), each one associated with a weight that reflects the importance of that specific particle.

Target tracking is often formulated as a state x_t estimation problem at instant *t*, which may differ in interpretation over the individual objects and can be a concatenation of different states $x_{k,t}$ (e.g., position and velocity). Particle filtering is a nonlinear state estimation technique, where a filtering distribution $\left(p(x_t|y_{1:t})\right)$ is done denoting all the observation up to the current step $(y_{1:t} = (y_1 \dots y_t))$. This distribution can be computed with Bayesian sequential estimation into two recursive steps. First step is the prediction step (see eq. 3.1) that follows from marginalization and the second step, the filtering step (see eq. 3.2), have a new filtering distribution that is obtained directly by applying the Bayes rule [8].

Prediction Step:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})\, dx_{t-1} \tag{3.1}$$

Filtering Step:

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)\, p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \tag{3.2}$$

To get a good recursion of these steps, it is necessary to specify two models: i) the model describing the state evolution $p(x_t|x_{t-1})$; and ii) the model for the state likelihood in light of current measurements $p(y_t|x_t)$. These models have to follow the Markov assumption which requires the current state to be independent of all the previous measurements given the previous state, and the current measurements are independent of all the previous measurements given about current state. The recursion has to be initialized with a distribution pattern. Once the sequence of filtering distributions is known, point estimates of the state can be obtained according to any appropriate loss function, leading to, e.g., maximum a posteriori (MAP) and minimum mean square error (MMSE) estimates [8].

If the tracking recursion results in a linear and Gaussian models (i.e., the input is assumed to have a Gaussian distribution and the system is linear) it can be used a well-known type filters, the Kalman Filter (KF), where the mean and variance of the particles are updated using simple adaptive filters. However, for the general case the linear and Gaussian assumptions do not hold, therefore, the tracking recursion becomes analytically intractable. To solve this problem, it is necessary the use of approximation techniques. One of them is the Extended Kalman Filter (EKF) [9][p.56], which provides a linearization of the models. In these cases, when the state space is discrete and consists on a finite number of states, the Sequential Monte Carlo methods, more known as particle filters, can be applied. This method is a numerical approximations strategy to compute tracking recursion on complex models.

This particle filters makes a very simple task, they start with a weighted set of samples approximately distributed according to a model, after that, new samples are generated from a suitable designed proposal distribution, which may depend on the old state and the new measurements. From time to time it is necessary to resample the particles -to avoid degeneracy of the importance weights of the particles (e.g., set of pixels in the corners of an object or near its centroid). So it multiplies high importance particles and discards low importance particles of the object to track.

This algorithm and its efficiency, simplicity, flexibility, easy implementation, and modelling success over a wide range of challenging applications makes this approach very popular in the surveillance researches and other areas [8]. In this Dissertation the idea is to avoid intensive processing. In the case of the Particle filtering approach the algorithm falls in searching for a lot of particles for each object that will be always being predicted and filtered involving computing matrices, a hard operation for a computer.

3.3 Particle Probability Hypothesis Density Filter

This algorithm uses a probability hypothesis density (PHD) filter to automatically track an unknown number of people. The targets state and the measurement of a sensor are represented by two random finite sets (RFS) [10]. One of them represents a clutter RFS of the targets and the other represents the predicted multi-target RFS, both of them are Poisson-distributed. Like in Particle filtering, the state of the targets can represent position, velocity or other characteristics of an individual person.

A measurement of a sensor is represented by a measurement vector y. The measurement set at time t is also represented as a random finite set $Y_t = \{y_1, \dots, y_{M(t)}\}$ and *M(t)* is the variable measurement number at time t. Y^t is used to denote the time sequence $Y_1, \dots, Y_t$, i.e., the measurement sets accumulated from time 1 to time *t*.

The PHD of a random finite set is the equivalent of the expectation of a random vector. The integral of the PHD over a region in a state space is the expected number of targets within this region, so the peaks of PHD are the points with the highest local concentration of the expected number of targets and can be used to generate the estimates for states of targets [10] like it can be seen in Figure 3.2.

The PHD filter consists of two steps, prediction and update where x_t is the state estimation at instant t and y_t represents the measurement set at time t The prediction step (see eq. 3.3) works with some factors like the intensity function of the spontaneous birth (new objects appearing in scene) RFS ($b(x_{t+1})$), the intensity function of the RFS of targets spawned from the previous state $\left(b(x_{t+1}|x_t)\right)$,where x_t is the previous state, the probability that the target still exist at time t+1 given it has previous state ($p_s(x_t)$) and the transition probability density of individual targets ($p(x_{t+1}|x_t)$). ~

Prediction equation:

$$D(x_{t+1}|Y^t) = b(x_{t+1}) + \int (p_s(x_t)p(x_{t+1}|x_t) + b(x_{t+1}|x_t))D(x_t|Y^t)dx_t \qquad (3.3)$$

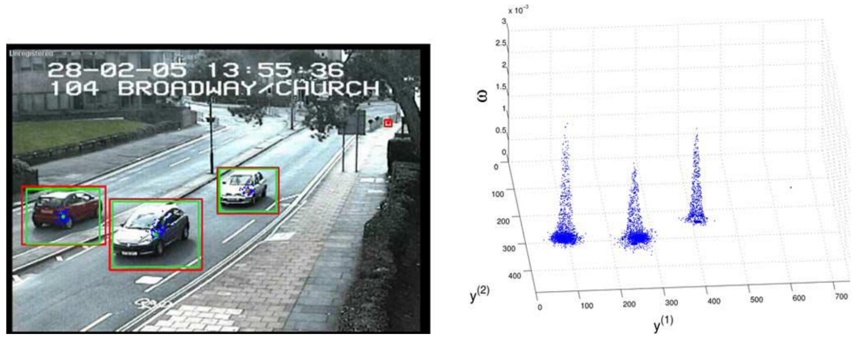


Figure 3.2 – Visualization of the particles approximating the PDH

In the update step (see eq. 3.4) the equation is composed by the probability of detection $(p_D(x_{t+1}))$, the likelihood of individual target $(p(y|x_{t+1}))$, the average number of clutter points per scan (λ) and the probability distribution of each clutter point (y) [10].

Update Equation:

$$D(x_{t+1}|Y^{t+1}) \cong F(Y_{t+1}|x_{t+1})D(x_{t+1}|Y^t)$$

$$F(Y_{t+1}|x_{t+1}) = 1 - p_D(x_{t+1}) + \sum_{y \in Y_{t+1}} \frac{p_D(x_{t+1})p(y|x_{t+1})}{\lambda c(y) + D[p_D(x_{t+1})p(y|x_{t+1})]} \tag{3.4}$$

$$D[h] = \int h(x_{t+1})\ D(x_{t+1}|Y^t)dx_{t+1}$$

Now for tracking the targets, the PHD filter is implemented using the particle filter like the one proposed by Vo et al. [11]. So the final algorithm has the following sequence of execution:

1. Detection – Detect foreground objects and the centroids ((x, y) coordinates of the object centre of mass in the image) off their blobs that are the measurement sets (background subtraction).
2. Prediction – Generate new samples for objects which are in the previous image and for new objects computing the predicted weights.
3. Updating – For each measurement use the PHD filter update step, and update weights with the new measurements.

4. Resampling – Compute the target number at next time, initialize the cumulative probability, draw a starting point and rescale the particle weights.

In order that the algorithm can track a variable number of targets in video sequences, with his approximation introduced by the PHD filter, that allows the reduction of computational cost from exponential to linear with regard to calculation of the number of targets. This algorithm still has the capability to remove non-persistent clutter, to filter missing detections, to smooth tracks, and to overcome short-term occlusions [12].

Similarly to Particle filtering, this algorithm is not suitable for the TRAPLE system because it involves intensive processing for the mathematical steps used to calculate the particles of each object.

3.4 Mean-Shift Tracking

The mean-shift tracking is one of most robust tracking algorithms for non-rigid targets, and is widely used by many computer vision researchers like Comaniciu [13][14].

This algorithm usually uses the colour information as the target feature, but it can also use texture and edge information, or any combination of them.

Colour distributions of the targets can be very robust to solve the problems of partial occlusions and appearance changes. In this case, the target is represented by a model of colour distribution as a discrete colour histogram, which is acquired from the target search region [15].

One of the lacks of this algorithm is the need to have an initial reference model to track in advance. Therefore it must have an automatic initialization process to start a mean-shift tracking.

Another problem comes from the fact that targets can change of appearance originating variations of their colour distribution, which can result in a loss of track.

Basically, what this algorithm does is a comparison between the histogram of the target model and its candidate using *Bhattacharyya* coefficients (divergent-type measure which has a straightforward geometric interpretation). So before getting the histogram of the target, a target candidate at a given location is estimated and their similarity it is evaluated. The comparison based on *Bhattacharyya* coefficients define the distance of these models (discrete distribution) as:

$$d(y) = \sqrt{1 - \rho[\hat{p}(y), \hat{q}]}, \tag{3.5}$$

where p and q are the compared models. To estimate the *Bhattacharyya* coefficients between the two models the following equation is used:

$$\hat{\rho}(y) = \rho[\hat{p}(y), \hat{q}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(y)\hat{q}_u}. \tag{3.6}$$

These comparisons are repeated for the neighbourhood position until convergence, and they are normally called as mean-shift iterations [15].

Several improvements to this algorithm have been proposed in the literature, like the association of mean-shift tracking with a particle filter initialization which facilitates the acquisition of the target models and improving them because it is difficult to consider variations on the colour distribution of the objects[15], the introduction of a scale and orientation estimation to deal with object sizes variation [16], a new spatial-colour (spatiogram) to define the object models[16], or a improvement for pedestrians occlusions introducing occlusion layers with some relations between them [17]. Another possible improvement is the tracking of fast moving objects, which can be solved with a tracker that uses spatial and colour similarity measures [18].

This algorithm demonstrates an approach for tracking video objects with good results on [13][14][15][16][17][18], but, like the previous algorithms, it is complex to manage the computational power spent in all the mean-shift iterations looking for an object model, this processing complexity goes out of purposes of this Dissertation.

3.5 Covariance Tracking

The covariance tracking, in particular the covariance tracking using model update based on Lie Algebra, is another possible approach to the video object tracking problem proposed in the literature [19].

This algorithm computes a covariance matrix of the features of the object target as the model of the object. In the current frame it can find the region that has the minimum covariance distance from the model, this allows to make that region the estimated location for that object. To adapt to variations, a set of previous covariance matrices is kept and an intrinsic mean using Lie algebra is extracted [19].

This algorithm has some advantages: 1) it embodies both spatial and statistical properties of objects; 2) it provides an elegant solution to fuse multiple features and modalities, e.g., thermal infrared (IR) and colour; and 3) It is robust against noise and severe lightning changes.

According to Porikli, Tuzel and Meer [19], this algorithm is able to find the global optimum solution unlike de local approaches such as mean-shift and particle filtering, but for the approach considered in this Dissertation it will be too much complex to deal with all the recursive covariance matrix computations.

3.6 Kernel-Based Object Tracking

The kernel-based object tracking (KBOT) is based on mean-shift iterations and uses a target representation by its probability density function (pdf) in the feature space (usually colour pdf). The pdf representation has to be discrete, enabling real-time processing. This representation only uses spectral information about the target, what can originate large variations for adjacent locations in the similarity function, so some spatial information is necessary. To regularize the similarity function, the objects are masked with an isotropic kernel in the spatial domain. So this isotropic kernel, with a convex and monotonic decreasing kernel profile, will assign smaller weights to the pixels farther from the centre of target objects and vice versa, building a model. Those pixels farther from the centre have smaller weights because of their high probability of being affected by occlusions or interference of the background [13].

After having a model for the objects, in next frame it has target candidates, whose normalized pixels are centred in other locations. The normalization is inherited from the frame containing the target model. Is used the same kernel profile, but with another bandwidth, that defines the scale of the target, like the number of pixels considered in the localization process.

Now a similarity function defines a distance among target model and candidates, that comparisons should have a metric structure for this distance. Like in mean shift *Bhattacharyya* coefficients (3.6) are used for this task (see Section 3.4).

For the target localization procedure is searched on the neighbourhood of the target model position (previous frame). Since our distance function is smooth, the procedure uses the gradient information provided by the mean shift vector. The search aims to minimize the distance between the target model and his candidate. This recursive search has a limitation of steps (mean shift procedure) [13].

Some improvements have been developed under this algorithm as the utilization of a background-weighted histogram for comparing salient parts of the objects to the background regions around them [13], a Kalman prediction to improve the number of iterations [13] or an adaptive multi-cue kernel tracking, where the object representations have cues with different importance weights increasing the robustness of tracking [20].

This algorithm is based on mean-shift iterations which, as referred previously, are too much complex for the proposed system.

3.7 Final Remarks

After presenting some of the constrains of video object tracking, some of the most popular algorithms proposed in the literature were revised, which in general require high computational power for executing their operations in real-time because they use complex mathematical operations involving matrix manipulation or suffer of recursive iterations to track the objects.

For the solution proposed, the tracking process has to be simple, without large computations or recursive operations. Therefore, some simplifications were adopted as will be seen in Chapter 4, such as tracking the objects centroids and estimating their positions in cases of locations uncertainty caused by occlusion on scene.

Chapter 4
Intelligent Video Object Tracking Solution Proposal for TRAPLE

4.1 Introduction

Looking at the objectives of the proposed project presented in Section 2.1, the proposed tracking solution has to complete the following tasks successfully:

- real-time processing of the video sequences;
- tracking of a large numbers of individual persons;
- coherent identification of the individuals along the video sequences;
- continuous localization of the identified persons in the monitored areas.

A tracking solution addressing all these tasks successfully cannot be easily achieved using the techniques reviewed in Chapter 3, at least with a reasonable computational power.

For these reasons, a simple detection method was adopted based on “low-level processing algorithm”. Nevertheless, since the detection method has to process all the video image pixels it spends a large fraction of the whole computational power when used with tracking methods. To compensate the low accuracy given by the detection module, a set of heuristic rules has been defined, making the system more intelligent in detecting events associated to the video objects tracking. The intelligence introduced in the system should be seen as “high-level processing” that gives some additional knowledgeable information about the scene without spending too much computational power.

The proposed tracking solution has to solve various problems that will be analyzed in more detail in this chapter as well as the proposed solutions to solve them.

4.2 Video Object Detection Problems

As referred before, the aim of this Dissertation is to propose and evaluate for video object tracking techniques in the context of large public environments. Therefore, the video object detection problem was left out of the scope of this Dissertation and a simple video object detection method developed in the TRAPLE project was adopted to provide input to the techniques developed under the scope of this Dissertation. This video object detection method is based on a background subtraction technique — subtraction between the input video frames and a background model — where an adaptive threshold is applied to the resulting pixel differences to classify the pixels of the video frames as background or foreground pixels. After this subtraction a morphologic filter is used to reduce the noise in the image (i.e., isolated pixels and small areas) as well to a better definition of the video objects contours, without some discontinuities (non-connected objects). The background model used to perform the background subtraction is updated at every frame processed, keeping the model updated along the time, because some variations can occur in the scene.

Since the image processing performed by the object detection module has to be kept simple, in order to save processing power for the tracking tasks, the video object detection is usually not perfect. In the following sections the main aspects affecting the quality of video object detection task are briefly described.

4.2.1 Background Noise

Background noise can be originated by various reasons, e.g., this noise can be caused by environment vibrations (e.g., a train arriving) or illumination variations (e.g., natural or artificial light) that affect the camera or simple variation of light in the scene. This type of noise is minimized by the use of morphologic filters but can be insufficient to a good accuracy in detection. This noise can be reflected in the tracking process, because some objects that do not exist can be tracked when is only noise in the background. Large amounts of noise can severely decrease the effectiveness of the tracking due to two main reasons. 1) The tracking process slowed down because it has to track much more objects, objects that do not exist; 2) The other is that after some time this noise can becomes accumulated making the tracking of some object impossible because the noise is dispersed around the entire image, e.g., like a lot of reflections being detected as foreground.

An example of some noise introduced in the background after a few minutes processing can be seen in Figure 4.1, which is marked on the left image with red boxes.

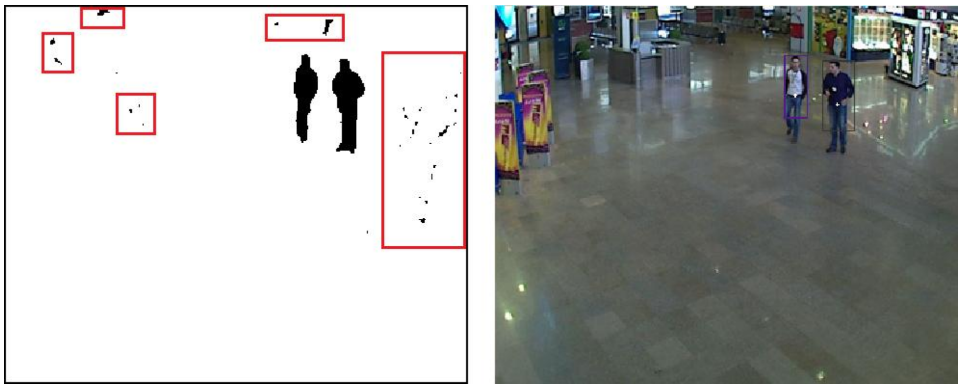


Figure 4.1 – Background noise example

4.2.2 Object Segmentation Imperfections

Although the video object detection method used does not provide the exact contour of the objects, for tracking purposes this is not a significant problem, since what is really important is the location of the object in the image and not its perfect contour.

One of the biggest causes of the segmentation imperfections is a phenomenon usually called camouflage [4]. This happens when the background and foreground colours are similar (e.g., a person wearing clothes that have similar colours to the scene background), video object detection becomes very difficult or even impossible using only digital cameras. This happens because the system is not able to detect variations of colour in the background.

One example of this issue can be seen on Figure 4.2 marked with red boxes on left.

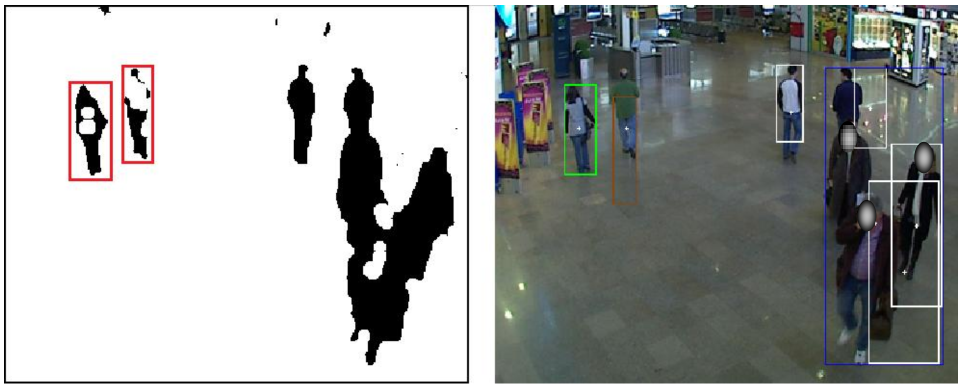


Figure 4.2 – Object segmentation imperfections example

4.2.3 Shadows

The illumination of the scene is another source of difficulties for a video object tracking system, notably due to shadows that appear near moving objects or even near some static obstacles in the scene. These variations in the intensity of the image pixels belonging to shadow regions will result in the detection of objects larger than they really are or phantom objects that the system tries to track. Needless to say that, ideally, the system should operate in a controlled environment with artificial lights and that natural light is an enemy, because is not constant and can bring many changes in the image pixels, which makes the detection of objects very difficult.

These situations are critical in exterior scenes that are not the cases that this project wants to cover. Despite being an interior space it also have shadows, but these are caused by indoor lights that are quite stable. So it will have little variation and can be treated by the tracking algorithm in a simple and efficient way as can be seen in Section 4.8.5.

As can be seen in Figure 4.3, the shadows could be a problem for video object detection.

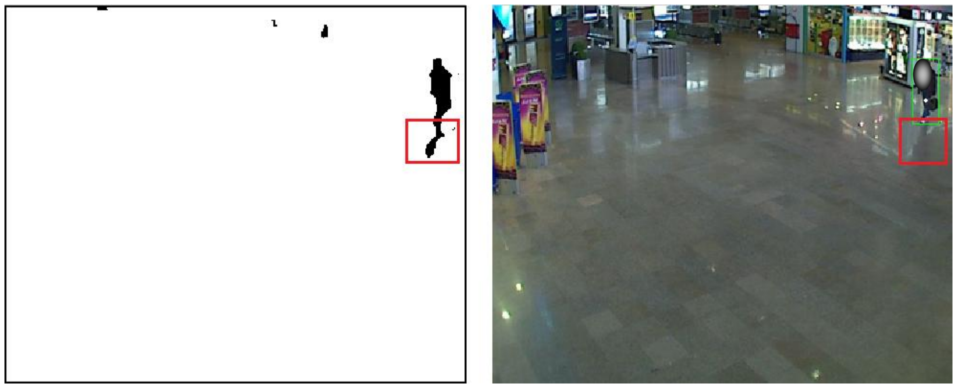


Figure 4.3 – Moving objects with shadows

4.2.4 Sudden Changes of Scene

Depending on the monitored environments, during some periods of time large variations in the background of the image can be observed. These variations can be large objects moving on the scene, for example on a train station, a train reaching the platform. These variations may be only some natural light entering in the space watched, causing a large variation in the pixels of the image on the illuminated space.

These situations are very difficult to handle since it is not viable to consider all possible scene variations. A possible solution to this problem is to reset the model used for the background when large scene changes are detected, and initiate a new training period to compute a new background model. These situations will result, typically, in loss of information about the tracked objects. However, if the background model is not reset, many abnormal situations may be detected by the tracking module afterwards.

4.2.5 Non-connected Objects

A simple background subtraction with some noise could result in broken objects like one person that is detected into two objects, e.g., upper-body and legs. Situations like this can be difficult for a tracking system because it can lead to the tracking of two separate segments of the person as two different objects or it can even classify these two segments as noise if they are very small. This problem may be reduced applying morphologic filters to the background subtraction; this reduces the noise in both the object and the background. Despite the improvements that morphologic filters can bring, such situations can always occur, although less frequently. Figure 4.4 illustrates this situation on the left image (labelled with a red box) where a person is detected as various non-connected objects.

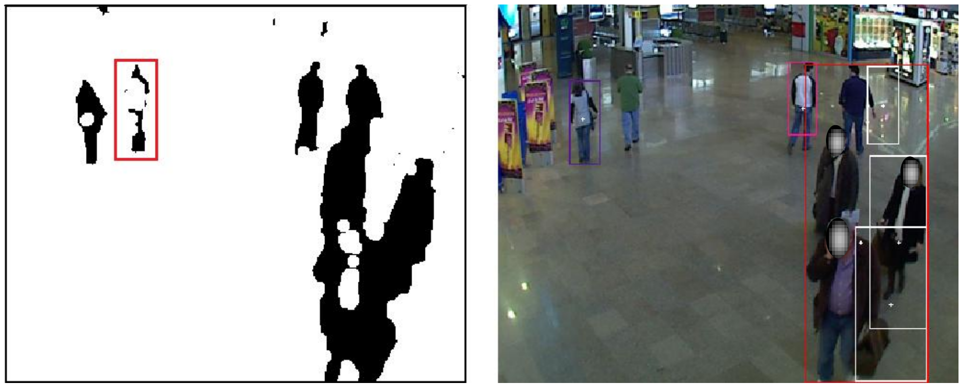


Figure 4.4 – Non-connected objects example

4.3 Tracking Problems

After the moving object detection in the scene for each video frame, the system should establish a correspondence between the moving objects in the actual and previous frames, i.e., the tracking of the moving objects. Video object tracking exhibits also some difficulties due to the non-ideal scene characteristics. These difficulties are described in the following sections.

4.3.1 Video Object Size Variation

Depending on the camera position relatively to the monitored scene and the video object motion relatively to the camera referential, video objects will be viewed with different and varying depths, this means that the objects will have different sizes depending on the depth at which they are on scene (see Figure 4.5). This may hinder the tracking of video objects because it is simpler to make the tracking of an object if it maintains a constant size and appearance. This situation is simpler because it makes possible a linear estimation of the size and positions of the moving objects. With the object size varying it does not allow using standard thresholds for select what is an object or what is noise and makes difficult the location on map, which is important in the project context.

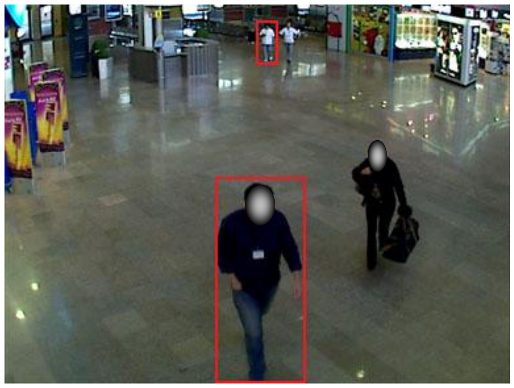


Figure 4.5 – Video object size variation with the depth of the camera

4.3.2 Video Object Motion Variation

Due to the perspective view of the camera, the speed of moving objects in the image plane will also vary depending on the distance of the object to the camera. While the objects that are away from the camera move a few pixels per second, objects very close to the camera can move very fast, entering and leaving the scene in consecutive frames.

Also the objects at same depth can have different speeds in the video images, this because two persons in scene may move at different speeds in real space, one almost a stop and another running.

This situation will bring some variation in the distance difference of an object between two processed frames, so it is impossible to define a standard difference not only because of the different depths in image, but also because of the motion variation. In cases of fast movements the identification of the objects can be lost. That can make the tracking more difficult if the processed video has a low temporal resolution.

4.3.3 Video Object Occlusions

Situations where the objects that are being tracked are not visible to the camera all the time are usually called occlusions. They may be caused by two or more moving objects crossing, leading to a new object containing several individuals, as can be seen in Figure 4.6. This situation is more common if the angle between the camera and the floor is close to zero (angle marked as α in Figure 4.7), this mean that, for persons walking closer to the camera, occlusions between them will not be so common. Therefore, the greater the angle of inclination of the camera with the floor is, the lower are the situations of object occlusions. This means that the camera should be located ideally in a high position to achieve a large covering area while reducing possible object occlusions. Although this situation is favourable to reduce occlusions, the objects will be smaller, which can become a problem in cases of noisy environments, as illustrated in Section 4.2.1; in these cases small objects can be considered as image noise and are not tracked.

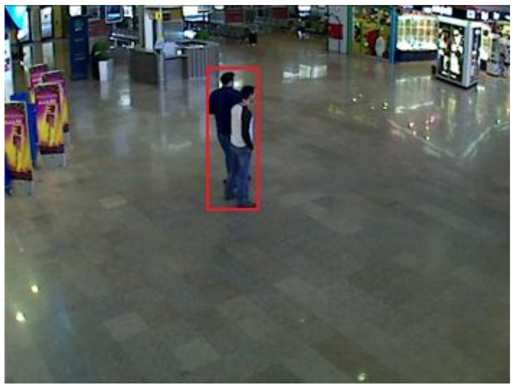


Figure 4.6 – Occlusion between two video objects

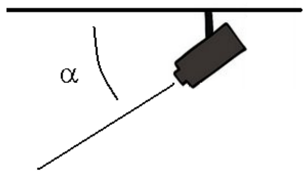


Figure 4.7 – Angle of inclination of a surveillance camera

Obstacles belonging to the scene can also generate occlusions of video objects, which lead to tracking failures of these objects because they cease to be seen for a few moments while they are occluded (see Figure 4.8).

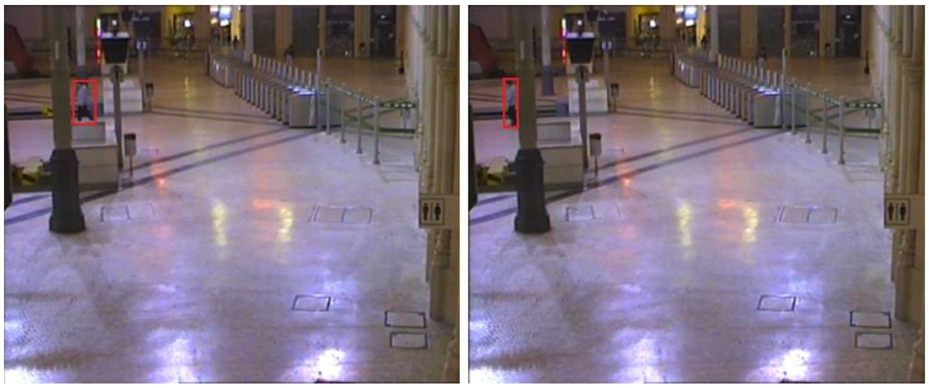


Figure 4.8 – Video object occlusion by an obstacle

4.3.4 Incoming and Outgoing Objects

Points of entry and exiting of people in the scene can create ambiguities in the tracking system. This is because people that cross these points can cause switching of identifications. Figure 4.9 illustrates this situation showing a video object (red rectangle) that occludes various persons exiting the scene making impossible their detection. In particular when a group of persons enters in the scene is difficult to count them and when a group is walking near these points some of the persons inside the group can exit the scene without being detected.

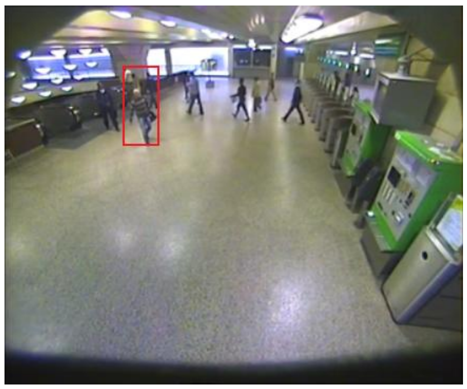


Figure 4.9 – Entering and exiting regions causing object occlusions

4.3.5 Video Object Identification Management

One of the most difficult tasks to accomplish in a tracking system is to maintain identity coherence for the tracked objects along the time. This is one of the major objectives of this Dissertation. The set of difficulties analysed in the preceding sections, which sometimes materialize into object visibility losses, clearly stress this fact. Adding to the fact that the identifiers have to be tracked in real-time the management of the identifiers is harder.

To do this task correctly and in real-time one simple method of tracking the centroids of the objects (which do not vary a lot along the time as explained in Section 4.6.3) was used. This type of tracking associates the objects positions with the old positions seen, but this works fine only when the objects walk without being mixed with others in the scene. In the cases of mixtures of objects or occlusion, the correspondence between objects and identifiers must be done using algorithms which compare the characteristics of objects such as colour, texture or contour. Some of these methods will be used in the proposed system and are explained with more detail in Sections 4.6, 4.7 and 4.8.

4.3.6 Large Volumes of People

In a crowded scene it is very difficult to track a video object because a large number of objects are mixed together, with many areas of occlusion which makes almost impossible to detect any object pattern. A tracking system such as the one developed in the context of this Dissertation should have a level on the flow of people to be followed. Reaching this level the system starts to do not draw any conclusions with the processing of such images because is unnecessary. The chaos generated by a crowd scene will bring many errors such as object that are not identified, groups of people with identifications of wrong person and others.

4.4 Video Object Tracking Objectives

Considering the various video object tracking problems identified above, the work developed under the scope of this Dissertation was directed towards the development of a simple and fast tracking system in order to be able to track large groups of people with reasonable computational power and without too much loss of information regarding the detection and identification of objects.

In the tracking results a continuous and smooth location of many moving objects has to be extracted from the video images, so the tracking gives their trajectory while they are in the vision of a camera. In should be possible to get full paths of the objects along the monitored scenes with the use of multi-cameras and studying the movement of people in the environments.

The basic concept of this tracking system is to recognize where the moving objects are in the monitored scenes, ideally in real-time, without worrying about a perfect detection of their contours or what they are doing, focusing the system to know how much people are there and where. Notice that, typically, the main goal in similar systems is the opposite, i.e., to know information about a few objects but with a good accuracy. To achieve the desired goals, whenever situations where the system have doubts about the objects locations occur, some estimation algorithms are used to maintain as possible the knowledge about these objects.

In summary, the principal objectives of this proposal are:

- Track a large number of individual persons;
- Maintain the identification of the individuals along the video sequences;
- Make the image processing as simple as possible without compromising the object detection;
- Make estimations about objects positions in case of doubts (occlusions);

After reviewing the main proposes to the tracking system in next section will be presented a possible architecture to complete the system proposes.

4.5 Video Object Tracking Architecture

To solve the complex problem that involves tracking, an intelligent system was proposed in the context of this Dissertation. This solution showed good results in some typical video sequences as shown in Chapter 5. This section presents the description of the proposed tracking system.

Taking into account the previous constraints, a simple, yet intelligent, tracking method is proposed to track video objects. As explained below, in Section 4.6, after processing each video frame it is possible to define intelligent rules that for the humans are obvious with respect to their field of vision, but for a machine if such rules are not explained, they simply do not recognize them as such. This set of rules will be explained throughout this chapter and in sections proceeding to Section 4.6.

Figure 4.10 presents with red boxes evolving the “low-level” modules a “high-level” structure of the proposed intelligent tacking solution, where the sequence of major tasks that the system performs over time for each frame of a surveillance video sequence is shown.

This sequence of tasks starts with the video objects detection, which gives as input to the tracking module the image regions where are the moving objects in scene, this module, however, is out of the scope of this Dissertation. The video object detection methods used were developed in [21].

The proposed video object tracking architecture is divided into the following four main blocks:

- *Image Information Extraction Layer* – This layer processes the images as raw data (pixels); after knowing what objects it has to track, the system extracts some of their characteristics generating a different object model for each detected object. The object models could be compared at each frame, which objective is to compare current models with the models of the last frame, allowing the identification correspondence;
- *Events Detection Layer*– This layer handles the events that can occur between video objects, a camera sees objects grouping and ungrouping over the time, thus the system should have some intelligent functions that capture this event types maintaining some coherence between the identification of the objects and trying to know their locations;
- *Intelligent Layer* – This layer introduces a number of intelligent rules that could be observed in the viewed space and how objects can move in this space; some information about the objects sizes in the image can be extracted from the space seen by the camera. This information is used to correct the objects sizes that are sometimes found with wrong sizes in wrong places;

- *Communication Layer* – This layer was implemented only to integrate this tracking system in the TRAPLE project, where the position of the video objects are transformed from the image coordinates to the coordinate space on the floor of the monitored space, which may allow the use of multiple cameras over a space, with consistency of space between them; after the spatial transformations all the information needed are sent in MPEG-7 messages.

All these tasks are executed for each video frame processed. After present the "high-level" concepts of the tracking system could be seen in more detail each block mentioned. Figure 4.10 shows the detailed architecture of the proposed video object tracking system. In the next sections all the modules of this architecture will be explained with more detail.

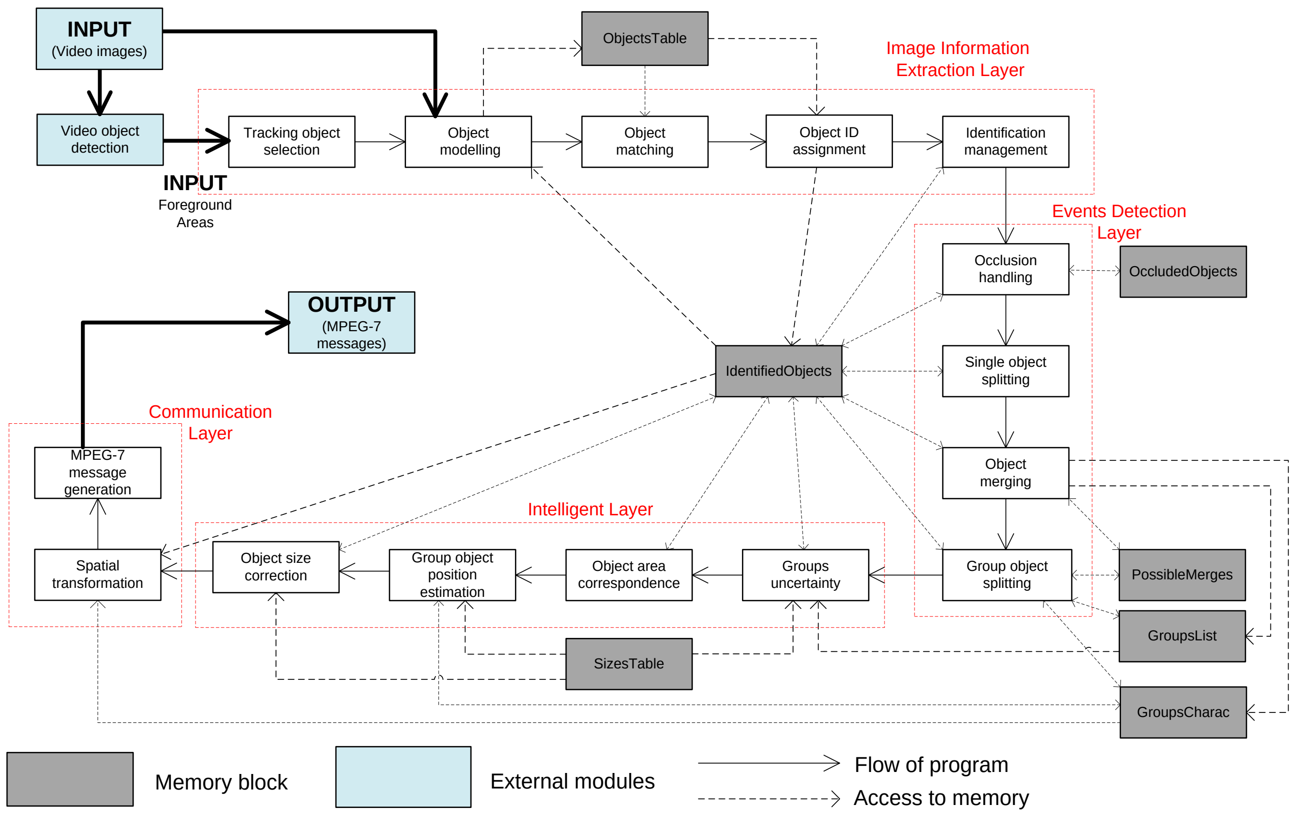


Figure 4.10 – Proposed video object tracking architecture

4.6 Image Information Extraction Layer

This layer will detail how the information needed for tracking video objects is extracted from the raw videos, e.g., positions, and delimitations of objects, and how to manage the objects identification.

4.6.1 Tracking Objects Selection

From the video object detection module, the tracking system receives as input a binary image for each processed frame indicating which pixels belong to moving objects (label 1) and which pixels belong to the background (label 0). An example of an image output of the detection process can be seen in the Figure 4.11.

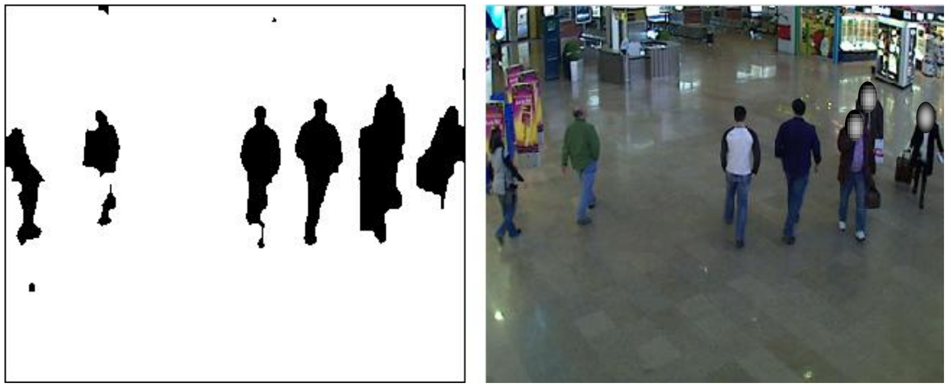


Figure 4.11 – Video object detection output example

To select the objects to track a labelling algorithm is used at each frame searching the foreground zones (blobs) labelling all of them. These labels are assigned according to the spatial location of blobs in each frame, assigning the label number to the objects from left to right and from top to bottom [22]. One little sample of this algorithm can be seen in Table 4.1 and Table 4.2, on the Table 4.1 are a video detection output sample of an image with 10x10 pixels, and on the Table 4.2 are the same image with the labels assigned to each object.

Table 4.1 – Video object detection output sample (10x10 pixels)

1	1	1	0	0	0	0	0	0	0
1	1	1	0	1	1	1	0	0	0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	0	0	1	1

Table 4.2 – Video object labelling sample (10x10 pixels)

1	1	1	0	0	0	0	0	0	0
1	1	1	0	2	2	2	0	0	0
0	0	0	0	2	2	2	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	3	3	3	0
0	0	0	0	0	0	3	3	3	0
0	0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	3	0	4	4
0	0	0	0	0	0	3	0	4	4
0	0	0	0	0	0	0	0	4	4

Since the detection method used is not perfect, some care should be taken regarding the video objects labelling. As can be seen in Figure 4.11, small isolated blobs may appear in places where there are no moving objects. Additionally, some objects may appear as non-connected areas (e.g., the man with a green pullover in Figure 4.11), which may lead to the false detection of two or more different objects. To tackle these problems the following approach is proposed:

- Establish a minimum object size in terms of the number of foreground pixels (minimum_object_size_threshold);
- Ignore objects smaller than the threshold established.

Objects with a size below the threshold established for that region of the image are ignored, because it is assumed that they correspond to noise or small irrelevant objects rather than a person.

One problem with this approach is that the area of each blob varies along the image, so the threshold established should be different in the various image regions, like the farther and closer places to the camera.

With respect to the problem of non-connected objects, this approach will lost track of objects when they are detected as multiple non-connected fragments and none of the fragments of the object is higher than the minimum object size threshold. As will be seen in Section 4.6.5, the tracking of an object can be lost with duration of less than three frames without losing its identification, but in a constant case of non-connected objects this approach will simply ignore the fragmented objects.

To some demonstrations of the system some assumptions are taken, in all the surveillance cameras the people gets little on image when they move to a farther location. If the surveillance camera used to track people is installed horizontally when the persons walk to farther locations, in image they move to the top of image and vice versa (see Figure 4.5), with some linearity.

To get acceptable thresholds to the object sizes (number of pixels), and assuming that the camera is installed horizontally, the following approach was followed:

- The image is divided vertically in five regions;
- Each region receives a different threshold based on image samples (a minimum number of pixels is defined for the near region and for further regions this value is multiplied for a decreasing factor).

After this kind of “noise filtering” is made, the labelling of objects will be reordered again without the filtered labels. Now the system only has the objects that are considered to be of interest to track with their respective labels and areas on the image. The process of selection, although can have some noise, it should not be too much, because it may take too much time doing the labelling and filtering of the blobs that do not have interest to track.

All this process of selecting objects to track is implemented in the *Tracking object selection* module (see Figure 4.10), where the system have a binary map input with the foreground areas and an output of one array with each object detected with all the position in scene of the object pixels.

4.6.2 Object Modelling

In order to be able to track the various objects in the scene it is necessary make a correspondence between each object in the current frame and the same object in the previous frame. Therefore, in order avoid ambiguities in terms of object correspondence, it is proposed in this Dissertation to characterize each object by a set of features that should be sufficiently discriminative while being simultaneously easy to compute. The module responsible for extracting the relevant characteristics of each detected object is called *Object modelling* (see Figure 4.10).

After knowing the locations of the video objects that have to be tracked, the system uses the current video image (i.e., RGB components) to generate the object model In this context, the set of features used to characterize each object (Object model) is the following:

- Colour – histogram of the RGB components of the object.
- Size – the number of foreground pixels of the object.
- Contour – bounding box that contains the object.

These features could be useful to track the objects.

The system has a table of objects for each frame with each object model. That will be used to compare the object in new frames with the objects that are previously identified and only should be accessible during the processing of the current frame and are saved in a matrix called *ObjectsTable*, which structure can be observed in Table 4.3. So this module receive the video objects locations in pixels and the correspondent video image and give as output this table with the current objects model.

Table 4.3 – *ObjectsTable* structure

Object_ID	Centroid(x,y)	Min(x,y)	Max(x,y)	Number of pixels	Histogram
1	…	…	…	…	…
…	…	…	…	…	…
N	…	…	…	…	…

Each line of the table represents a video object, with the following characteristic fields:

- Object ID a number that identifies the object temporarily in the current frame;
- Centroid(x,y) – the mean point of the pixels of a blob in x, y image coordinates (as calculated in eq. 4.1);
- Min(x,y) and Max(x,y) – the minimum and maximum values that define the bounding box of the object;
- Number of pixels – the number of pixels that the object occupies in the image, is useful to make calculations regarding the density of objects;

- Histogram – One important characteristic of a video object is its colour distribution captured through a histogram. The histograms used can be in greyscale or RGB colour spaces. The colour space used depends on the wanted precision. In this Dissertation the RGB colour space has been used, because it has demonstrated better results in people differentiation the other tested colour spaces. The tested colour spaces testes were the greyscale and the luminance component.

4.6.3 Object Matching

After identifying all the objects that have interest to track, then the system must provide some method of how to achieve a correspondence between different frames of the identifications assigned to the video objects, ensuring their identity over time.

One of the possible methods to perform this task and that was used in this Dissertation is about the calculation of the centroids of the video objects; this is the average position of the sample pixels of the observed object, both in the X-axis and the Y-axis. As can be seen in the following formula:

$$C = \left(\frac{\sum_{i=1}^{n} x_i}{n}, \frac{\sum_{i=1}^{n} y_i}{n}\right) \quad ; \tag{4.1}$$

where "n" is the number of pixels of the object blob and x_i and y_i are de index of each pixel of the object in the image given by the detection module.

Looking at consecutive video frames it can be seen that the centroid of each object in the image exhibits, typically, small changes from one frame to the next one. Additionally, from one frame to the next the centroid of an object is, typically, the closest centroid of its last centroid position even when the object has close neighbours (this fact can be observed in Figure 4.12 where the centroids are marked with crosses). This means that it is possible to track video objects through their centroids using a matching process for the closer centroids maintaining the objects identification over the video frames. An example for a better understanding is provided in Figure 4.12, where two consecutive image samples of a video recorded at 8 fps are presented with three moving objects with its centroids marked with white points. It can be seen from this figure that from one frame to another it is difficult to understand the movement of the centroid; at a first look they seem to be in the same place. Table 4.4 shows the centroid distances between the objects in the two consecutive video frames of Figure 4.12.

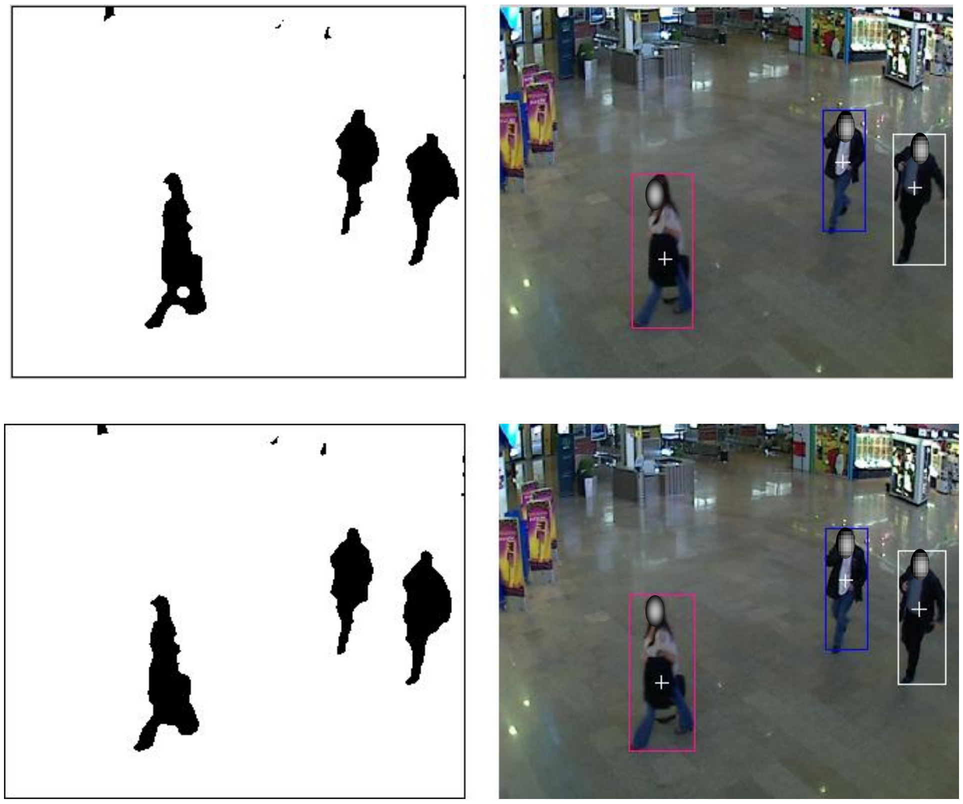


Figure 4.12 – Centroids tracking: two consecutive frames of a video at 8 fps

As can be seen in Table 4.4, the minimum distance (marked in gray) in each column/line provides a good indication of the correspondent video object, i.e., the correspondence between the actual identification of the video object – VO(*t*) – with its old identification – VO(*t*-1). Notice that this table is not usually symmetric because the upper right cells of Table 4.4 correspond to a different time instant of the lower left cells.

Table 4.4 – Distance in pixels between the video object centroids of two consecutive frames (Figure 4.12)

VOs(*t*-1) / *VOs*(*t*)	Pink Object	Blue Object	White Object
Pink Object	9.1	165.8	210.9
Blue Object	153.6	3.3	62.3
White Object	197.1	57.4	5.3

To manage the objects identification by their centroids distance between two consecutive frames, a maximum distance to a possible correspondence (centroids_distance_threshold) should be established, preventing a correspondence of distant objects. This maximum distance should not be static because the objects have different sizes and motion. Normally, the small objects on image (farther from the camera) have small displacements on the image and bigger objects (closer to the camera) have bigger movements. Then small objects should have small differences on their centroids and vice versa. This distance should be adjusted for a video image resolution.

The temporal resolution of the processed video is another factor that influences the centroids_distance_threshold. With lower temporal resolutions a video object can exhibit larger displacements between frames, therefore, the maximum distance between centroids of two consecutive frames must be higher. In the tested video sequences (see Chapter 5) with CIF resolution (352x288 pixels) (e.g. Figure 4.12) the centroids_distance_threshold used was a distance in a radius of 50 pixels. This value was not constant being reduced with the depth of the image. The image is divided vertically in five depth regions and the value decreases by 10% for each increasing depth region.

The *ObjectsTable* contains the centroids of the current objects in the scene, which are compared to the centroids of the objects in the previous frame that are stored in the *IdentifiedObject* table (defined in the Section 4.6.5).

4.6.4 Object ID Assignment

Although the centroid-based correspondence is relatively robust in maintaining object identification coherence, it may happen that the same object can have two identifications associated when two moving objects are very close. This may happen because in the table of distances, shown in Table 4.4, one object may exhibit two or more minimums in the same row if two objects are at the same distance or in the same column if two objects move to near places.

To solve this problem, the table of comparisons is again processed to correct repeated identifications in the objects (collision), obtaining the minimum distance between the centroids of the objects in collision. The old identifiers which were not assigned in these cases are saved and later are made a new search to this identifier be given to an object that has not yet its corresponding identifier. This identifier is given only if their centroids distance does not exceed the centroids_distance_threshold

When an object has no previous identifier that can be assigned through the described methods, then a new identifier will be assigned to it. This process is accomplished by the *Object ID Assignment* module. Yet in the same module, after all the processes of attribution of identifiers, the object models present in *ObjectsTable* are transferred to the *IdentifiedObjects* table with their correspondent identification and his model. The *IdentifiedObjects* will be described in next section.

4.6.5 Identification Manager

When an object is considered as identified, a set of features can be computed for it, e.g., colour, position or if it can correspond to more than one object mixed, as seen in Section 4.6.2. This set of features will help to distinguish video objects in the future.

The *ObjectsTable* information, after been confirmed, will be copied to another table (*IdentifiedObjects*) that will be accessible for the next frame. The temporal features that are only needed to the tracked objects along the time could be inserted in the system in this transition of values to the *IdentifiedObjects*. As seen in Table 4.5 (*IdentifiedObjects*), and comparing with Table 4.3 (*ObjectsTable*), some new fields appear in this table with these needed features.

The first field to change is the identification number, in *ObjectsTable* (see Table 4.3), the identification numeration (Object_ID) used is only valid for one frame. In this new table, the identification numeration changes from Object_ID to VO_ID (**V**ideo **O**bject **ID**entifier), which will be an identification that is unique to one object along all the video sequence analyzed. These identifications could be used for other modules external to the tracking module because they are unique.

One new field is the *Presence variable,* this field is computed by the *Identification management* module (see Figure 4.10). This field should tells the system if the video object appears for the first time in scene, if the object has been seen in previous video frames or if it is not possible to detect this object in the current frame. For this task, a simple integer variable is used, which can take five different values:

- -1 - when a video object is seen for the first time in the current frame;
- 0 - when a video object has been seen in the previous frame;
- 1 - when a video object disappeared from the previous to the current frame;
- 2 - when a video object has not been seen in two consecutive frames;
- 3 - when a video object has not been seen in three consecutive frames.

The *Presence variable* value acts as a counter to indicate when an object does not have more interest to be tracked, i.e., when the value is higher than three the objects can be deleted from the system. When the *Presence variable* takes the value 3 the *Identification management* module automatically deletes the video object from the *IdentifiedObjects* table. This video object is deleted because in three frames some other object could take its place in the video image, which may result in swapping of identifications if the centroid of the other object is close to the centroid of the disappeared object. This delay deleting the tracked objects will allow handling some object detection fails without losing the objects identification, e.g., in cases of non-connected objects or occlusions.

Other new field in the *IdentifiedObjects* table is the Group identification. This field has a default value of zero, because it is very difficult to take conclusions concerning a given object being an individual person or a group of persons by analyzing one single frame. As it will be described in the Section 4.7, sometimes there is a high confidence that a given video object represents a group of persons instead of a single person, namely when persons are seen merging, so in these cases this field will be changed to a reference identification of a group.

The list of possible groups present in the system is stored in the *GroupsList* that is created with the existing groups in the scene. When a group is detected, a new identifier is assigned to the video object as a group on that list. If a video object classified as a group disappears during three frames it will be deleted from the *IdentifiedObjects* as in the single objects and all its references in *GroupsList* are also deleted.

The *Lifetime* field is a counter that counts how long an object is in the scene; it represents the number of processed frames in which the object is present. Therefore, at each processed frame the *Lifetime* field of all video objects present in the scene is incremented. This field only gives information about how long is the object in monitored space.

The *Possible Group* field is set to zero by default as happens with the *Group Identification* field. However, this field only changes when the *Group Identification* field is zero, which means that does not know if the video object is a group of persons. This field will be set to one in other module (*Group estimation* in Figure 4.10). As the name of module indicates, when the value is set to one the video object may correspond to a group of persons. The group handling tasks are described with more detail in Section 4.7.

All the other fields (*Centroid(x,y)*, *Min(x)*, *Max(x)*, *Min(y)*, *Max(y)*, *Number of pixels* and *Histogram*) are copies of the object model from the *ObjectsTable* at the current frame.

Table 4.5 – *IdentifiedObjects* structure

VO_ID	Centroid(x,y)	Min(x,y)	Max(x,y)	Number of pixels	Presence variable	Group identification	Lifetime	Possible group (estimation)	Histogram
1	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…
N	…	…	…	…	…	…	…	…	…

After the features description can be concluded that in this solution a video object can be classified as three different objects:

- Individual person (IP);
- Group of known persons (GKP);
- Possible group of unknown persons (PGUP).

The video objects do not have always the same classification, e.g., when the system does not see the objects when they enter mixed in the monitored scene (PGUP) or when the people are seen by the system being mixed with others (GKP). To improve these classifications a set of methods which detect the changes of classifications of the objects can be applied to the tracking system. These methods are described in the next chapter.

4.7 Events Detection Layer

People moving around in the video scene do not, usually, exhibit constant motion. Therefore, they can be mixed in the same detected video object (same blob). In some cases people entering the scene are already mixed (i.e., a PGUP), but in other cases people are seen as a single video object (IP). In these cases it is possible to detect when this single person goes mixed with other person(s) (i.e., an IP merged into a GKP), or when they become separated from the group (split). These three possible operations between video objects in two consecutive frames are schematized in Figure 4.13.

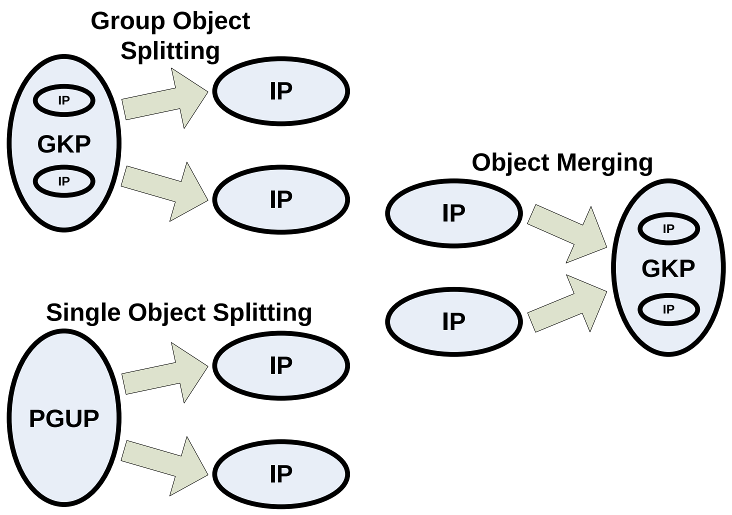


Figure 4.13 – Merge and split operations

For these operations to be caught, some heuristics have to be applied at each video frame, organizing necessary information about the objects. The heuristics applied are explained in the next sub-sections.

Before catch the merge and split events between objects the group management concept used has to be introduced. A group of persons is considered when in a video image various persons are detected as the same blob. That kind of video object cannot be treated as an individual person because these objects can have more than one person identification, so this objects are marked in the *IdentifiedObjects* table (see Table 4.5) with Group identification field as seen in Section 4.6.3. This group identification does a kind of hyperlink to other table called *GroupsList* containing the video objects identifications that are inside the group where the group identification is the index of this table. An example of this table is shown in Table 4.6 where the video object marked with group identification 1 contains three persons. These have been seen with the VO_IDs 2, 3 and 5 before being merged into a group.

Table 4.6 – *GroupsList* structure

Group ID	VO_IDs contained
1	2 3 4
...	...
5	12 18

To make possible future identifications of the persons moving as a group, their object models should be stored in order to be able to compare them with future objects. Therefore, another table is necessary to store all the characteristics of the object models contained in the video object groups. This table, called *GroupsCharac*, allows the identification of the individual video objects after they leave the corresponding group. The *GroupsCharac* table has the same columns as the *IdentifiedObjects* table, except the fields *Lifetime* and *Possible Group* that are used to store the last motion vector – Vector(x,y) on Table 4.7 – of the object that entered the group; this vector will allow performing some estimations of the object location within the group. All the other fields are copied from the *IdentifiedObjects* table when the object entered in the group. The fields that correspond to the object location (centroids, minimums and maximums) are used with the same information but these will be updated with the estimations done while the object is inside the group as will be seen in Section 4.8.3. The structure of this table can be seen in Table 4.7.

The *GroupsCharac* table should have all the object models of the objects inside the *GroupsList* table (Objects IDs contained).

All the tables used can be seen on Figure 4.10 as memory blocks.

Table 4.7 – *GroupsCharac* structure

VO_ID	Centroid(x,y)	Min(x,y)	Max(x,y)	Number of pixels	Presence variable	Group Identification	Vector(x,y)	Histogram
1	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…
N	…	…	…	…	…	…	…	…

4.7.1 Object Merging

This module is designed to capture the object event of merging, trying to manage the entrance of persons in groups. It receives the list of objects in scene and a list of possible merges to that frame (see description below).

This event represents the union of two or more video objects in the same blob. The human observer can easily associate and individualize two or more objects that have been merged into a single object. However, for the computer it is hard to understand if a blob corresponds to one or more persons mixed. In these cases the identification assignment of the objects cannot be done simply by their centroids because they are lost when the objects enter in a group.

To capture the merge events and make the correct object identification association inside the group it cannot be done only using information of the current frame.

Based only on the current frame object detection data, obtained through the Video Object Detection Module (see Figure 4.10), the system is only able to recognize that a new object appears in scene (a group). To make the association of the group to the object identifications inside it, the object merges have to be estimated based on data from the previous frame.

If in the previous frame, a group of possible merges (as described below) is envisaged (with the object identifiers that could be merged at the current frame), it is possible to make an association of existing object identifications to a group. This situation occurs when two or more objects disappear and a new bigger object appears in the same image region. In this case, the system still checks if the mean of the centroids of the disappeared objects is not farther than a threshold (maximum_merge_threshold) from the centroid of the new object. Since the system knows which objects have disappeared and which identifiers could be merged, it only needs to verify if the disappeared objects are in this estimated group list. This list is stored in the *PossibleMerges* table (see Figure 4.10). Table 4.8 shows an example illustrating the structure of this table.

Table 4.8 – *PossibleMerges* structure

Object 1	Object 2	Distance (pixels)
1	2	4
6	9	10

Two objects are inserted in the *PossibleMerges* table if the bounding box of each object is within a minimum distance of pixels (possible_merge_threshold), whose value can vary with image resolution and the position of the object on the image as the centroids_distance_threshold in Section 4.6.3. The distance between the objects is stored in this table because it will be used to decide which objects will be merged in the next frame.

The algorithm used to detect possible merges performs one iteration at each *PossibleMerges* line and verifies if the objects of this line have disappeared by inspecting the *Presence variable* in the *IdentifiedObject* table. For the lines where both objects have disappeared it will be verified if some of the disappeared object is a group. This verification is needed because if one of the video objects is a group is not necessary to create a new group; only a new object is added to that group.

If only one of the objects has disappeared and it is not in the limit of the monitored area, it is necessary to verify if the size in pixels of its possible merged object has grown more than a threshold of pixels (minimum_merge_pixel_threshold). In the affirmative case, these object identification will be replaced by a group identification and make the association of the merged objects to the group identification. In this case it is necessary to verify if some of the merged objects are a group for the same reasons explained in the previous case.

Figure 4.14 illustrates in a flowchart how this task is implemented.

On Figure 4.15 can be seen an object merging process in two consecutive frames. This Figure (top), shows three objects with three different VO_ID (the objects are marked with bounding boxes of different colours). The information extracted from this frame to the *IdentifiedObjects* table is represented in Table 4.9 – the green bounding box corresponds to VO_ID 1, the red to VO_ID 2 and the blue to VO_ID 3. It can be seen that all the objects are detected as IP in the field group identification that maintains the default value zero.

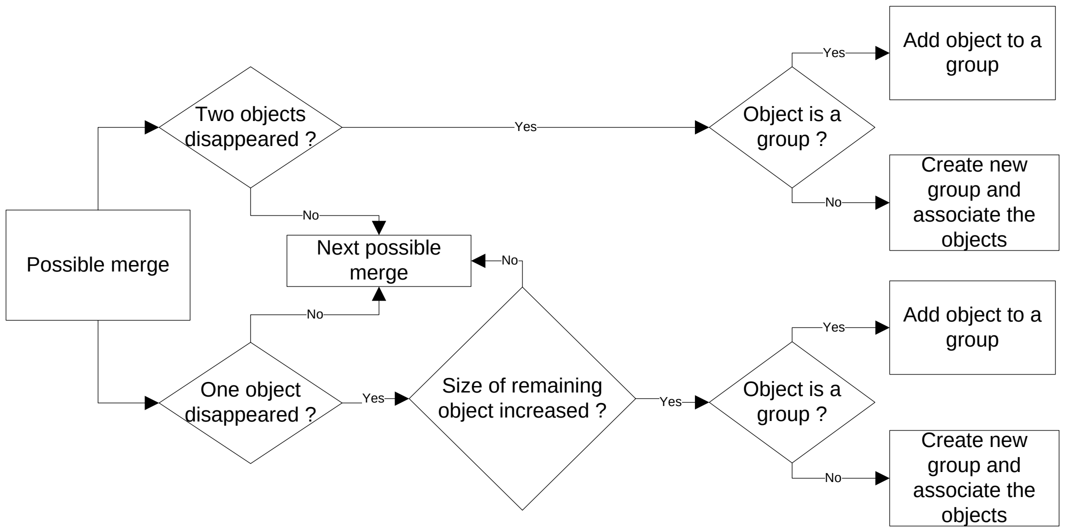


Figure 4.14 – Merging verification flowchart for each possible merge

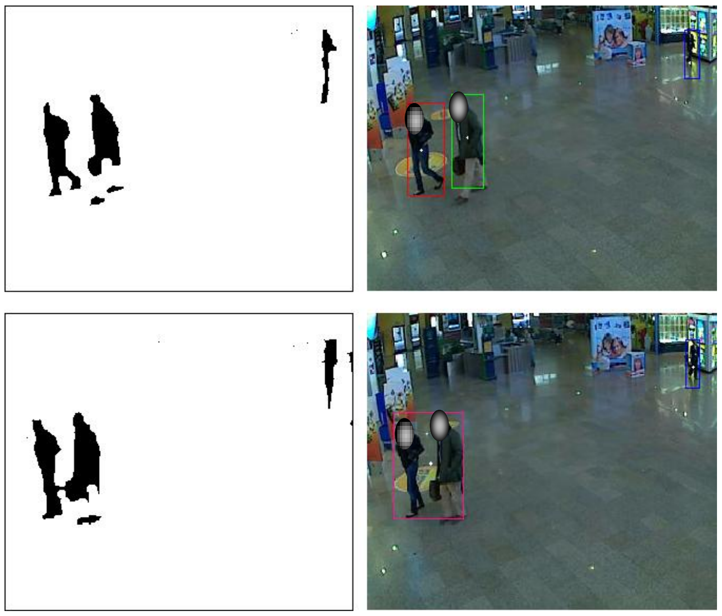


Figure 4.15 – Object merging example

At the next frame (see Figure 4.15 bottom) the objects with the VO_ID 1 and 2 have been merged to the same detected object. In the video image they are represented by the same object but the information that there are two objects inside that blob is preserved in the table.

Table 4.9 – Object merges example (*IdentifiedObjects* – a)

VO ID	Centroid(x,y)	Min(x,y)	Max(x,y)	Number of pixels	Presence Variable	Group Identification	Lifetime	Possible Group (estimation)	Histogram
1	(45,148)	(30,20)	(59,101)	1480	0	0	89	0	…
2	(84,141)	(69,28)	(102,96)	1820	0	0	72	1	…
3	(328,54)	(322,26)	(335,93)	528	0	0	36	0	…

As are known through the *PossibleMerges* table generated in the previous frame, the objects 1 and 2 are possible objects to be merged in that image region. Knowing that information, a new VO_ID is generated to represent this group of persons. In Table 4.10 the VO_ID represents this group and in Figure 4.15 it appears marked with a pink bounding box.

Table 4.10 – Object merges example (*IdentifiedObjects* – b)

VO ID	Centroid(x,y)	Min(x,y)	Max(x,y)	Number of pixels	Presence variable	Group Identification	Lifetime	Possible Group (estimation)	Histogram
4	(64,146)	(28,97)	(98,200)	3437	0	1	90	0	…
3	(329,51)	(321,26)	(336,93)	529	0	0	37	0	…

It can be observed that the group identification field has now a new value (1) indicating a reference to an index of the *GroupsList* table represented in Table 4.11. This index represents the union of the object 1 and 2 in object 4.

Table 4.11 – Object merges example (*GroupsList*)

Group ID	Objects IDs contained
4	1 2

The object models of the objects contained in the group should also be stored, as referred before. Therefore, the lines in Table 4.9, representing the objects 1 and 2 in the previous frame, are stored in the *GroupsCharac* table for future estimations. The motion vector of the object in the last two frames is also stored in the *GroupsCharac* table. The structure of this table is illustrated in Table 4.12.

Table 4.12 – Object merges example (*GroupsCharac*)

VO_ID	Centroid(x,y)	Min(x,y)	Max(x,y)	Number of pixels	Presence variable	Group Identification	Vector(x,y)	Histogram
1	(45,148)	(30,20)	(59,101)	1480	0	0	(-1.2,1.6)	…
2	(84,141)	(69,28)	(102,96)	1820	0	0	(-3,2.3)	…

All these operations to perform an object merge are done in the Object merging module represented in the Figure 4.10, recording all the data generated by the merges on the referred tables. Figure 4.10, also shows that this module has access to all the necessary tables in memory (.

4.7.2 Object Splitting

Video objects can also be split between consecutive frames. This is another video event that represents the disaggregation of two or more objects. To have a split of a video object (blob) this normally as been classified as a GKP previously. If the system has a GKP it has the information about the objects that are contained in the group and their models in the *GroupsCharac* table. The correspondence between the various objects resulting from an object splitting event can be done using various sources of information. A simple approach is to use the object colour information stored as a histogram in the *GroupsCharac* table. This approach can have some problems, notably when the colour histograms of the various persons included in the split group are similar, which prevents the correct identification of the various persons in the group. The method of comparison of histograms used is the mean absolute difference (MAD) between two histograms that is described by (eq. 4.2):

$$MAD(k,j) = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \left|\mathrm{p}_{\mathrm{rgb}}^{\mathrm{k}}(\mathrm{c}_{\mathrm{i}}) - \mathrm{p}_{\mathrm{rgb}}^{\mathrm{j}}(\mathrm{c}_{\mathrm{i}})\right|; \qquad (4.2)$$

where N is the colour histogram size (for an RGB colour space N = 3×256), $\mathrm{p}_{\mathrm{rgb}}^{\mathrm{k}}(\mathrm{c}_{\mathrm{i}})$ and $\mathrm{p}_{\mathrm{rgb}}^{\mathrm{j}}(\mathrm{c}_{\mathrm{i}})$ are, respectively, the value of the colour histogram index c_{i} for object (k) and for object (j). The system uses a histogram composed by the three RGB components that are compared in sequence.

To help the system in persons distinction on splitting operations some spatial estimation are compared to each object inside the group like is done for the centroids correspondence. The method used to distinct the persons on splitting compare estimations of the centroid of the objects inside the groups and compares them to the split objects centroids. The method of the estimations of the centroids will be explained in the Section 4.8.3. With these estimations, it is possible to have an idea of where the people can be inside the group, so in case of a split the colour information can be used additionally to the estimated object position to achieve a better match of the identifiers after a split.

To detect this kind of events all the groups present in the *IdentifiedObjects* table should be verified. If some group disappears from the system it is verified if some new objects appear in the same image region, because this disappearance could be a group splitting. The new objects detected in this image region will be the target of the identifications that were inside the group. To assign these identifications to the new objects, the objects colour verification is done and the position estimation is compared with the new object positions (eq. 4.3), the colour difference (CD) has a weight of 65% to the identifier choice and the distance difference (DD) of the position estimations has the others 35% of weight.

$$Object\ \mathrm{ID}\ Correspondance = 100 * [(CD * 0.65) + (DD * 0.35)] \tag{4.3}$$

These weights were obtained empirically after various tests with this typical situation. The colour histograms verifications have higher weight because they are based on real object data instead of estimated data as is the case for the object position.

It may happen that an object group is split into a number of objects higher than its actual number of individual objects (this situation occurs, typically, due to object detection inaccuracies). For example, a group of two identified objects that splits into three different objects. In this case, the previous identifications are assigned to the closer objects and the object that does not have an identification to be assigned is considered as a new object in scene.

Another possible object splitting situation occurs when the group does not disappear but some new objects appear close to it within a maximum distance threshold (maximum_split_threshold), which are listed in the current *PossibleMerges* table. In this case, other verification needs to be done, notably, if the object classified as a group has lost more than a minimum established threshold number of pixels (pixels_loss_splitting_threshold) to consider that a split has occurred. In these cases, the new objects are seen as objects that go out of the group, so their colour and position verifications are performed to make the correct correspondence between the old and the new objects. When a group with a number of objects split to less objects than it had, it should be maintained the group identification in the object that were estimated by its size to be the possible group.

A common situation of object splitting in surveillance video sequences occurs when two or more persons enter the scene together as the same video object and after some frames they are separated. These situations are treated as particular cases because they have not be seen merging, so the system does not know that object is a GKP. This case occurs when two or more objects appear close in the place where another object has disappeared in the same frame, so it is verified if the mean centroid of these objects is not farther than an established threshold (maximum_single_split_threshold) from the last centroid of our possible group. When these objects are split, they are automatically classified as possible merges with the same method used to the possible merges classification explained in Section 4.7.1.

Figure 4.16 presents one example of these cases where the video object marked as pink has two persons that have not been identified individually when they entered the scene and after some frames the video object is split in two objects (marked as yellow and brown)

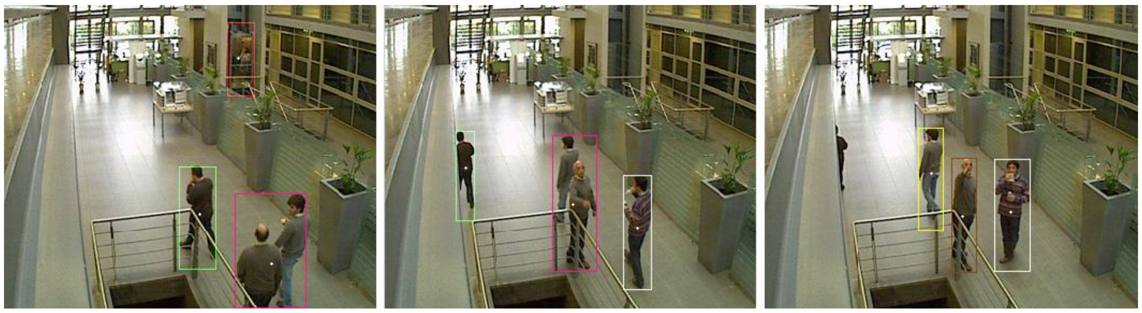


Figure 4.16 – One simple object splitting

These particular cases of single object splitting are processed before all the others events processes because the new split objects could be wrongly classified as objects exiting from other closer groups, so a little module was developed to deal with these cases. This module is represented in Figure 4.10 as the Simple object splitting module. All the other cases of split objects are treated in the Group object splitting module.

4.8 Intelligent Layer

This section will describe the heuristics applied to the information extracted directly from the video frames and the events detected between objects (splitting and merging operations). These heuristics aim to reduce the tracking errors trying to handle the uncertainties about the tracked objects (e.g., in the GKP and in PGUP), notably, how many objects are in a group and what is the exact position of each object. Other heuristics handle information about single object such as its occupied areas and bounding boxes sizes.

On this various modules will use a table with the standard sizes of the persons for different locations on image space (*SizesTable* on Figure 4.10). The size of the persons in the table is computed during a training phase before starting the tracking process. This phase consist in extract samples of the people sizes in different places from a video sequence of the monitored space and interpolate the sizes in the intermediate positions, storing all the measurements of height and width of objects in the *SizesTable* (see Figure 4.10). All the measurements have with reference the head of the persons (the higher y value of blob on image). It is worthwhile mentioning that this table is different to different camera locations.

4.8.1 Groups Uncertainty

This module deals with the cases of uncertainties associated to groups, notably, how to infer if a new video object corresponds to a single or multiple persons and how to infer the number of persons in a given video object located near the limits of the scene. Therefore, in these cases of uncertainty, the proposed solution is to estimate the number of persons through some additional heuristics, as described below.

These problems are handled by the *Groups Uncertainty* module (see Figure 4.10). After the events detection, described in Section 4.7, a "high-level" processing approach is proposed where some verifications and estimations are performed in order to improve the tracking accuracy in the presence of possible groups of persons. This "high-level" or intelligent processing approach relies on the following methods:

- group estimation;

- image border group verification.

The **group estimation** method is used to evaluate if a given detected object can be assumed as being a group of persons. Based on *a priori* knowledge about the monitored space, it is possible to compute a typical person size for various locations in the image and, based on this information, infer if a given object corresponds to a single person or not. The information about the size of the persons is stored on the *SizesTable* and with it is possible to compare the size of a detected object with the estimated typical size of a person in the same place of the scene image. If the detected object exceeds this estimated typical size, the system assumes that this object corresponds to a group of persons that have entered together in the scene. Consequently, the object is inserted in the *IdentifiedObjects* list where a field is reserved to mark the object as a possible group.

The **image border group verification** method is used when a group of persons moves near the image border to handle situations where some persons in the group may go out of the image or some persons enter the group from the outside of the image. This happens because the tracking process relies to the blobs detected by the *Video object detection* module and when these blobs are located near the border of the image, events like persons joining/leaving a group from/to the outside of the image are virtually impossible to detect. The alleviate these problems, some verifications can be done to catch some of these events. This module looks at the groups of persons that are close to the image border (i.e., the bounding box is closer to the image border by less than a given threshold – border_vicinity_threshold) and if the width or height of some of these groups decreases by more than an established threshold (blob_size_decreasing_threshold) the system assumes that some persons went out of the group and need to be deleted from the group. To select which is/are the person(s) to delete from the group, the estimated position of the persons inside the group is used. The person selected to be deleted from group will be the person which centroid position is estimated closer to the boundaries of image. In the proposed system, the blob_size_decreasing_threshold value proposed is equal to 40% of the blob size and was obtained empirically after various tests with this typical situation. The position estimation used to the selection of the object to exit the group is explained in Section 4.8.3.

4.8.2 Object Area Correspondence

The *Object areas correspondence* module is used to verify if there is a matching between the areas of two candidate objects at different instants. This method is used when one object disappears from the *IdentifiedObjects* table and some other object appears near to the image position of the disappeared object. In this case, the system verifies if the area of the first object matches the area occupied by the new object. If their bounding boxes match by more than the area_correspondance_threshold, the system considers that the new object is the same object that has disappeared. The area_correspondance_threshold proposed is 50% of the object area and was obtained empirically after various tests with this typical situation. This verification is useful because in some cases the centroid of an object can exhibit large variations between consecutive frames resulting in change on the identification (VO_ID) of the processed object. However, the object area does not change abruptly and, consequently, this method can save some of the objects identification losses during the tracking process.

4.8.3 Group Object Position Estimation

During object tracking is usually difficult to obtain an accurate object position at all processed frames. This problem is usually due to occlusions that generate inaccurate object detections. When these situations occur, it is necessary to estimate the unknown object position aiming at having a continuous and fluid trajectory. There are essentially two kinds of occlusions in a video sequence:

- an occlusion between two or more moving objects which is a merge;
- an occlusion caused by a scene obstacle.

Considering the case of the occlusion between moving objects, the estimation of the objects positions while they are occluded is especially useful to help on their identification assignment when they split out. The object position on the occlusions by scene obstacles are estimated too, but it in the merges is necessary to detect the entry and exit events of the occlusion region this events detection in done in other way as will be seen in the Section 4.8.4

For the first case of occlusion the group object position estimation is initialized when two or more objects are merged and is differently handled depending on how much previous information the system has about the considered object before the occlusion (object merging):

1. **The system has two previous samples of the object position** – This situation allows calculating the object motion vector before entering the group, this vector will be used to estimate the object movements inside the group.
2. **The system has only one sample of the object position** – This situation, which does not allow computing the object motion vector, occurs when the object enters in the scene and in next frame is merged into a group or when an object jumps from one group to another.

Consequently, depending on the conditions before the merge, the estimations will be handled differently as explained in the following two sections.

4.8.3.1Object Position Estimation Based on its Motion Vector

When a video object gets occluded by another object the system confirms if the object existed already in the two previous frames. In the affirmative case the system computes the object centroid motion vector on the image plane by subtracting the two last object centroid positions. This motion vector is then used to estimate the object position while the object is occluded.

While occluded, the object position is estimated by adding the object motion vector to the last estimated or known object position. This proposed object position estimation method works well if the tracked objects have a linear motion in the image plane, which is not typically the case. When the movements of an object are vertically on the image the movement vector of the object will not be linear because the object moves to a farther/near location. This correspond to different depths on the image scene, the movements in bigger depths on image will be smaller in the image plane and vice versa. A simple approach to overcome this problem is to compensate the non-linear motion by a multiplicative factor that reduces the motion vector amplitude for bigger image depths and increases the motion vector amplitude when the object becomes closer to the camera. The multiplicative factor used has a value of 3%, so for each pixel in direction to a higher depth (in this case a higher value of y) the norm of motion vector is reduced by 3%. This value was obtained empirically after various tests with these typical situations. Since this method only provides a rough estimation of the occluded object position, the system checks if the estimated position is out of the group area and resets its position to the limit of the group boundary in the affirmative case. An example of this kind of estimation can be seen in Figure 4.17, on the object with the blue label that when merged with the object with pink label it starts to be estimated (with white label).

4.8.3.2 Object Position Estimation Based on its Position on the Group

When the system does not have sufficient previous samples of the object positions to compute its motion vector before their merge (occlusion), the object position estimation becomes more difficult. To handle this problem, the proposed approach is to estimate the object position based on its relative position in the group. This means that, if the object enters in one group on an upper-middle position, it should be on this relative position to the group during the occlusion, even if the group moves to another place. Like the estimation based on the motion vector, if the estimated object position exceeds the group area limits it will be placed in a possible correct place inside the group area. An example of this kind of estimation can be seen in Figure 4.17. When the objects with the pink and blue labels are merged the pink object is always estimated on the bottom-right region of the group area.

In both types of group object position estimation, the size of the object must be corrected according to the estimated position. For this purpose is used the *SizesTable* which contain the standard object sizes for the various possible positions in the image allowing at each estimated position to correct the size of the objects.

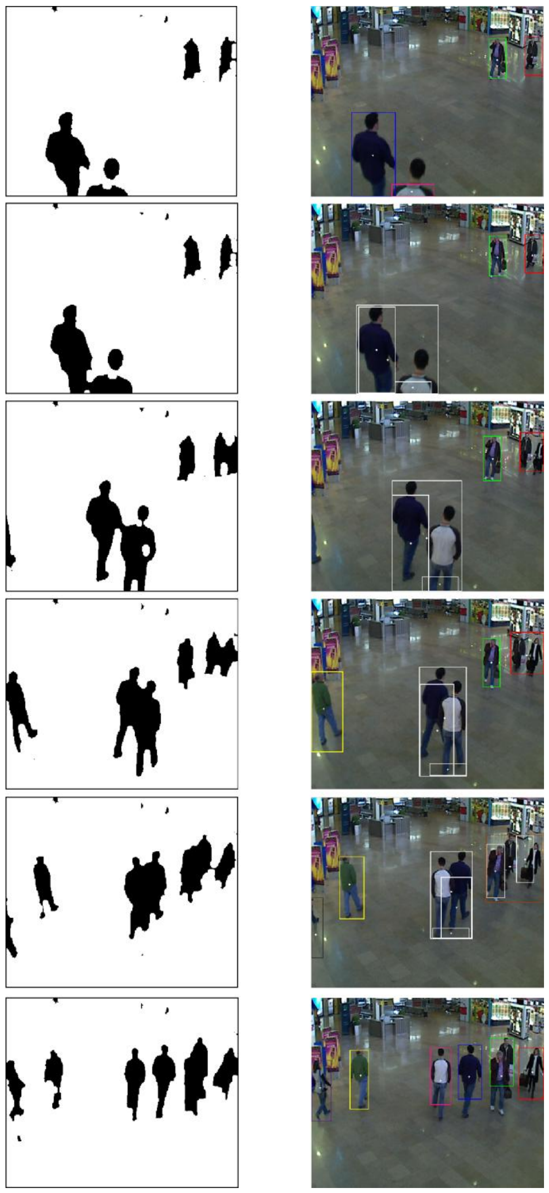


Figure 4.17 – Group object position estimations

4.8.4 Occlusion Handling

This module is useful to handle object identifiers when the object to be tracked becomes occluded by scene obstacles as illustrated in Figure 4.8. In these cases the system has to detect when the object enters in the occluded region and afterwards when the object leaves this region appearing in one of the borders of the obstacle. In the proposed approach, information about occluded regions has to be provided offline, indicating the obstacle surroundings and the area that occupied in scene, as can be seen in Figure 4.18. At left are marked the surroundings of the obstacle where the events of the occluded objects can be detected. The events to detect can be an object appearing or disappear. In the same figure at right are marked the region where the occluded object can stay occluded which is the area where their position can be estimated.

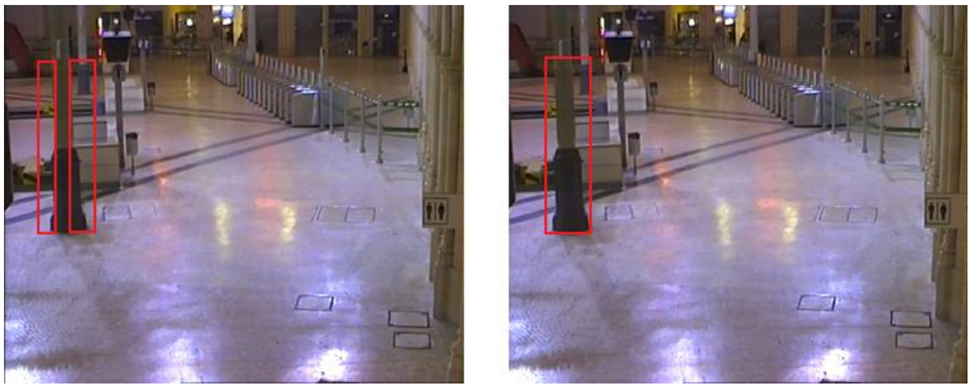


Figure 4.18 – Static-occlusion regions due to a scene obstacle

The system starts to capture when an objects goes to the backs of the obstacle (object disappearing), it is detected if one object disappear in one of the surroundings areas of the obstacle. When this events is captured it can be used a position estimation of the object like in groups of people. Like in group object position estimation the object information has to be saved to do comparisons when the object exits the occluded area. This information is saved in *OccludedObjects* table as can be seen in Figure 4.10. In this table are saved the characteristics of the occluded objects and the needed values for the estimation as in the group estimation is used the *GroupsCharac* table.

While the object is being estimated it can be detected an event of an object appearing on one of the surroundings areas of the obstacle, which means that one object goes out from the obstacles backs. When the occluded object appears their histogram and estimation position is compared with all the objects that are behind the obstacle to maintain the objects identification after the object has been occluded.

These operations done by the *Occlusion handling* module although they are a kind of intelligent event detection they have to be executed before the others events detections (merge and split events) because if this not happens some objects that have been occluded could be added to a near group when they really go to the backs of an obstacle.

4.8.5 Object Size Correction

Besides the detection of events between objects and the described verifications above, the bounding boxes information about the objects can be improved. The video object detection sometimes are incomplete or have shadows or reflections that distort the objects size and the system return wrong bounding boxes. To improve these situations, this module should receive the bounding boxes information about the objects (Min(x,y) and Max(x,y) on *IdentifiedObjects* – Table 4.5) giving as output the corrected bounding boxes. So, these improvements can be done with the standard sizes of the *SizesTable*, which can correct situations of people that are bigger than the standard because of shadows or reflections on the scene or even correct some cases of incomplete detections or non-connected objects. To correct the bounding box is used as reference in y coordinate the top of the blob, usually person heads, and the x coordinate of the centroid, adjusting the object height and width. Using these standards it could introduce some errors in the tracking; in the case of children, they are small in relation to standards, so it can think that is a case of an incomplete detection increasing his size. The opposite case can happen too, if two people enter in scene together they will be seen as an object bigger than the standards, so it can consider that the excess of size in relation to the standard can be a shadow or reflection of the person on the ground.

In Figure 4.19 can be seen a case of a size correction where the shadow of the object is corrected on both sides can be seen in with a red bounding box the contours of the object blob which were corrected to the green bounding box, using the described method on this Section for the object size correction.

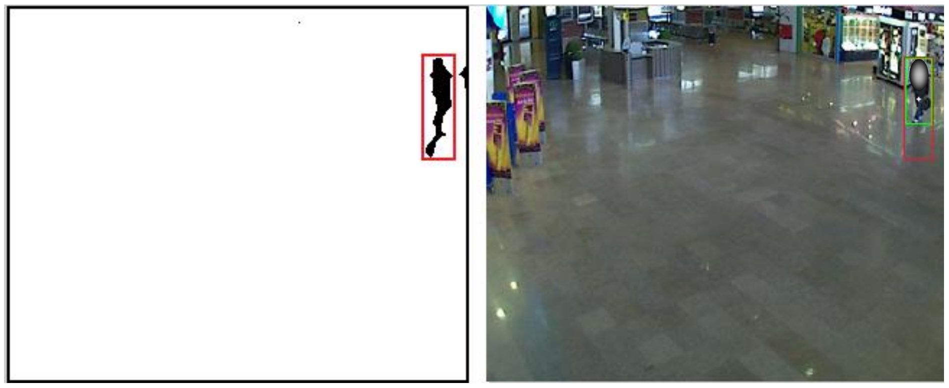


Figure 4.19 – Object size correction

4.9 Communication Layer

After the entire tracking process the information extracted from the video images must be sent to the other TRAPLE modules for further processing and handling. Therefore, the tracking system has to undertake two prior data processing steps in order that the generated data can be used by the other TRAPLE modules. These steps consist of a spatial transformation and MPEG-7 standard messages generation, as can be seen in Figure 4.10.

4.9.1 Spatial Transformation

The first step does the standardization of the monitored space, changing from the image coordinates of the video objects into the common coordinate space of the monitored scene map, as illustrated in Figure 4.20.

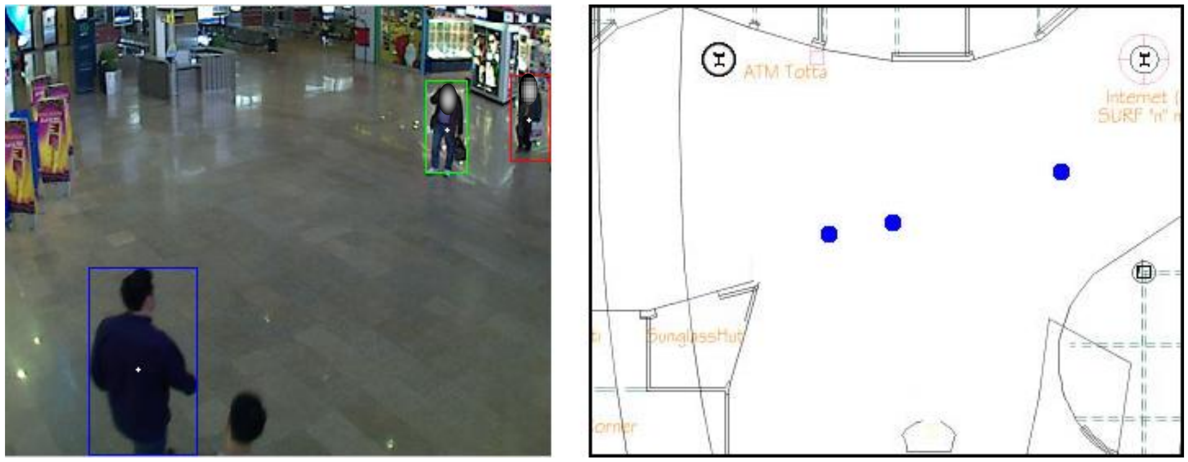


Figure 4.20 – Spatial transformation (Image to scene map coordinate system translation)

Using the scene map coordinate system, the problem of positioning objects from different cameras is solved. This way, an arbitrary number of cameras covering the monitored scene can be used, each one with a different spatial transformation. This module is implemented using the Camera Calibration Toolbox for Matlab [23], which performs a change of location from the image plane to the ground plane. This module receives as input the object positions in the 2D image plane (image coordinate system) and outputs the corresponding positions in the 2D ground plane (scene map coordinate system).

4.9.2 MPEG-7 Messages Generation

Having a standard common coordinate system and the positions of the objects with their correct identification, the tracking sub-system has all the information to exchange with the other TRAPLE modules or eventually with other external modules. The problem is how to exchange these messages making sure that they are understood by the other module. To achieve this goal it was decided to use standard communication messages for all the modules which are defined in the MPEG-7 standard [2]. This module constructs the MPEG-7 messages according to the standard encapsulating the information generated by the tracking algorithms. These messages will contain object positions, new objects declarations and split and merge events. All the information generated has temporal and spatial references (on image coordinates and on map coordinates – BB and locations) and relation with the objects that generate the information.

4.10 Final Remarks

This chapter has described the Intelligent Video Object Tracking Solution Proposal for TRAPLE and its conceptual approach, presenting the objectives of the solution and the issues that the system has to handle in detection and tracking modules. The proposed tracking solution architecture is divided in four main layers: 1) Image Information Extraction Layer, 2) Events Detection Layer, 3) Intelligent Layer, and 4) Communication Layer. The first one deals with the needed information about the objects in scene (object model) and the objects identifications; the second layer detects the events between them; the third layer handles some uncertainty information maintaining the tracking more coherent along the time; and finally the fourth layer implements the interface with the exterior modules.

Chapter 5

Performance Evaluation

5.1 Introduction

The new video object tracking approach proposed in this Dissertation has been evaluated in some different environments to validate the concept of tracking video objects through their centroids and estimating the video object positions when not enough information about the objects position is available. Three different types of video sequences are used in this evaluation representing various types of environment, from the simpler (e.g., a small office with few persons entering and leaving the scene) to the harder situations to process (e.g., an airport and a train station).

5.2 Computational Limitations

As mentioned in the introduction of this Dissertation, one of the critical problems in real-time video processing is the amount of time required for processing each frame. In the case of this solution this limitation derives mainly from the video object detection module, which spends most of the computational resources used for video processing as can be seen later in this chapter.

The major factors that directly affect the processing time of video images are the spatial and temporal resolutions of the video frames, which define the amount of data to process per time unit. Therefore, after some testing, it was decided to use a maximum spatial resolution of 352×288 pixels (CIF) for all test sequences. This spatial resolution was chosen as a trade-off between processing time and object detection accuracy since higher spatial resolutions would require too much time to process (inhibiting the real-time video processing) while lower spatial resolutions would lead sometimes to less perceptible images endangering accurate object detection. Regarding the temporal resolution, using a fixed spatial resolution, various video processing temporal resolutions have been investigated ranging from 16 fps to 3 fps looking for the better configuration.

5.3 Test Sequences

The following test video sequences have been used:

- *Hall Monitor* – A video sequence of a small office hall environment with only a maximum of 2 persons walking simultaneously in the scene, corresponding to the less complex scenario. The sequence has a duration of 11 seconds at 30 fps and CIF spatial resolution
- *Airport* – A video sequence captured in Faro airport in south Portugal, where the TRAPLE system has been tested. This sequence exhibits 10 persons walking in the scene with a maximum of 7 persons simultaneously under the field of view of the camera. The sequence has a duration of 82 seconds at 16 fps and CIF spatial resolution.
- *Train Station* – A video sequence captured from a surveillance camera in the Rossio train station in Lisbon where various persons walk to a waiting platform from the ticket gates. This sequence exhibits 15 persons walking in the scene with a maximum of 6 persons simultaneously under the field of view of the camera. This sequence covers a large environment, like in the Airport sequence, but in this case the camera is positioned far from the monitored space and contains some obstacles, which difficult the tracking of the people. The sequence has a duration of 75 seconds at 3 fps and CIF spatial resolution.

Figure 5.1 illustrates the three test environments used in the performance evaluation of the proposed video object tracing system with samples of the corresponding test sequences.

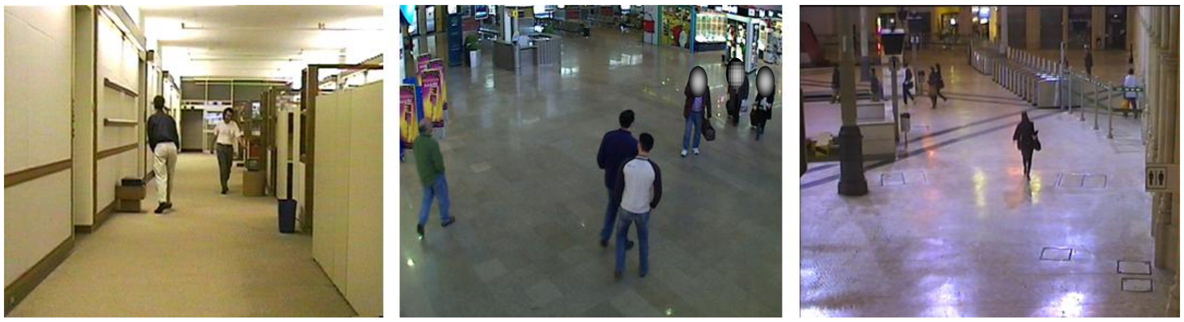


Figure 5.1 – Test sequences: *Hall Monitor* (left), *Airport* (center), *Train Station* (right)

In the *Hall Monitor* test sequence, the environment monitored is very controlled without external illumination and without obstacles. The camera provides, essentially, a frontal view of the monitored scene. This situation does not help when the people are mixed in scene. This example is very simple, only with 2 persons walking in the scene and it is not a problem for tracking them. The objects are walking very close to the camera, which facilitates the tracking because the foreground areas detected (blob) are, usually, large.

In the case of the *Airport* test sequence, the environment is more complex: the covered area is bigger and has some natural lights in the bottom of the scene, which can be a source of noise to the detection of objects. This sequence has some persons walking mixed at the same time, which will test the merge and split algorithm developed. The camera is placed in a higher place than in the *Hall Monitor* sequence and a little tilted down, which is better for the tracking of many persons because this causes less occlusions between the objects when they are close to the camera. Because the covered area is large in this test sequence, objects that are located far from the camera will be difficult to detect being mixed with the noise of the natural lights. This sequence has also some regions where some shadows can be seen, e.g., near to the shop at top right in image.

The *Train Station* sequence is similar in complexity to the *Airport* sequence but all the objects are moving far from the camera, which makes object detection more difficult in this case. The scene has one obstacle that permits testing the algorithm developed to estimate the occluded objects behind obstacles (Occlusion Handling on Figure 4.10). Another important aspect of this sequence is the scene illumination causing large shadows around the moving persons. It is important to notice that the camera is located at a higher place than in the other two test sequences and tilted down. This camera location is more adequate to avoid occlusion between objects when the objects are closer to the camera. In cases where the objects are at distant location, the objects become smaller in the image plane and, consequently, become more difficult to detect.

Notice that the angle and the distance of the camera to a given point in the scene will determine the actual size of a person located at that point in the image plane. Consequently, process different tracking cameras it is necessary to calibrate the system to know the size model of the persons on the different positions of image as described in Section 4.8.

5.4 System Configuration Parameters

Table 5.1 shows the configuration parameters of the proposed tracking system for the set of CIF sequences used during the performance evaluation tests. Although this set of parameters works fine for the set of test sequences used, for other sequences this set of parameters would need some adjustments for a better tracking. In terms of temporal resolution, all these parameters were tuned for a frame processing rate of 3 fps; different frame processing rates would also require some further parameter adjustments. Table 5.1 presents in the two right columns the type of dependency (spatial or temporal) for each parameter.

Table 5.1 – System configuration parameters

Parameter	Description	Value (pixels)	Spatial dependency	Temporal dependency
minimum_object_size_threshold	Minimum number of pixels that an object can have to be considered an object to be tracked	600	✓	
centroids_distance_threshold	Maximum distance between centroids of the same object in two consecutive frames	70	✓	✓
minimum_merge_pixel_threshold	Minimum difference of pixels to consider a merge	500	✓	
maximum_merge_threshold	Maximum distance between centroids to consider a merge	100	✓	✓
possible_merge_threshold	Maximum distance between bounding boxes of two objects to consider a possible merge	40	✓	✓
maximum_split_threshold	Maximum distance between centroids to consider a split	80	✓	✓
pixels_loss_splitting_threshold	Difference of lost pixels of an object group to consider a single object split	500	✓	
maximum_single_split_threshold	Maximum distance between bounding boxes of a group and a single object to consider that the object has been split from the group	24	✓	✓
border_vicinity_threshold	Minimum distance to the image border to consider an object in the image limits	40	✓	✓
blob_size_decreasing_threshold	Maximum group blob size decreasing percentage to consider that some objects have leaved the group over the image border	40%		
area_correspondance_threshold	Minimum object area percentage that a lost and a new object can have in common to be considered the same object	50%		

The calibration of these parameters, such as, for example, the standard sizes of the persons, is very important for the proper functioning of the system. These parameters are established manually but in future works this should be done automatically before the system track someone.

5.5 Tracking Results

The proposed system has been evaluated with respect to important object tracking objectives using the test sequences described above and two major sets of tests:

- Temporal tests – This set of tests evaluate the ability of the proposed tracking system to work in real-time for a given amount of computation power available.
- Accuracy tests – This set of tests evaluate the reliability characteristics of the proposed tracking system as, for example, the detection error and the object identifications consistency.

The following sections present and discuss these results.

5.5.1 Temporal Results

The platform used to carry the evaluation tests was based on a Dual-Core Intel Centrino processor at 2.53 GHz with 4 GB of RAM running the Windows Vista Operating System and the system was implemented in MATLAB.

The tracking process can be done using a visual interface (w/ GUI) to view the video being processed and without any visual interface (w/o GUI) where no time is spent in visual presentation, which frees more computational power for the tracking process rendering it faster. The GUI is a simple window that shows two images (see Figure 4.19):

- The output binary image of the detection module of the current frame;
- The current frame with the bounding boxes of the tracked objects drawn on top.

Others results compared are the mean time spent in processing each frame (mean time per frame), the percentage of time spent during the video object detection process and reading from a video file. Reading a video from a live stream should be faster too.

Table 5.2, Table 5.3 and Table 5.4 summarize the temporal evaluation results of the proposed tracking system for the three chosen video sequences at different frame rates. The detailed set of results is given in Annex A. The following information is presented in these tables:

- Target frame rate – The frame rate at which the test sequence has been processed; lower frame rates are obtained through temporal sub-sampling discarding k out of n frames for a processing frame rate of k/n times the base frame rate (i.e., the acquisition frame rate).
- Processing time with/without GUI – The time spent by the proposed tracking system in processing the given test sequence with/without a graphical user interface (GUI) for showing the tracking results. The percentage of the total actual video processing time spent relatively to the real-time video duration with/without the GUI is shown in parenthesis.
- Mean time per frame with/without GUI – The mean time spent by the proposed tracking system in processing each video frame with/without GUI.
- Detection time percentage – The percentage of processing time spent by the detection module relatively to the entire video processing time (i.e., read video + detection + tracking).
- Read video time percentage – The percentage of time spent in reading the video files from disk relatively to the entire video processing time (i.e., read video + detection + tracking).

Table 5.2 shows that the minimum frame rate at which the system can achieve real-time processing for the *Hall Monitor* test video sequence is 3 fps whether the GUI is used or not. In this case, the tracking system spends 91% of the available processing time between frames when the GUI is used and 78% without the GUI.

These results also show that, when the frame rate of the processed video is reduced, the mean time per frame increases. This happens because, when the processed video frames are more distant in time, the tracking module will have more differences between frames to process, which requires more computational power. This also reflects the trade-off that has to be achieved when selecting the system video frame rate: higher frame rates require more processing time because more images have to processed per time unit; however, decreasing the video frame rate will render more information to process per video frame which tend to increase the computational power required.

It is worth mentioning that the tracking process only spends around 20% of all the video processing time. Most of the time spent in video processing is spent by the video object detection module (between 72% and 81%). In the tested setup, the video frames are read from a file on disk, which requires also non-negligible amount of time (between 4% and 5%).

Table 5.2 – Tracking results: *Hall Monitor* (11 s)

Target frame rate (fps)	Proc. time w/ GUI(sec)	Proc. time w/o GUI (sec)	Mean time per frame w/ GUI (sec)	Mean time per frame w/o GUI (sec)	Detection time percentage	Read video time percentage
10	28.86 (262%)	24.34 (221%)	0.26	0.22	81.2%	4.1%
6	18.47 (168%)	15.24 (139%)	0.27	0.23	78.7%	4.3%
3	9.96 (91%)	8.55 (78%)	0.30	0.25	71.5%	5.2%

Regarding the more complex *Airport* test video sequence (see Table 5.3) similar conclusions can be taken. The frame rates tested were not exactly the same because the original test video sequence was captured at a base frame rate of 16 fps. In this case, at 3.2 fps it can only be possible to process the sequence in real-time without the GUI.

Although this sequence has more persons walking in the scene and they become sometimes mixed, this does not influence significantly the mean time spent to process each frame, which is typically around 200 ms per frame (see Figure 5.3).

Comparing the time spent by the video object detection module, it can be concluded that, as more frames are processed the bigger is the percentage of time spent by this module.

Table 5.3 – Tracking results: *Airport* (82 s)

Target frame rate (fps)	Proc. time w/ GUI(sec)	Proc. time w/o GUI (sec)	Mean time per frame w/ GUI (sec)	Mean time per frame w/o GUI (sec)	Detection time percentage	Read video time percentage
16	328.95 (401%)	275.94 (337%)	0.24	0.20	86.4%	2.7%
8	158.63 (193%)	132.40 (161%)	0.24	0.20	88.9%	3.3%
4	112.03 (137%)	89.07 (109%)	0.25	0.20	87.2%	5.3%
3.2	83.97 (102%)	67.67 (83%)	0.25	0.20	72.8%	5.4%

Table 5.4 shows the results obtained for the *Train Station* test sequence. Only one frame rate is shown because the original video was captured at 3 fps. As can be seen in this table, all processing times are smaller when compared to the other sequences, this is because the video file is read in its original frame rate. The percentage of time of the video object detection is higher in this case relatively to the other test sequences because the video reading time is smaller for this test video sequence relatively to the other two. The frame rate at which this video is processed can, eventually, be increased from 3 fps to 4fps without GUI because, as can be seen in Table 5.4, the system has a margin higher than 33.3% of the processing time free.

Table 5.4 – Tracking results*: Train Station* (75 s)

Target frame rate (fps)	Proc. time w/ GUI(sec)	Proc. time w/o GUI (sec)	Mean time per frame w/ GUI (sec)	Mean time per frame w/o GUI (sec)	Detection time percentage	Read video time percentage
3	55.62 (74%)	44.97 (60%)	0.23	0.19	89.2%	2.8%

Considering that a real tracking system like the one proposed in this dissertation is designed to work in a live environment reading the video frames from a live stream, it is reasonable to assume that the actual processing times should be lower since the video input will be faster than reading from a file, as in the tests described in this section.

Figure 5.2, Figure 5.3 and Figure 5.4 show the processing times for each frame of the selected test video sequences, respectively, *Hall Monitor*, *Airport, and Train Station.* As can be seen in these figures, where the vertical axis represents the processing time per frame and the horizontal axis the frame number, the processing time per frame is always below 0.3s, revealing the stability of the processing time spent at each video frame for all the sequences tested and, consequently, the ability to perform video object tracking in real-time.

Note that the time spent processing each frame is fairly stable in all the video sequences and are always around 0.2 seconds per frame. But looking for the time spent in the detection module and reading the video file is clear that the tracking system is executed very fast. For example, in the *Airport* sequence at a frame rate of 3.2fps the percentage of the total processing time devoted to tracking is around 21.8% (*100-(72.8+54)* – see Table 5.3), i.e., 43.6 ms per frame.

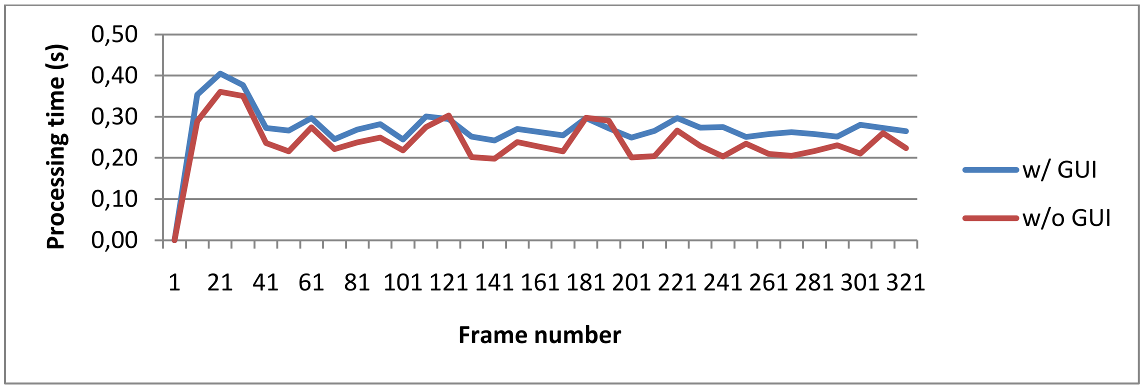


Figure 5.2 – Frame processing time: *Hall Monitor* – 3fps

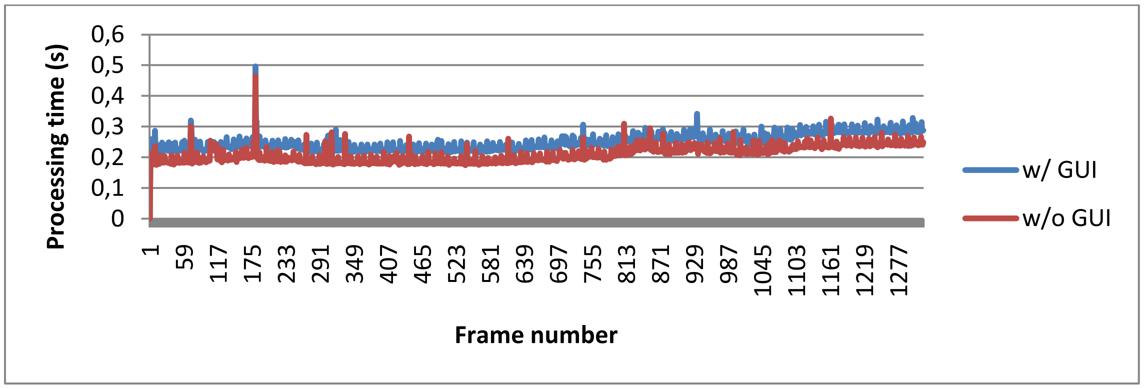


Figure 5.3 – Frame processing time: *Airport* – 3.2fps

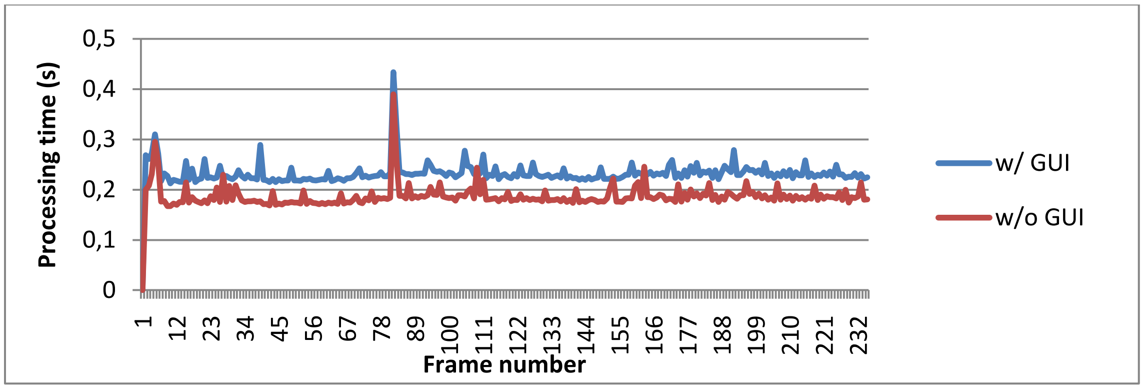


Figure 5.4 – Frame processing time: *Train Station* – 3fps

5.5.2 Accuracy Results

This second set of results intends to evaluate the performance of the proposed tracking system in maintaining the video object identification coherent along time.

Since the performance of the video object tracking module is very much dependent on the results of the video object detection module, this section starts by analyzing the accuracy of the video detection module adopted by the TRAPLE system. The performance evaluation approach adopted for this case relies on measuring the difference between the actual number of objects in each video frame and the number of objects tracked by the proposed tracking system. These comparisons measure the amount of errors transmitted to the tracking system by the video object detection module. For this purpose the following metrics are used:

- **Mean detection error (MDE)** – Evaluates the difference between the real number of objects (RO) in each video frame and the number of tracked objects (TO) averaged over the entire video sequence, as seen on eq. 5.1, where *N* is the number of frames in the video sequence;

$$MDE = \frac{1}{N}\sum_{i=1}^{N} |\mathrm{RO}_\mathrm{i} - \mathrm{TO}_\mathrm{i}| \tag{5.1}$$

- **Mean successful detection (MSD)** – Evaluates the average percentage of successful object detections over all the objects present in scene computed by eq. 5.2 where $\mathrm{NA_i}$ and $\mathrm{ND_i}$ are, respectively, the number of appearances and the number of detections of object i in the entire video sequence, and N is the number of objects in the video sequence.

$$MSD = \frac{1}{N}\sum_{i=1}^{N} \frac{NA_i}{ND_i} \times 100 \tag{5.2}$$

- **Mean identification changes (MIC)** – Evaluates the number of times that each object changes of identification (VO_ID) during his permanency in the scene averaged over the number of objects in the scene, as seen in eq. 5.3, where $\mathrm{NC_i}$ is the number of changes of identification of object i during its presence in the scene and N is the number of objects in the video sequence.

$$MIC = \frac{1}{N}\sum_{i=1}^{N} \mathrm{NC_i} \tag{5.3}$$

For the easier sequence – *Hall Monitor* – the number of tracked objects has shown good results with 100% of success (*MDE*=0; *MSD*=100%, and *MIC*=0), i.e., the persons present in the scene are always detected while they are in the scene. This accuracy is illustrated in Figure A.14 and Figure A.15 of Annex A where the analysed video sequence is shown at every 15th frame. While Figure A.14 shows the thresholded filtered background subtraction results, Figure A.15 shows the tracked video objects with the corresponding coloured bounding boxes.

The accuracy results for the second sequence – *Airport* – are shown in Table 5.5. From this table it can be concluded that for the *Airport* test video sequence the number of tracked objects has some detection errors comparing to the *Hall monitor* sequence. It can be seen that this sequence has a *MDE* of about 0.3 persons per frame and a *MSD* of about 92%.

Table 5.5 – Accuracy results: *Airport*

Frame rate (fps)	Mean detection error (number of objects)	Mean successful detection (MSD)
16	0,30	92,09%
8	0,32	92,97%
4	0,33	91,97%
3.2	0,31	92,92%

The accuracy results for the last sequence – *Train Station* – are similar to those obtained for the *Airport* test video sequence with some better results as can be seen in Table 5.6, where the *MDE* decreases to 0.26 and the *MSD* increases to 99%. This situation happens because in this sequence the objects although are farther from the camera they moves in most of cases separately without occlusion between object.

Table 5.6 – Accuracy results: *Train Station*

Frame rate (fps)	Mean detection error (number of objects)	Mean successful detection (MSD)
3 fps	0,26	99,49%

Figure A.1 to Figure A.13 in Annex A illustrate the differences between the number of tracked objects and the real number of video objects in each video frame for the three tested video sequences for different processing frame rates.

The accuracy results presented above give information about the detection errors in the test video sequences, but one of the main objectives of the tracking system is to keep the identification of the objects coherent along the video sequence.

This evaluation can only be done visually, i.e., by comparing the identifications assigned to each person in the video sequence with the tracking results. The numerical results of these visual analyses are presented in Table A.1, Table A.2 and Table A.3 of Annex A, where he the number of VO_ID changes for each video object is presented.

A summary of these tables are presented in Table 5.7 and Table 5.8, where the mean identification changes are presented for the *Airport* and *Train Station* test video sequences for various frame rates.

In the *Hall Monitor* sequence only two persons are walking in the scene without any occlusion and, as was shown before, there are no detection errors in this sequence. Additionally, since there are no occlusions, the MIC is zero, showing that the proposed tracking solution presents good results for video scenes without occlusions.

For the *Airport* sequence, with a complex environment without obstacles, the results are not as perfect as for the *Hall Monitor* sequence. The average person walking on the scene changes his identification rarely as can be seen by the low value of the MIC metric, which is less than one change per person (see Table 5.7). It is important to note that the objects that enter in scene in group are first identified as a single object (PGUP) and when they split (the PGUP becomes two or more IPs) their identification changes, which is the main cause for the MIC increase.

The *Train Station* sequence represents a large environment where the persons are moving far from the camera and have obstacles in scene. In this type of environment the existence of obstacles and the higher distance between the camera and the moving objects in the scene makes the tracking more difficult increasing the *MIC* to 1.46 changes per person (see Table 5.8).

Table 5.7 – Identification accuracy: *Airport*

Frame rate(fps)	Mean identification changes (MIC)
16	0,5
8	0,7
4	0,7
3.2	0,3

Table 5.8 – Identification accuracy: *Train Station*

Frame rate(fps)	Mean identification changes (MIC)
3 fps	1,46

5.6 Final Remarks

This chapter presented a set of tests to evaluate the video object tracking system proposed on this Dissertation. The evaluation presented demonstrates the good viability of the proposed system both in terms of detection and identification tracking for various environments and spatio-temporal resolutions.

For the most difficult test sequences (*Airport* and *Train Station*) the results are slightly worst than for the *Hall Monitor* sequence because these sequences have much more people moving along the scene with a large number of occlusions between persons and, in the case of the *Train Station* sequence, also occlusions caused by obstacles in scene.

The proposed system, although not being currently implemented in an optimized platform, has demonstrated good ability to support real-time video processing at 3 fps in a Dual Core Platform at 2.4 GHz running Windows Vista Operating System. In these conditions the percentage of correctly tracked objects was always above 90% on average with a reduced number of identification changes.

Chapter 6

Conclusions

This Dissertation has proposed and evaluated a new approach for tracking video objects relying on a simple video object detection method in order to improve the processing time of the overall tracking system. In this context, a simple and fast centroid tracking method is proposed to track the centroids of the video objects exploiting the typical small movements between two processed frames at reasonable frame processing rates. Since this simple detection method originates some inaccuracies in the video objects detection, in cases of uncertainties about the objects positions and sizes, the proposed system tries to estimate the “missing” positions through some heuristics. This approach is especially useful in complex scenes because of occlusions between objects or caused by obstacles presents in the scene. These estimations can also handle detection inaccuracies caused by shadows or reflections in the scene, which are corrected with the estimations of the persons sizes for the corresponding location in scene.

Some tracking algorithms have been reviewed from the literature to contextualize the scope of this Dissertation. The algorithms reviewed demonstrated good solutions to track video objects, but always with complex mathematical operations or recursive searches to track the video objects. After reviewing the literature the proposed intelligent video object tracking solution is detailed, exposing the system architectures and the algorithms used to concretize this approach. It was shown that the tracking solution proposed works on four different layers to deal with different problems: 1) Image Information Extraction Layer, 2) Events Detection Layer, 3) Intelligent Layer, and 4) Communication Layer. The first one deals with the needed information about the objects in scene (object model) and the objects identifications; the second layer detects the events between them (split and merge); the third layer handles some uncertainty information maintaining the tracking more coherent along the time (estimations); and finally the fourth layer implements the interface with the exterior modules.

The tracking solution proposed was evaluated in three different realistic scenarios with different conditions and has demonstrated good accuracy in tracking video objects in real-time at 3 fps, as was shown in Chapter 5. The successful video object detection rates were always above 90% of success having low rates of identification changes of the video objects tracked along the scene during their presence. The time processing shown in the tests only for the tracking operations (without video streaming and video object detection module) was about 45 ms per frame, which, cooperating with faster video object detection, could provide higher video processing frame rates. With higher frame rates the tracking process will also have less tracking errors because it will have more samples to analyse.

The proposed video object tracking system was developed within the context of a partnership between ISCTE-IUL and THALES, which has originated the TRAPLE project, whose architecture and algorithms approaches have been patented in the European Patent Office [3].

Some modules of the proposed tracking system could still be further improved in future works:

- **Group estimation** – Improve the blob processing trying to understand how many persons belong to it;
- **Automated calibration** – Develop a system calibration tool to facilitate the implementation of the system in various environments with reduced human intervention during setup.

The video object detection and tracking modules were implemented in MATLAB, which is a good platform for testing since it facilitates implementation but it has the drawback of not being the most efficient platform to process video images in terms of computational power required. Another possible direction for continuing this work would be the migration to another more efficient platform such as OpenCV [24] using a C++ environment. The additional efficiency gain could be exploited by allowing a higher frame rate on the video processing or even allowing processing various cameras in the same machine.

Annex A

Detailed Results

Figure A.1 to Figure A.13 present the comparison of the real number of objects in the tested sequences in Chapter 5 and the number of tracked objects.

Table A.1 to Table A.3 present the Individual identification changes and sucessful detections in the tested sequences in Chapter 5.

Figure A.14 and Figure A.15 present a detailed tracking of *Hall Monitor* sequence at each 15 frames.

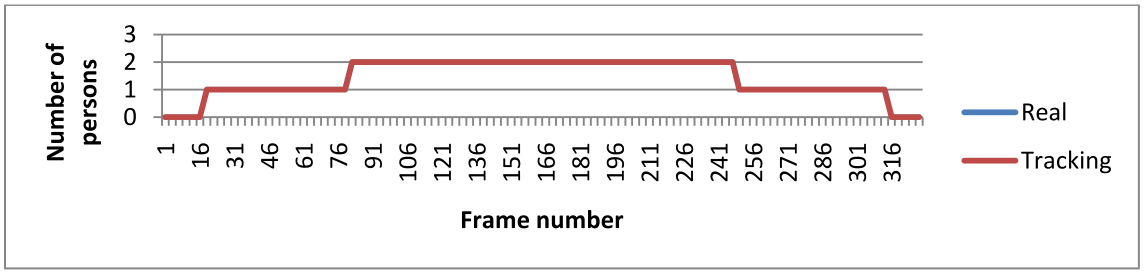


Figure A.1 – Number of detected objects: *Hall Monitor* – 10fps

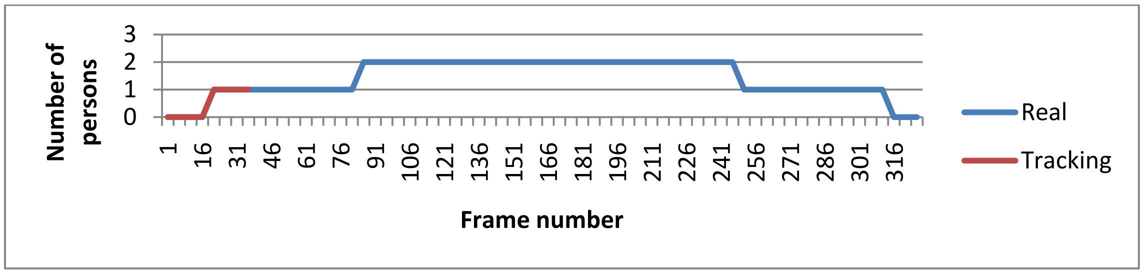


Figure A.2 – Number of detected objects: *Hall Monitor* – 6fps

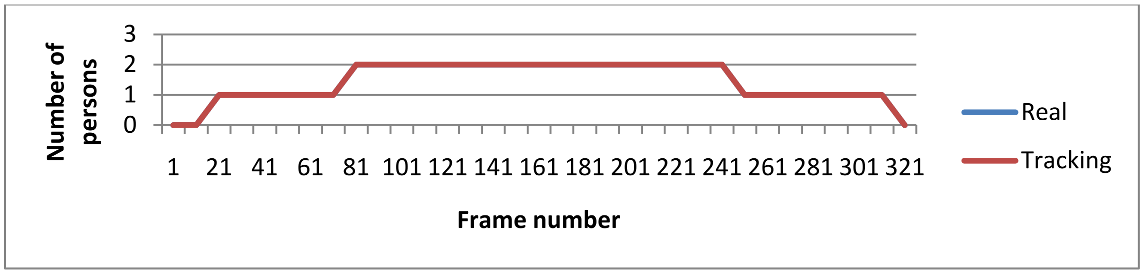


Figure A.3 – Number of detected objects: *Hall Monitor* – 3fps

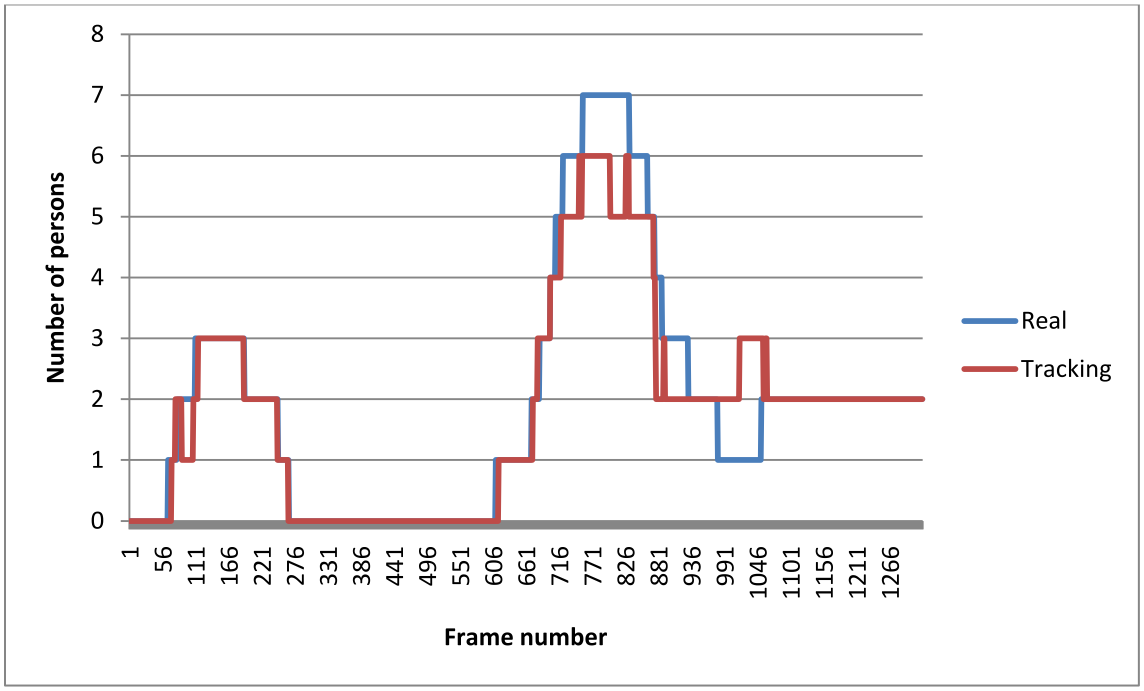


Figure A.4 – Number of detected objects: *Airport* – 16fps

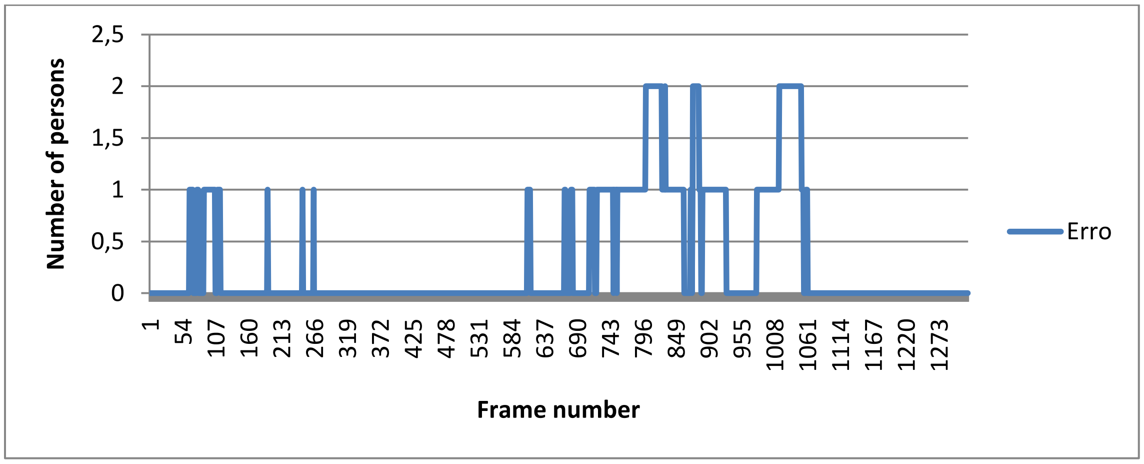


Figure A.5 – Object detection error: *Airport* – 16fps

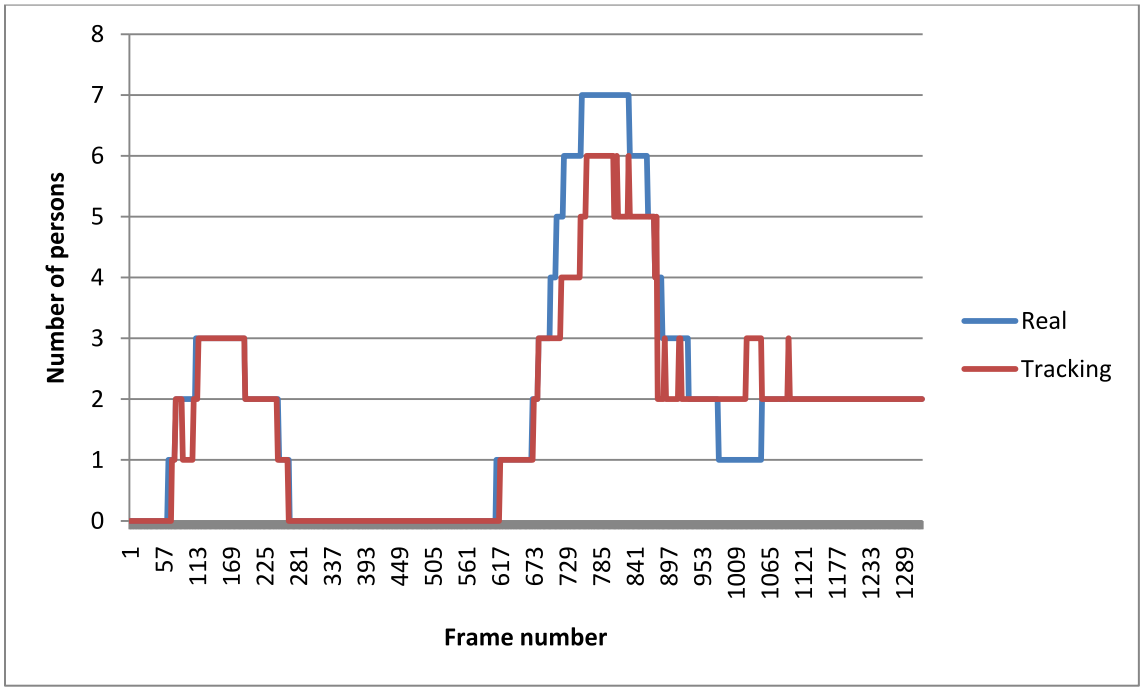


Figure A.6 – Number of detected objects: *Airport* – 8fps

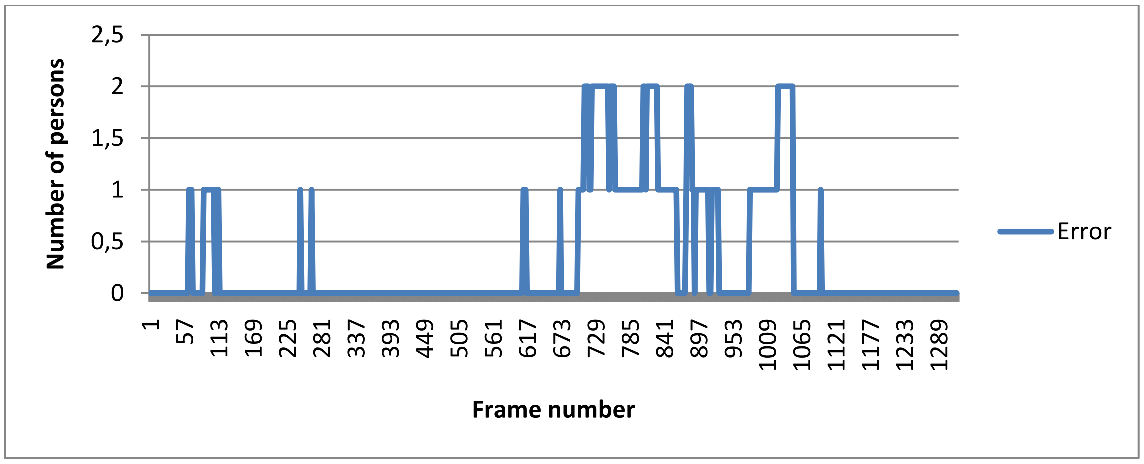


Figure A.7 – Object detection error: *Airport* – 8fps

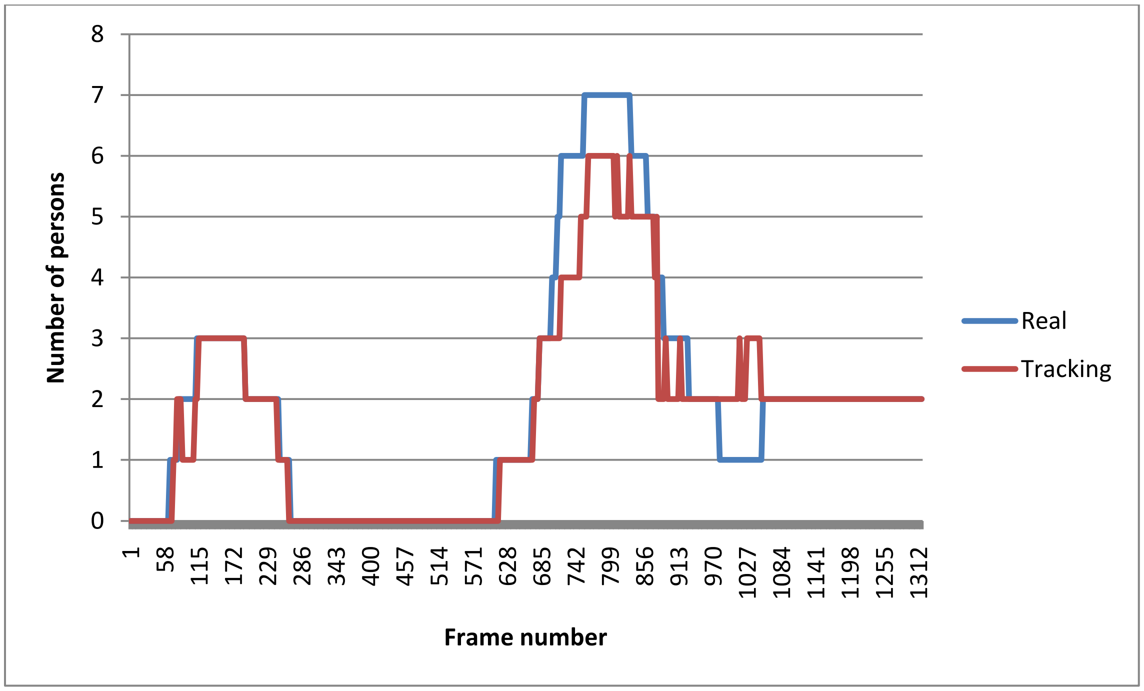


Figure A.8 – Number of detected objects: *Airport* – 4fps

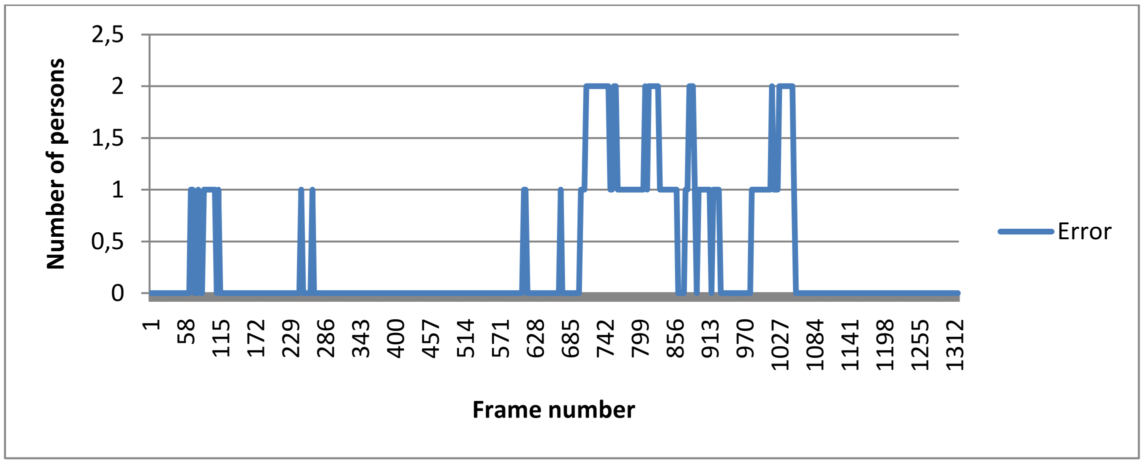


Figure A.9 – Object detection error: *Airport* – 4fps

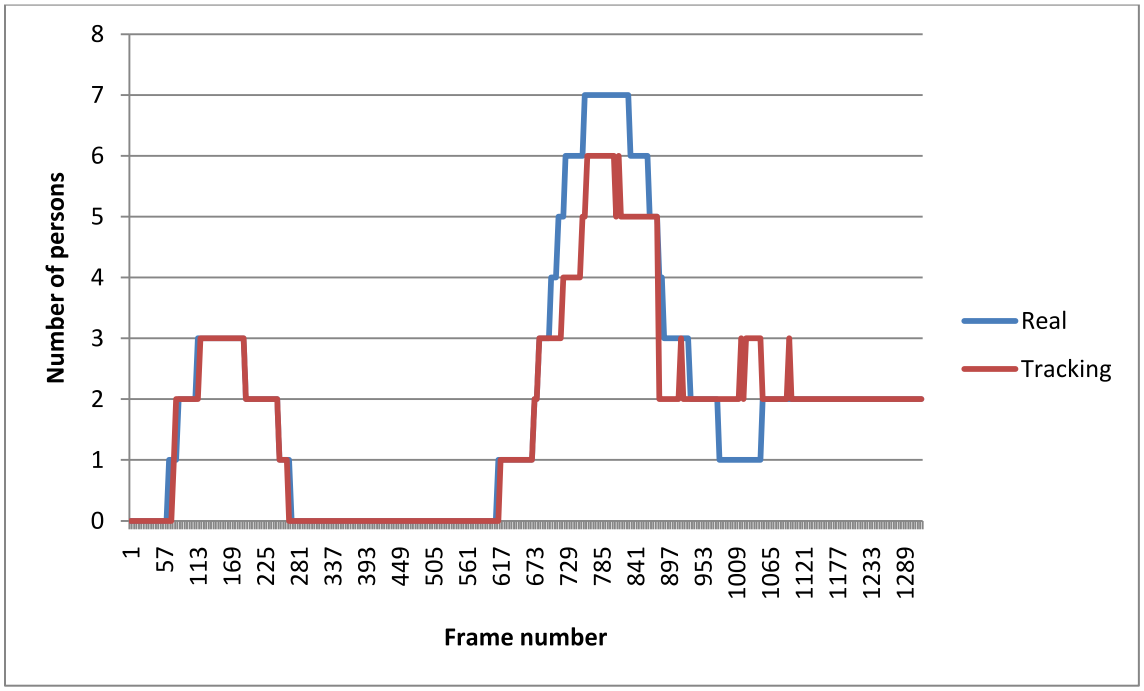


Figure A.10 – Number of detected objects: *Airport* – 3.2fps

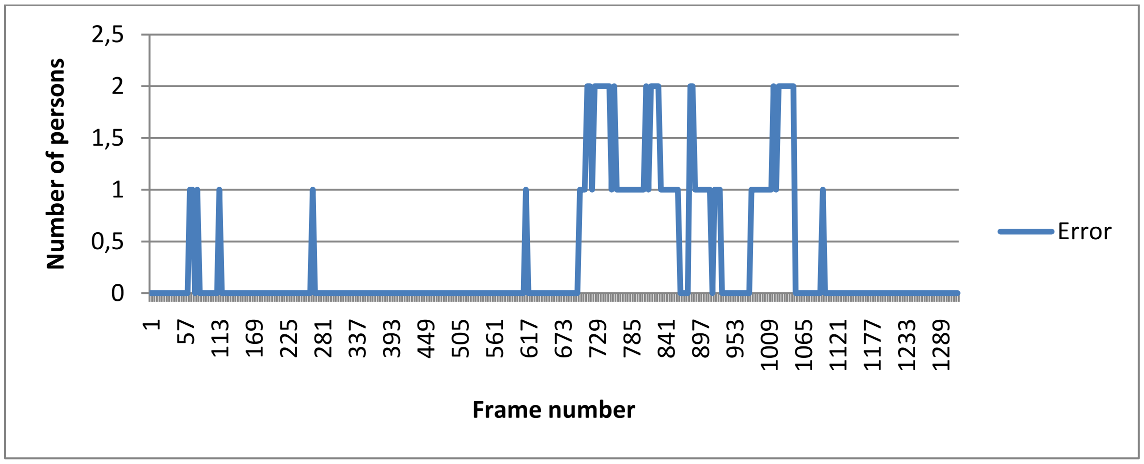


Figure A.11 – Object detection error: *Airport* – 3.2fps

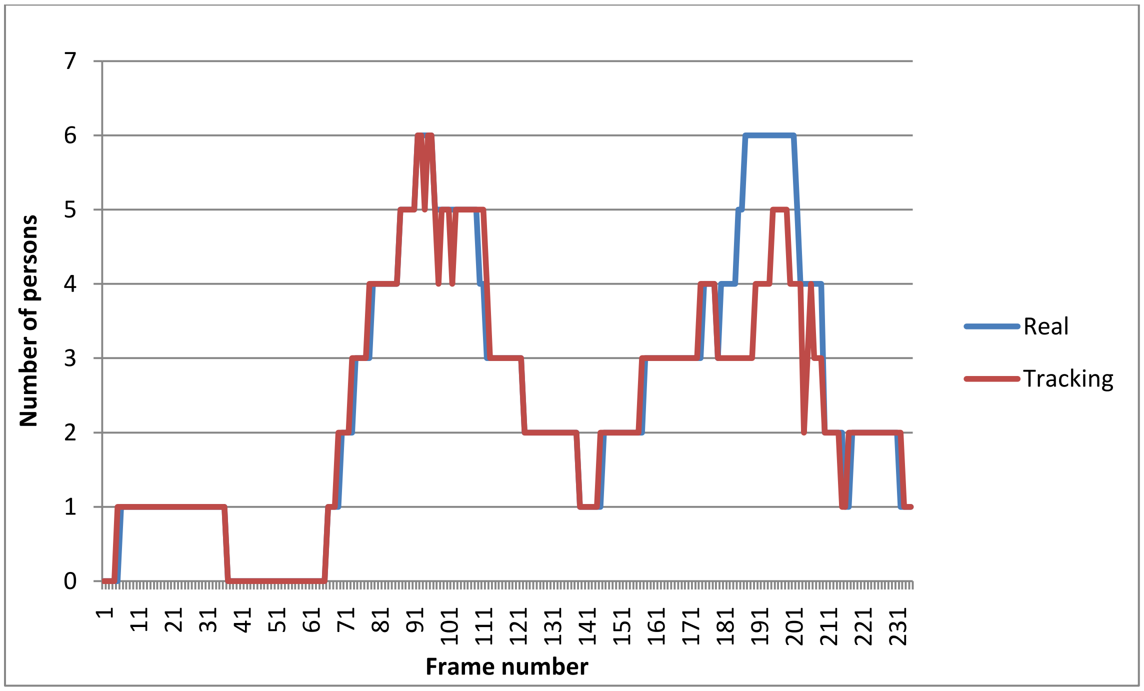


Figure A.12 – Number of detected objects: *Train Station* – 3fps

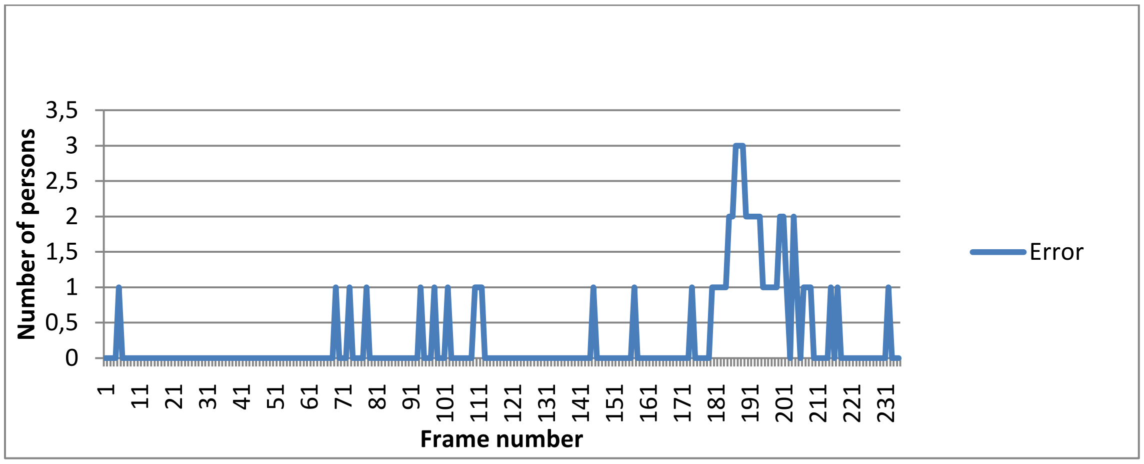


Figure A.13 – Object detection error: *Train Station* – 3fps

Table A.1 – Individual object identification changes and successful detection ratio (*Hall Monitor* – 3fps)

	10 fps		6 fps		3 fps	
	Identification Changes	Successful Detection	Identification Changes	Successful Detection	Identification Changes	Successful Detection
Object 1	0	100%	0	100%	0	100%
Object 2	0	100%	0	100%	0	100%
Mean	0	100%	0	100%	0	100%

Table A.2 – Individual object identification changes and successful detection ratio (*Airport* – 3.2fps)

	16 fps		8 fps		4 fps		3.2 fps	
	Identification Changes	Successful Detection	Identification Changes	Successful Detection	Identification Changes	Successful Detection	Identification Changes	Successful Detection
Object 1	2	99,45%	2	96,70%	2	95,08%	0	100,00%
Object 2	1	99,11%	1	100,00%	1	100,00%	0	100,00%
Object 3	0	96,77%	0	97,40%	0	96,15%	0	94,87%
Object 4	0	97,76%	1	97,30%	1	97,33%	0	98,18%
Object 5	2	99,51%	1	99,04%	1	100,00%	1	100,00%
Object 6	0	100,00%	1	100,00%	1	100,00%	1	100,00%
Object 7	0	100,00%	1	100,00%	1	100,00%	1	100,00%
Object 8	0	100,00%	0	100,00%	0	96,00%	0	100,00%
Object 9	0	38,86%	0	48,11%	0	46,48%	0	45,28%
Object 10	0	89,39%	0	91,18%	0	88,64%	0	90,91%
Mean	0,5	92,09%	0,7	92,97%	0,7	91,97%	0,3	92,92%

Table A.3 – Individual object identification changes and successful detection ratio (*Train Station* – 3fps)

	3 fps	
	Identification Changes	Successful Detection
Object 1	0	100,00%
Object 2	0	100,00%
Object 3	0	97,56%
Object 4	1	94,74%
Object 5	1	100,00%
Object 6	2	100,00%
Object 7	2	100,00%
Object 8	0	100,00%
Object 9	1	100,00%
Object 10	3	100,00%
Object 11	4	100,00%
Object 12	2	100,00%
Object 13	4	100,00%
Object 14	2	100,00%
Object 15	0	100,00%
Mean	1,46	99,49%

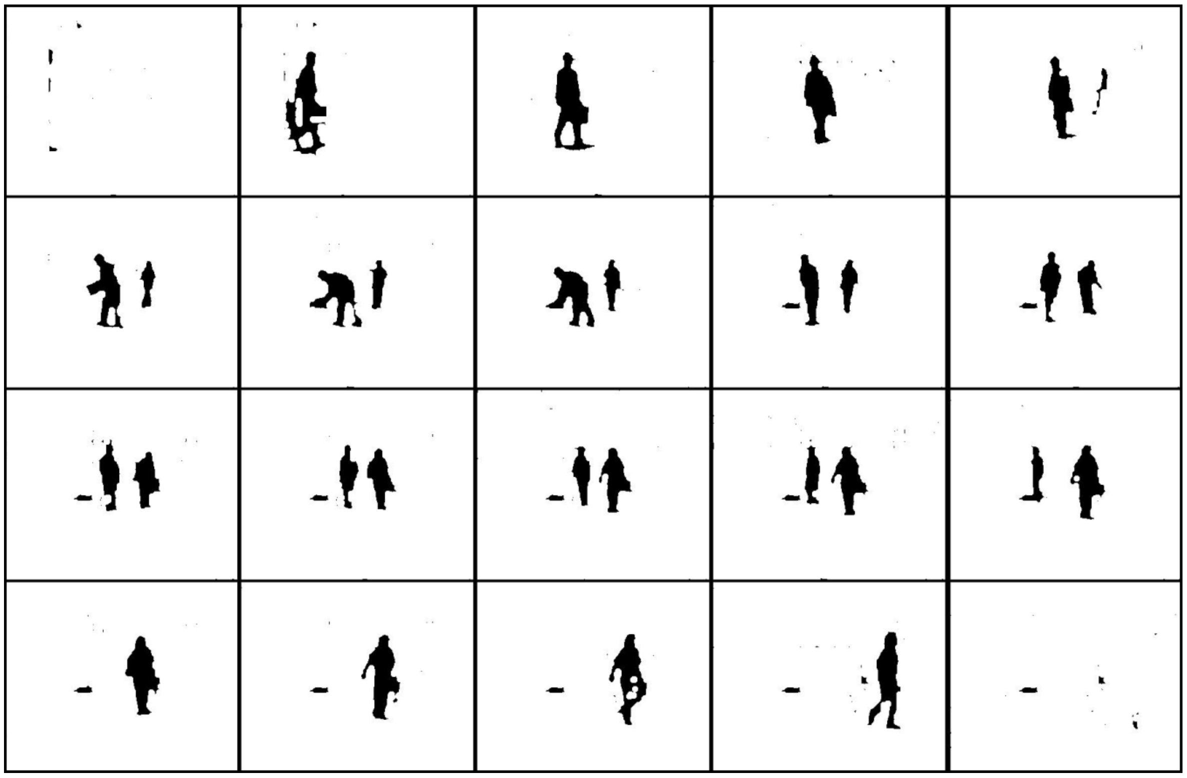


Figure A.14 – Background subtraction of *Hall Monitor* sequence (at each 15 frames)

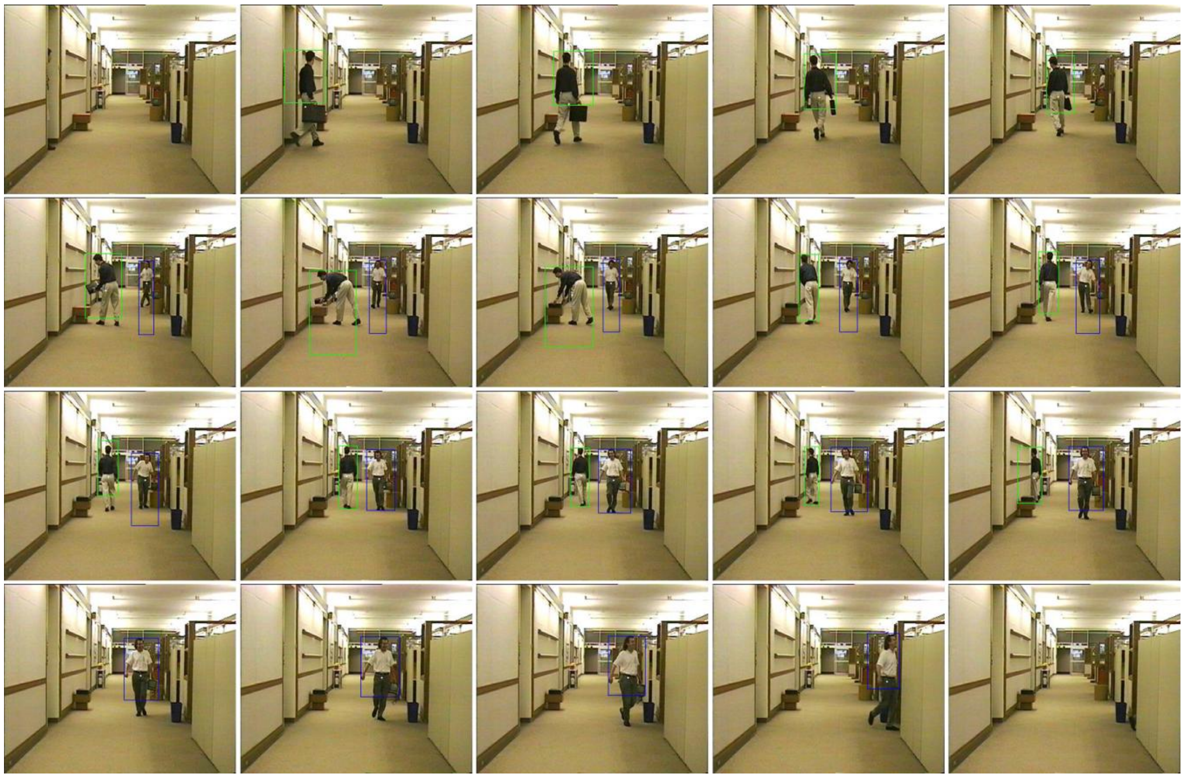


Figure A.15 – Tracking in *Hall Monitor* sequence (at each 15 frames)

Annex B

Curriculum Vitae

Europass-Curriculum Vitae

Informação pessoal

Apelido(s) / Nome(s) próprio(s)	**Ferreira, Marco Paulo Fernandes**
Morada(s)	Av.ª de Fitares Nº 33 1º Esq., 2635-453, Rio de Mouro, Portugal
Telefone(s)	(351-21) 015 27 56 Telemóvel: (351-96) 777 31 16
Correio(s) electrónico(s)	marco.ferreira85@gmail.com marco.ferreira@external.thalesgroup.com
Nacionalidade	Portuguesa
Data de nascimento	05-09-1985
Sexo	Masculino

Emprego pretendido / Área funcional

Telecomunicações, Informática e Multimédia

Experiência profissional

Datas	De Outubro de 2008 a Julho de 2009
Função ou cargo ocupado	Prestação de Serviços (Programador/Investigador)
Principais actividades e responsabilidades	Desenvolvimento de Projecto de Sistema de Videovigilância Inteligente (TRAPLE - TRAcking People in Large Environments) Desenvolvimento da plataforma de processamento de vídeo (Video Object Tracking)
Nome e morada do empregador	THALES Security Solutions & Services - Portugal Rua Calvet Magalhães 254, Bloco A 2774-550 Paço de Arcos Portugal

Tipo de empresa ou sector	Fabrico e da comercialização de equipamentos e de sistemas electrónicos destinados aos sectores da aeronáutica, naval e da defesa. - sistemas de defesa (42,5%) - sistemas aeronáuticos e espaciais (29,2%) - sistemas de segurança (27,8%) - outros (0,5%)
Educação e formação	
Datas	De Setembro de 2007 a Setembro de 2009
Designação da qualificação atribuída	Frequência em Mestrado de Engenharia de Telecomunicações e Informática
Principais disciplinas/competências profissionais	Computação Gráfica Processamento de Sinal Multimédia Inteligência e Gestão de Redes e Serviços Programação em Rede Programação Avançada Engenharia de Software Sistemas e Redes de Comunicação Móvel Avançado Gestão Financeira de Empresas e Projectos
Nome e tipo da organização de ensino ou formação	Instituto Superior de Ciências do Trabalho e da Empresa Av.ª das Forças Armadas 1649-026 Lisboa Portugal
Nível segundo a classificação nacional ou internacional	16 valores (Provisório - Dissertação ainda não avaliada)
Datas	De Setembro de 2003 a Julho de 2007
Designação da qualificação atribuída	Licenciatura em Engenharia de Telecomunicações e Informática

Principais disciplinas/competências profissionais	Redes Digitais (Fundamentos, Sistemas, Aplicações, Serviços, Segurança, Multimédia e Gestão) Processamento de Sinal Multimédia Sistemas e Redes de Comunicação para Móveis Multiplexagem, Comutação e Integração de Serviços Sistemas de Telecomunicações Rádio Sistemas de Telecomunicações Guiados Programação Orientada para Objectos Programação Concorrente e Distribuída Bases de Dados Inteligência Artificial Tecnologias para Sistemas Inteligentes Teoria do Sinal Modulação e Codificação Circuitos e Sistemas Electrónicos para TIC Electrónica Programada para TIC Sistemas Operativos
Nome e tipo da organização de ensino ou formação	Instituto Superior de Ciências do Trabalho e da Empresa Av.ª das Forças Armadas 1649-026 Lisboa
Nível segundo a classificação nacional ou internacional	Treze valores
Datas	De Setembro de 1999 a Junho de 2002
Designação da qualificação atribuída	Curso de Inglês do Nível 6 (nível de preparação para o First Certificate Exam)
Principais disciplinas/competências profissionais	Língua Inglesa
Nome e tipo da organização de ensino ou formação	Instituto de Línguas de Rio de Mouro R. Óscar Monteiro Torres, N.º 27 c/v – 1 E 2735 Rio de Mouro
Nível segundo a classificação nacional ou internacional	Bom
Datas	De Abril de 2000 a Junho de 2000

Designação da qualificação atribuída	Curso de Informática Fundamental
Principais disciplinas/competências profissionais	Microsoft PowerPoint 97 Microsoft Access 97 Corel Draw 8.0 Utilização Alargada da Internet
Nome e tipo da organização de ensino ou formação	Classe 86 – Formação e Serviços, LDA R. Capitão Salgueiro Maia Lt.4 r/c 2350 Torres Novas
Nível segundo a classificação nacional ou internacional	Dezassete valores
Datas	De Agosto de 1999 a Outubro de 1999
Designação da qualificação atribuída	Curso de Informática Fundamental
Principais disciplinas/competências profissionais	Iniciação à Informática Windows 98 Microsoft Word 97 Microsoft Excel 97 Introdução à Internet e Navegação
Nome e tipo da organização de ensino ou formação	Classe 86 – Formação e Serviços, LDA R. Capitão Salgueiro Maia Lt.4 r/c 2350 Torres Novas
Nível segundo a classificação nacional ou internacional	Dezoito valores

Aptidões e competências pessoais

Língua(s) materna(s)	**Português**
Outra(s) língua(s)	**Inglês, Francês**

Auto-avaliação	Compreensão				Conversação				Escrita	
Nível europeu (*)	Compreensão oral		Leitura		Interacção oral		Produção oral			
Inglês	B2	Utilizador independente	C1	Utilizador experiente	B1	Utilizador independente	B1	Utilizador independente	B1	Utilizador independente
Francês	A1	Utilizador elementar	A2	Utilizador elementar	A1	Utilizador elementar	A1	Utilizador elementar	A1	Utilizador elementar

(*) Nível do Quadro Europeu Comum de Referência (CECR)

Aptidões e competências sociais	Espírito de equipa Capacidade de adaptação a ambientes multiculturais
Aptidões e competências informáticas	Domínio de software Office™ (Word™, PowerPoint™, Excel™, Visio™, Access™) Domínio de linguagens de programação (MATLAB, Java, C++, SQL, Lisp, Prolog, Assembley, etc) Domínio de tecnologias Web (HTML, JSP, XML, Javascript, CSS, Flash, etc) Conhecimentos básicos de aplicações gráficas (Corel Draw™, Photoshop™)
Aptidões e competências artísticas	---
Outras aptidões e competências	Praticante de musculação
Carta de condução	Carta de condução de categoria B (veículos ligeiros), A1 e A (motociclos)
Informação adicional	---
Anexos	Certificado de Licenciatura de Telecomunicações e Informática Diplomas referentes aos cursos de Informática Fundamental Certificado de Curso de Inglês do nível 6

References

[1] M. Shah, O. Javed, and K. Shafique, "Special issue on video communications, processing, and understanding for third generation surveillance systems," *Proceedings of the IEEE*, vol. 89, pp. 1355-1539, 2001.

[2] ISO/IEC 15938-1, "Information technology - Multimedia content description interface - Part 1: Systems," 2002.

[3] M. Ferreira, et al., "System architecture and process for tracking individuals in large crowded environments," European Patent 09164345.2 - 1229, Jul. 01, 2009.

[4] Z. Q. Huang and Z. Jiang, "Tracking camouflaged objects with weighted region consolidation," in *Proc. Digital Image Computing: Techniques and Applications*, 2005, pp. 24-24.

[5] K. M. El-Sayed and S. H. Ahmed, "Moving object detection in spatial domain using background removal techniques – State-of-art," *Recent Patents on Computer Science 2008*, vol. 1, pp. 32-54, 2008.

[6] M. Shah, O. Javed, and K. Shafique, "Automated visual surveillance in realistic scenarios," *IEEE Multimedia*, vol. 14, pp. 30-39, 2007.

[7] Y. D. Wang, J. K. Wu, A. A. Kassim, and W. Huang, "Data-driven probability hypothesis density filter for visual tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1085-1095, 2008.

[8] J. Vermaak, S. J. Godsill, and P. Perez, "Monte Carlo filtering for multi target tracking and data association," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 309-332, 2005.

[9] Y. Bar-Shalom and T. Fortmann, *Tracking and data association*. Academic Press.,

1988.

[10] Y.-D. Wang, J.-K. Wu, A. A. Kassim, and W.-M. Huang, "Tracking a variable number of human groups in video using probability hypothesis density," in *Proc. 18th International Conference on Pattern Recognition ICPR 2006*, vol. 3, 2006, pp. 1127-1130.

[11] B. N. Vo, S. Singh, and A. Doucet, "Sequential Monte Carlo methods for multitarget filtering with random finite sets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 1224-1245, 2005.

[12] E. Maggio, M. Taj, and A. Cavallaro, "Efficient multitarget visual tracking using random finite sets," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1016-1027, 2008.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 564-577, 2003.

[14] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000, pp. 142-149.

[15] S. Yonemoto and M. Sato, "Multitarget tracking using mean-shift with particle filter based initialization," in *Proc. 12th International Conference Information Visualisation IV '08*, 2008, pp. 521-526.

[16] C.-W. Juan and J.-S. Hu, "A new spatial-color mean-shift object tracking algorithm with scale and orientation estimation," in *Proc. IEEE International Conference on Robotics and Automation ICRA 2008*, 2008, pp. 2265-2270.

[17] Z. Li, Q. L. Tang, and N. Sang, "Improved mean shift algorithm for occlusion pedestrian tracking," *Electronics Letters*, vol. 44, pp. 622-623, 2008.

[18] R. V. Babu and A. Makur, "Kernel-based spatial-color modeling for fast moving object tracking," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2007*, vol. 1, 2007, pp. I-901--I-904.

[19] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on Lie algebra," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2006, pp. 728-735.

[20] Y. Wang, Y. Liang, C. Zhao, and Q. Pan, "Adaptive multi-cue kernel tracking," in *Proc. IEEE International Conference on Multimedia and Expo*, 2007, pp. 1814-1817.

[21] F. Azevedo, "Video object detection in large public environments: Low-complexity video object detection solution," Master's Dissertation, ISCTE-IUL, Lisbon, 2009.

[22] R. Haralick and L. Shapiro, "Computer and robot vision," *Addison-Wesley*, vol. I, pp. 28-48, 1992.

[23] http://www.vision.caltech.edu/bouguetj/calib_doc.

[24] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.