

# IMAGE ANALYSIS ARCHITECTURES AND TECHNIQUES FOR INTELLIGENT SURVEILLANCE SYSTEMS

X. Desurmont<sup>a</sup>, A. Bastide<sup>a</sup>, C. Chaudy<sup>a</sup>, C. Parisot<sup>a</sup>, J.F. Delaigle<sup>a</sup> and B. Macq<sup>b</sup>

{desurmont, bastide, chaudy, parisot, delaigle}@multitel.be

<sup>a</sup>Multitel A.S.B.L, Belgium

<sup>b</sup>Université Catholique de Louvain, Belgium

## Abstract

Video security is becoming more and more important today, as the number of installed cameras can attest. There are many challenging commercial applications to monitor people or vehicle traffic. The work reported here has both research and commercial motivations. Our goals are first to obtain an efficient intelligent system that can meet strong industrial surveillance system requirements and therefore be real-time, distributed, generic and robust. Our second goal is to have a development platform that allows researchers to conceive and easily test new vision algorithms thanks to its modularity and easy set-up. This paper focuses on the image analysis modules. It considers the different kind of inputs, algorithm models as well as delay and the need of generality.

**Keywords:** multi-camera, distributed architecture, real-time, computer vision, intelligent visual surveillance, tracking.

## 3. Introduction

Video surveillance is a large market as the number of installed cameras can attest. Nevertheless, there is still a need for complete and generic systems that can be inserted in an existing camera network (e.g. CCTV) to increase intelligence and handle automatic processing. Examples of challenging applications [1] are monitoring metro stations [2] or detecting highways traffic jams, intelligent content access and detection of loitering. The requirements for these systems are to be network-connected, multi-cameras, modular, the display must be user-friendly and the overall system must be highly reliable and robust. The work reported here has both research [3][4][5] and industrial motivations. In this paper we present a generic, flexible and robust approach for an intelligent real-time video-surveillance system.

The paper is organized as follows: section 2 gives a global description of the system and its main characteristics, section 3 goes deeper in the understanding of each underlying module, section 4 is devoted to the image analysis module, section 5 shows some typical results and section 6 presents the main conclusions.

## **4. System overview**

The CCTV system presented in this paper is based on a digital network architecture. This kind of system can be deployed in a building, for instance, or can be connected to an existing data network. Basically, the system is composed of computers connected together through a typical LAN. The various cameras are plugged either into an acquisition board on a PC or directly in the local network hub for IP cameras. A human computer interface and a storage space are also plugged in this system. The main advantage of such architecture is its flexibility. The logical architecture has been designed in a modular way to allow a fair resource allocation over the cluster. Future additional needs in computing power will be simply addressed by adding a PC in the cluster. The major components of the modular architecture are presented in Fig 1. In this implementation, each software module is dedicated to a specific task (e.g. coding, network management, etc.) and will be discussed in section 3. The various processing modules are distributed on the PC units according to a configuration set-up. In this set-up, the operator defines the architecture and the scheduling of the distributed system and moreover, they are able to customize the action of the different modules (e.g. the vision processing units require a lot of parameters). A master process is assigned the task of communicating the configuration to all the PC units. A change in the distribution of the different tasks between the units simply requires changing the configuration file (no need to compile again). The robustness of the overall system is provided by the logical architecture. It manages the various problems that can arise: a network transmission interruption, a hard drive stopping, etc. Moreover, we can improve the overall system performances if we dedicate a vision task to a co-processor.

### **4.1. Data management**

Because of the real-time and low resource requirements, the system basic structure for data management between modules handles the concurrent access for sharing information for multiple producers and consumers. It optimises the memory needs by avoiding copies of data, the network communication (TCP/IP) for multiple instances of the data (compression handling if necessary), the dynamic connection and access to the stream, the buffer to deal with processing peaks, the propagation of consumers needs to producers to optimise CPU resources (e.g. if no module uses a segmentation result, the segmentation module will be advised not to produce it), the monitoring of data streams (performance, rate), the dispatching priority of data towards specific modules (it helps to process a real-time framework by decreasing latency). Fig 2 represents an example of a

stream of data (control, source image, dynamic descriptors of scene, events) and a typical processing. Image acquisition is performed in each frame (25 fps), but tracking is done every other frame (and therefore the interpretation too).

## **4.2. Data handling**

Video surveillance applications need to be real-time, because of the security requirement of minimum time reactions. Therefore video-surveillance requires low delay and timing constraints for processing. Classical real-time systems usually use internal periodical loop (e.g. a new image is acquired every 40 ms and should be processed before the next one, thus the process should last less than 40 ms). If the process costs more than 40 ms for one of the stages of a pipeline (if there is no pipeline, we consider the system as a unique-stage pipeline), a delay is introduced and grows for every new frame, this delay is limited to a maximum value (e.g. 200 ms), if it exceeds this limit, the system fails (e.g. the memory buffer is full, some data is lost and processing of this data, such as integration or derivation, will fail). We overcome these kinds of problems by adding to each observation (image) a time-stamp, and then take it into account for processing (e.g. background adaptation, objects speed computing, etc.). Thus, if processing resources exceed the available system resources, some information is ignored but without disturbing the integrity of the system because all algorithms can handle loss of data as soon as the time-stamp of every data is known.

## **5. System components description**

The various modules of the software part of the system are explained here. We successively explain the acquisition module, the codec module, the network module, the storage module. We do not describe here the graphical user interface (GUI) previously reported in [6], but clearly it is an important module from the user's point of view. Section 4 deals with the image analysis module.

### **5.1. Acquisition**

We are currently able to handle several protocols for image acquisition: IP (JPEG and MJPEG), IEEE1394 (raw and DV), wireless (analogue, WiFi) and composite (PAL, NTSC, SECAM). A time-stamp is attached to each frame at grabbing time, as this information is useful in the subsequent processing stages. For file replay, we use the Ffmpeg [7] library that handles many codecs.

## **5.2. Codec**

The goal of the coding process is to have a good compromise between compression ratio and bandwidth utilisation. The choice of quality of image is done according to the viewer and also to image processing requirements when it occurs after compression. We propose here a MPEG-4 SP compression scheme. Because video-surveillance scenes are quite static when cameras are fixed, compression methods suppressing temporal redundancy, such as MPEG4 SP, are more efficient than classical MJPEG encoding [8]. This technique allows us to transmit up to 20 CIF (352×288) video streams at 25 fps on a typical 100baseT network.

## **5.3. Network**

Having a distributed system implies an efficient use of the bandwidth. We have seen that the various modules related to a video signal can operate on several computers. For example, we can have the acquisition on computer 1, the storage on computer 2 and the display on computer 3. We have chosen a multicasting technique to solve the bandwidth utilisation problem. Each video source has an associated multicast channel. This multicast channel is accessed through a UDP connection by every module that needs the video input. Since UDP does not offer a quality of service (QoS), we have developed a protocol that can detect when a transmission failure occurs. We guarantee small delays because the network load is controlled in order to avoid buffer queuing. This delay is small enough to be imperceptible for the user.

## **5.4. Storage**

The storage module has to deal with the significant amounts of data to store and it must allow 24 hours a day storage. This module has two levels: level 0 is a classical storage process with MPEG4 technology. This level stores a CIF video flow at 25 fps for three days on a 40 GB hard drive. We can further improve this number if we allow a two-pass encoder to have a constant quality stream. Up to this point, we have a constant bandwidth stream. Level 1 is an intelligent storage process. It stores only events of interest that the user has defined. This level saves a substantial storage space, typically it could be from 70 to 95%. It allows also a fast search to retrieve a stored sequence.

## **6. Image analysis module**

High-level interpretation of events within the scene requires low-level vision computing of the image and of the moving objects. The generation of a representation for the appearance objects in the scene is usually needed. For our system, the architecture of the vision part is divided into three main levels of computation that achieve the interpretation (Fig 3): image level (acquisition, image filtering, background evaluation and segmentation), blob level (description, blobs filtering, matching, tracking description and filtering), event level (tracking analysis, finite state machine, performance evaluations). In the general state-space approach, the global problem definition is to predict the state-variables (position of objects, events, etc.) through observations from sensors (cameras, infra-ray detector, etc.).

### **6.1. Calibration and 2D and 3D context**

In many situations, it is of interest to have a calibration of the camera (e.g. for multiple cameras application or when doing 3D with for ground plane area). An easy and manual tool for calibration (Fig 4) has been developed with the same procedure described in [9], it takes around 5 minutes to calibrate a camera, and it has to be done every time the camera has moved. We also handle radial deformation via the OpenCV [10] library. For fixed cameras, a 2D context can be defined by users identifying areas in the image such as input/output regions, zones to ignore, etc. A 3D context (Fig 5) can be set up to describe scenes in 3D (the floor, the walls and the objects present at beginning of sequence). The system also allows a user to define contextual information either globally (information corresponding to many cameras viewing the same scene) or specifically. This contextual information is represented by means of 2D polygons on the image, each of them having a list of attributes (entering, exiting, noisy, occlusion etc.) that is fed to the image analysis module to help the scenario recognition process.

### **6.2. Image filtering**

After acquisition, the current image is filtered in the pixel domain to decrease spatial high-frequency noise. Usually the image is convoluted via a linear filter with a Gaussian kernel, but in our scheme we use the optimised 2-pass exponential filter [11] for reduced processing time. The image is then downsized, to reduce the computational cost of the segmentation process. When compressed video streams are used, we do not consider the loss of quality as we have not addressed this problem for the moment.

### 6.3. Segmentation

The common bottom-up approach for segmenting moving objects uses both background estimation and foreground extraction (usually called background subtraction) [12].

The typical problems of this approach are changes of illumination, phantom objects, camera vibrations, and other natural effects such as tree branches moving. In the system it is possible to use different reference image models representing the backgrounds (low pass temporal recursive filter, median filter, uni-modal Gaussian, mixture of Gaussians [13], vector quantisation [14]). In the literature, the most basic background model is updated to take small illumination changes into account by:

$$B_t = \alpha I_t + (1 - \alpha) B_{t-1} \quad (1)$$

Equation (1) is computed for each pixel of the image.  $\alpha$  is a parameter of the algorithm,  $B_{t-1}$  is the previous background pixel value,  $I_t$  is the current image pixel value and  $B_t$  is the updated current background value. The foreground is then typically extracted through a threshold  $T$  on the difference between  $B_t$  and  $I_t$ . A pixel is foreground when (2) is true:

$$|B_t - I_t| > T \quad (2)$$

This model is quite simple and can handle basic small or slow variations of background. But real applications need more improvement in the model as well as the conceptualisation to be more robust to common noises such as monitor flickering or moving branches in trees. However, these algorithms should satisfy the non-periodic processing framework of the system as described in 2.2. The segmentation is fully configurable as one can choose between all types of background models and parameters on-line (during the running of the system). For all these models, we have the following two methods: estimate foreground, update background. We also add the possibility to predict the background state at a given time and also the possibility of a feedback to update selectively the background [15]. Fig 6 explains why it is useful to have different stages to update and estimate the background. At time  $t_1$ , the model has the history from 0 to  $t_1$  (0: $t_1$ ) and is defined as  $B_{0:t_1}$ . Then there is an update at time  $t_2$  and it became  $B_{0:t_2}$ . One can ask the estimation of the background at time  $t_3$  which is  $B_{0:t_2}(t_3)$  and compare it to the value of the pixel at this time  $I_{t_2}$  to know if it is to be classified as a foreground or background pixel. In this example (Fig 6) the background is bi-modal and the light is increasing with time.

Let us define  $t_1 < t_2$  two times after image acquisition. The current image  $I_{t_2}$  has just been acquired at time  $t_2$  and we have  $B_{0,t_1}$ , the background model updated at time  $t_1$ . The process is divided in several parts (Fig 7, 8):

1. An *a priori* belief map is computed with a normalized distance between from  $I_{t_2}$  and  $B_{0,t_1}$ . This distance is dependant of the background model (e.g. for mixture of Gaussians, it is the difference between the pixel value and the mean of the nearest Gaussian). The most probable foreground is then computed: for all pixel in belief map pixel greater than 0.5, the pixel is *a priori* (without knowledge of neighbourhood) considered as foreground.
2. If the pixel is inside a closed outline of pixel with a belief  $> 0.5$ , the neighbourhood rule will consider it as a foreground pixel. A second neighbourhood rule is applied: a pixel cannot be foreground if there is no path of connected *a priori* foreground pixel until a pixel of belief  $> 0.8$ . These two rules permit hysteresis phenomena to decrease noise in the foreground.
3. Then two steps of decision are made at two different stages of the process to filter foreground objects: after the description (ignore zone, object too small, phantom object, etc.) and after the tracking (integrated object, etc.). We do not consider here these processes (see subsections 4.4-4.7). After them, some foreground objects from step 2) are discarded or some non-detected objects are added to the structure of data that contains foreground map. At this time, it is called update map.
4. The background model is then updated with image at time  $t_2$  for regions of the scene that the update map defines as background.

In Subsections 4.4-4.7, we will briefly investigate the processes that occur between estimation 1. and 4.

## 6.4. Blobs description and filtering

The aim of blobs description and filtering is to make the interface between foreground extraction and tracking and to simplify the information. The description process translates video data into a symbolic representation (i.e. descriptors). The goal is to reduce the amount of information to what is necessary for the tracking module. The description process calculates, from the image and the segmentation results at time  $t$ , the  $k$  different observed features  $f_{k,i}^t$  of a blob  $i$ : 2D position in image, 3D position in the scene, bounding box, mean RGB colour, 2D visual surface, inertial axis of blob shape, extreme points of the shape, probability to be a phantom blob, etc. At this point, there is another filtering process to remove small blobs, blobs in an area of the image not considered, etc. Other model-based vision descriptors can also be integrated for specific application such as vehicle or human 3D model parameters.

## 6.5. Tracking algorithm

As the other modules, the tracking part of the system is flexible and fully parametrical on-line. The set-up should be done for a trade-off between computational resources, needs of robustness and segmentation behaviour. It is divided into four steps that follow a straightforward approach:

- Prediction,
- Cost matrix,
- Matching decision(s),
- Tracks update(s).

Note that there are multiple predictions and cost matrices when the last matching decision is not unique, and there are only multiple matching decisions for some matching algorithms in MHT (multiple hypothesis tracking [16]). Fig 9 briefly explains the architecture.

-Prediction: The estimation process is very basic. It predicts the blob's features (position, colour, size, etc.). It is integrated as a recursive estimator to handle various estimator cores like Kalman filter [14], explicit Euler, etc. The estimation is then taken as the maximum a posteriori probability (MAP).

$$\hat{f}_{k,i}^t = \underset{f_{k,i}^t}{\operatorname{argmax}} p(f_{k,i}^t | f_{k,i}^{0:t-1}) \quad (3)$$

-Cost matrix computation: The aim of the matrix is to identify a cost  $C_{ij}$  for a matching between blobs  $i$  and  $j$  in current and precedent frame (or the estimation of it in current frame). The cost, which is low when blob looks similar and high otherwise, is performed by a procedure based on all the features of a blob. e.g. if images are colour, the cost depends on colour dissemblance. In fact,  $C_{ij}$  is a linear combination of weighted distance of each feature (4). If  $\alpha_k=0$  the distance  $d_k$  is not computed. Currently,  $\alpha_k$  is defined by an expert during the set-up of the system, but in the future we hope to implement a learning algorithm that maximises some performance aspect of the system.

$$C_{i,j} = \sum_{k=1 \dots N} \alpha_k d_k(F_k^i, F_k^j) \quad (4)$$

-Matching: From a given cost matrix, there are many potential hypothesis matrices for the real matching matrix [16]. The library we designed handles three matching algorithms that can be executed (one at a time) : “full multi-hypothesis matching”, “light multi-hypothesis matching” and “nearest neighbour matching”:

- “full multi-hypothesis matching”: all hypothesis of matching are tested, the K best global hypothesis are kept. It takes a significant amount of computation time. e.g. from the K best global previous hypothesis in a case of n blobs per frame, the number of hypothesis to test is  $K \times 2^{n \times n}$ . In practice, some hypothesis are more likely to occur than others. That is why we define the two heuristics below.
- “light multi-hypothesis matching”: it reduces the number of matching hypothesis matrix. For a given blob in the current image, it only performs matching hypothesis between the N blobs in the precedent image that have the lowest costs.
- “nearest neighbour matching”: this should be the simplest matching algorithm. Every blob from the precedent frame is matched to its nearest blob (lowest cost) in the current frame. Furthermore every blob from the current frame is matched to its nearest blob in the precedent frame. However, if the cost between two blobs exceeds a threshold given as parameter, the match is removed.

Note that at each iteration one can permute the matching algorithm with another. Note also that an ambiguity multi-disjoint-hypothesis tracking can be developed to efficiently reduce the combinatorics [16].

## 6.6. Tracking description and filtering

The aim of tracking description and filtering is to make the interface between the tracking and the analysis processes and to simplify the information. The tracking description converts the internal tracking result to a graph (Fig 10), then it adds some useful information to the matching data. It computes the time of life of every blob of a track (i.e. the duration of the track from apparition to the specify blob), the time before death (i.e. the duration of the track to disappearance of the specify blob). It also describes a piece of track restricted in a small area as a stopped object.

The grammar of the tracking description of blobs behaviour includes apparition (new target), split, merge, disappearance, stopped, unattended object, entering a zone, exiting a zone. For apparition, there is not “confirmTarget” as in [17] because the filter “*smalltrack*” removes tracks that last for less than a fixed duration.

At the tracking description output, the tracking filtering is performed. It is as necessary as the other filters of the vision system. As usual the filter is used to remove the noise. At this level of processing, it can use temporal consistency. We describe below

some types of filters that can be used in chain. Because the tracking description is a construction built piece by piece during the progression of the video sequence it can process on-line or off-line.

“*smalltrack*”: it detects and removes tracks that last for less than a fixed duration. The duration of the track is the delay between apparition and disappearance. A similar filter is used in [18]. This kind of noise comes after the segmentation has detected noise in the image as blob. One can see on Fig 11 the object labelled 3 at t3 has been erased by this filter.

“*simplifycurvetrack*”: simplifies tracks by removing samples of blobs that give poor information (i.e. if we delete it, we can interpolate it from other parts of the track). It is done by removing a blob instance if it is near the interpolated track made without taking it in account. It could be seen as a dynamic re-sampling algorithm. Fig 12 shows graphically the difference with and without this filter. Fig 13 shows the output in tracking description.

“*simplifysplitmerge*”: removes one part of a double track stemming from a split and then a merge. This kind of noise comes when the segmentation detects two blobs when in all likelihood there is a unique object. It is done by finding when two blobs result from a split and will merge again in the next frame. This filter also splits tracks that have been merged because of proximity of objects in the image. To do so it matches objects before and after the merge/split with features similarities. Fig 14 and 15 shows the results.

We do not describe here other implemented filters in detail. Another filter removes tracks that start or end in defined region of the image. A further one removes the tracks of phantom objects.

## **6.7. Feedback to segmentation**

At this stage it is possible to make high-level decisions that would be useful for the segmentation stage. One can see in Fig 3 the feedback of these kinds of data. For example it is possible with a heuristic to find so called “phantom objects” and tell the segmentation to integrate the corresponding blob. Typically the segmentation output is a map of blobs. A blob could represent a real object or a phantom, i.e. a blob given by segmentation as object but by mistake. One simple heuristic can detect around 90% of these blobs; it uses the *edge mean square gradient (EMSG)*. When this measure is below a threshold, the blob is considered as phantom. An example is given in Fig 16.

### Definition of *EMSG*:

We define the *edge* of a blob as the set of pixels of the blob which have at least one neighbour not included in the blob (in connectivity 4). The *length* of the edge is defined as the number of pixels of the edge. We define the inner edge and the outer edge as the pixels next to edge pixels inside and outside the blob respectively. For each pixel in the edge, we find a pixel in the inner edge and a pixel in the outer edge. The difference of value of the inner and outer pixel is the projection of the gradient of value along and orthogonal axis of locally edge front. We assume that, in the background the gradient is typically low, but orthogonally to the edge of an object it is high, except if the object looks like the locally background. In this scope we use RGB images and define the colour distance of two pixels P and P' as *SDISTANCE* (5).

$$SDISTANCE(P, P') = (P'r - Pr)^2 + (P'g - Pg)^2 + (P'b - Pb) \quad (5)$$

We define the *EMSG* (6) as the quadratic sum of distance of inner and outer pixel of the edge divided by the length and the range of values.

$$EMSG = \frac{\sum_{p \in edge} SDISTANCE(P_{inner}, P_{outer})}{length \cdot 3 range^2} \quad (6)$$

## **6.8. Tracking analysis and event generation**

The tracking analysis is a process that receives the tracking description. It can find predefined patterns like objects entering from a defined zone of the image and exiting by another one, or objects which have exceeded a certain limit speed, or also stopped objects for a minimum time which stem from another mobile object. Fig 17 shows this particular pattern of tracking description. In our application we use this last pattern recognition as “someone is taking an object”. The processing of this module looks into the tracking description graph (e.g. Fig 15) to find the predefined patterns. A similar approach has been described in [17].

## **7. Results and performance**

The system architecture has been validated in several distributed and stand alone applications such as people counting, indoor surveillance, unattended object detection. The research team of the lab is now using it to easily set-up applications. The use of standard libraries ensures good stability. However, the goal of a computer vision system is to address a particular vision

problem with high robustness. That is why we report here the results obtained with a case study, which is the surveillance of areas against unattended object deposits. The detailed application could be found in [19]. We test it on 217 sequences (5 fps 192x144 pixels) taken from 7 sets of sequences with a total length of almost two hours of video with activities at almost all times. Table 1 describes the ground truth. For each sequence, the system should give a Boolean value for unattended object or not. We then compare these two populations of values (positive and negative) from observations (ground truth) against responses from the system. We use the metric described in [20] to characterize the success and failures of the algorithm. Table 2 shows a detection rate of 100% but a false positive rate of 18%. This means that 18% of the negative sequences trigger a false alarm. If we consider that the 217 sequences represent 2 days of activities, thus there is less than one false alarm per hour, and none of the good detections are missed. The first hard requirement for a security system is to achieve a detection rate of nearly 100%. Nevertheless, one of the most frustrating problems for users of such systems is the false activation of alarms [21]. That is why it is usually required to have a few false alarms. However, visual surveillance permits to check by hand the image of the scene directly on screen in order to invalidate the alarm.

## **8. Conclusion**

In this paper, we have proposed an approach for a third-generation video-surveillance platform that can provide the flexibility needed by researchers and that can meet the strong efficiency requirements of industrial applications. We described parts of the image analysis modules focusing on segmentation with background differencing but also on tracking and analysis. We then showed performance evaluations for a study case.

We are currently investigating new vision modules, e.g. better segmentation and tracking methods. Moreover, other extensions and improvements will be made on the global system. Future works will also integrate full evaluation of best parameters and methods by comparing and analysing [20] the output of the global system for a specific application and the ground truth.

**Acknowledgements:** This work has been supported by the Walloon Region in the scope of the ITEA CANDELA Project. We thank anonymous reviewers for their useful comments.

## References

- [1] A.Cavallaro, D. Douchamps, T. Ebrahimi and B. Macq, "Segmenting moving objects : the MODEST video object kernel", Proceedings of Workshop on Image Analysis For Multimedia Interactive Services (WIAMIS-2001), 16-17 May 2001.
- [2] F. Cupillard, F. Brémond and M. Thonnat, "Tracking groups of people for video surveillance", Proceedings of the 2nd European Workshop on Advanced Video-Based Surveillance Systems, Kingston University, London, Sep 2001.
- [3] T. Shcoepflin, C. Lau, R. Garg, D. Kim and Y. Kim, "A research Environment for Developing and Testing Object Tracking Algorithms", Proceedings of the SPIE, Electronic Imaging 2001, vol. 4310, pp. 667-675, February 2001.
- [4] C. Jaynes, S. Webb, R. Steele and Q. Xiong, "An open development environment for evaluation of video surveillance systems", Proceeding 3<sup>rd</sup> IEEE Int. Workshop on PETS, Copenhagen, pp. 32-39, June 1 2002.
- [5] M. Valera and S.A. Velastin, "An Approach for Designing a Real-Time Intelligent Distributed Surveillance", First Symposium on Intelligent Distributed Surveillance Systems, IEE, London, pp. 6/1-6/5, 26 February 2003.
- [6] B. Georis, X. Desurmont, D. Demaret, S. Redureau, J.F. Delaigle, B. Macq, "IP-Distributed Computer-aided Video-Surveillance System", First Symposium on Intelligent Distributed Surveillance Systems, IEE, London, pp. 18/1-18/5, 26 February 2003.
- [7] <http://ffmpeg.sourceforge.net/>
- [8] P. Nunes and F. M. B. Pereira, "Scene level rate control algorithm for MPEG-4 video coding", Visual Communications and Image Processing 2001, San Jose, Proceedings of SPIE, vol. 4310, pp. 194-205, 2001.
- [9] A. D. Worrall, G. D. Sullivan and K. D. Baker, "A simple, intuitive camera calibration tool for natural images", Proceedings of 5th British Machine Vision Conference, University of York, York, pp. 781-790, 13-16 September 1994.
- [10] <http://sourceforge.net/projects/opencvlibrary/>
- [11] J. Shen and S. Castan, "An Optimal Linea Operator for Step Edge Detection", CVGIP, vol. 54, pp. 112-133, (1992).
- [12] A. Elgammal, R. Duraiswami, D. Harwood and L.S. Davis, "Background and Foreground Modeling Using Nonparametric Kernel Density Estimation", Proceedings of the IEEE, vol. 90, No. 7, pp. 1151- 1163, July 2002.
- [13] C. Stauffer, W.E.L. Grimson, "Adaptive Background mixture models for real-time tracking", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, Fort Collins, Colorado, pp. 2246-2252, June 1999.
- [14] T. H. Chalidabhongse, K. Kim, D. Harwood and L. Davis, "A Perturbation Method for Evaluating Background Subtraction Algorithms", Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS 2003), Nice, France, pp. 10-116, Oct. 11-12, 2003.
- [15] R. Cucchiara, C. Grana, A. Prati, R. Vezzani, "Using computer vision techniques for dangerous situation detection in domotics applications", Second Symposium on Intelligent Distributed Surveillance Systems, IEE, London, pp. 1-5, February 2004.
- [16] I. J. Cox and S.L. Hingorani, "An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the purpose of Visual Tracking", IEEE Transactions on Pattern Analysis and Machine intelligence, vol. 18, Issue 2, pp. 138-150, February 1996.
- [17] J.H. Piater, S. Richetto, and J. L. Crowley, "Event-based Activity Analysis in Live Video using a Generic Object Tracker", Proceeding 3<sup>rd</sup> IEEE Int. Workshop on PETS, Copenhagen, pp. 1-8, June 1 2002.
- [18] A.E.C. Pece, "From Cluster Tracking to People Counting", Institute of Computer Science University of Copenhagen, Proceedings 3<sup>rd</sup> IEEE Int. Workshop on PETS, pp. 9-17, Copenhagen, June 1 2002.
- [19] X. Desurmont, A. Bastide, J.F. Delaigle, B. Macq, "A Seamless Modular Approach For Real-Time Video Analysis For Surveillance", Proceedings of 5th International Workshop on Image Analysis for Multimedia Interactive Services, Lisboa, Portugal, April 21-23, 2004.
- [20] T. Ellis, "Performance Metrics and Methods for Tracking in Surveillance", Proceeding 3<sup>rd</sup> IEEE Int. Workshop on PETS, pp. 26-31, Copenhagen, June 1 2002.
- [21] "Alarming false alarms", CCTV TODAY, pp. 18, July/August 2003.



Fig 1: The modular architecture.

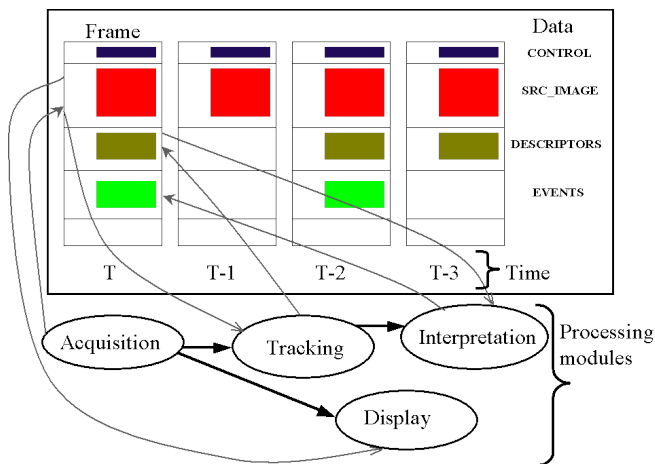


Fig 2: Example of data management.

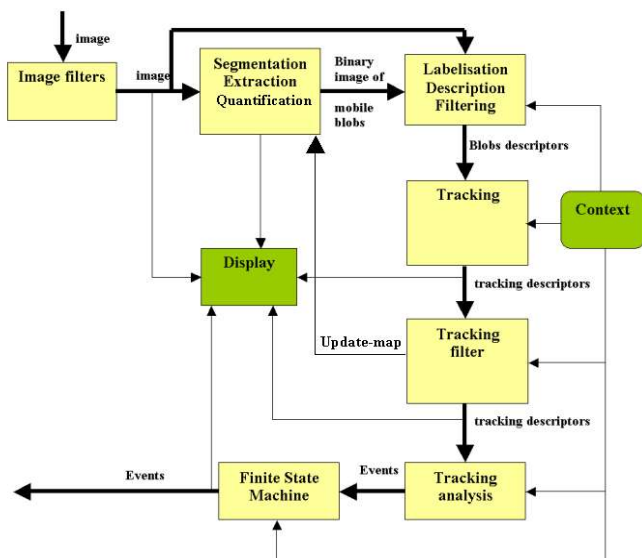


Fig 3: Design of the vision system components.



Fig 4: Interactive tool for manual calibration.

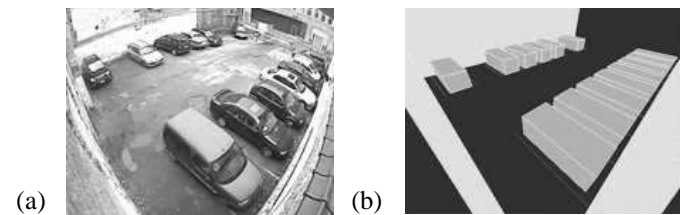


Fig 5: (a) Car park: Initial view from fixed camera, (b) Initial 3D context of the car park.

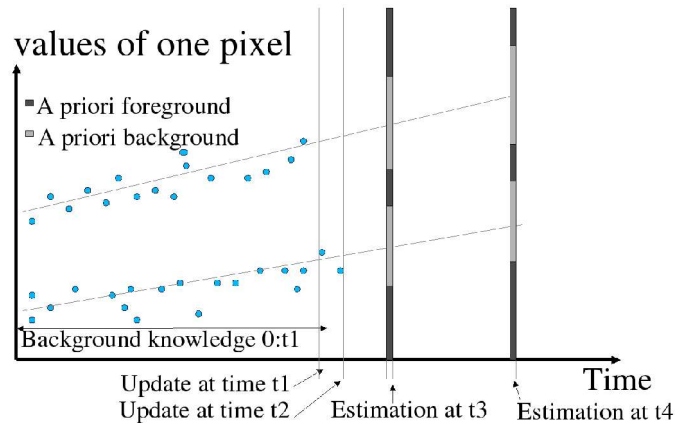


Fig 6: Representation of a value history and background model for one pixel in the image. The dots are the values of luminance of the same pixel in a sequence of images for time 0 to t2. The estimation for time t3 or t4 shows the rule of classification of a new pixel between background and foreground (dark=background, bright=foreground).

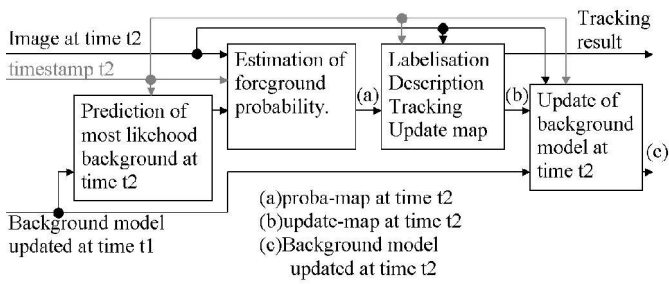


Fig 7: Architecture of the segmentation process within the whole architecture.

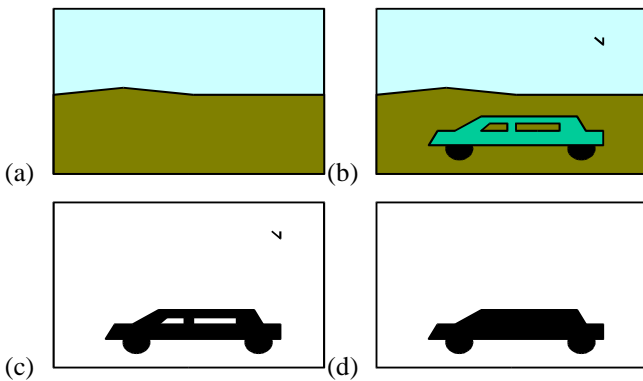


Fig 8: (a) representation of background model at time  $t_1$ , (b) image from the scene at time  $t_2$ , (c) foreground at step 2, (d) foreground at step 3 (small blobs removed and holes also)

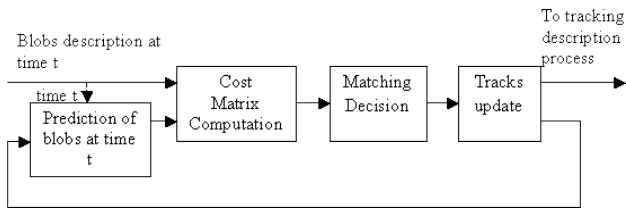


Fig 9: Basic architecture of tracking.

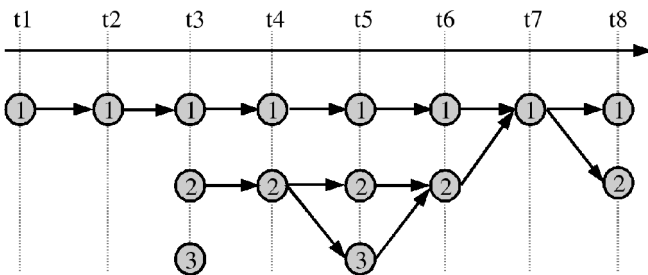


Fig 10: Internal tracking result description.  $t_1$ - $t_8$  are the time-stamps. Circles show objects in the image at time  $t$  and arrows show matchings between objects in different frames.

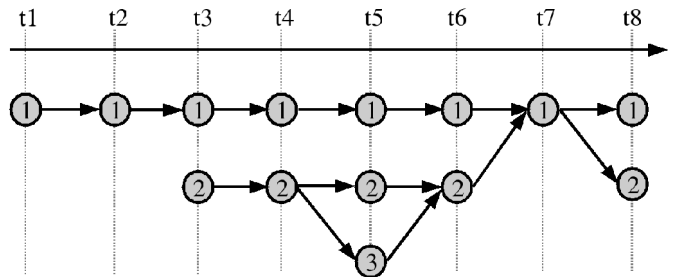


Fig 11: Output of "smalltrack" filter.

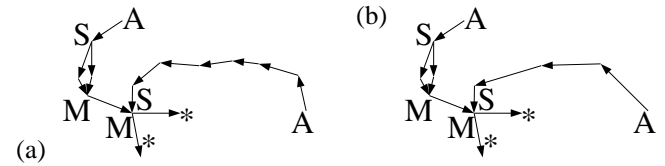


Fig 12: (a) raw tracking description, (b) tracking description filtered by "simplifcurvetrack". Symbols A, D, S, M and \* mean respectively apparition, disappearance, split, merge and "object in current frame".

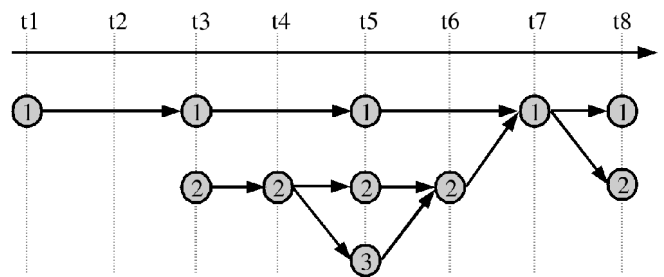


Fig 13: After filter "simplifcurvetrack" a track has been simplified by removing some objects instances.

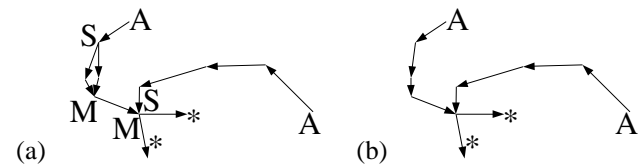


Fig 14: (a) raw tracking description, (b) tracking description filtered by "simplifysplitmerge".

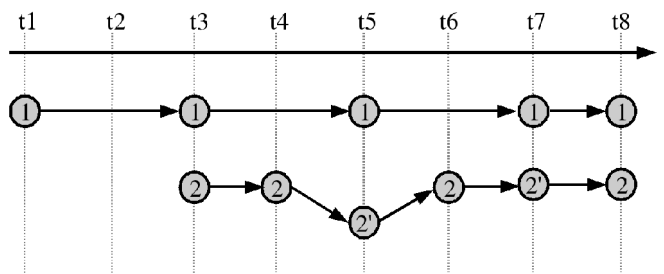


Fig 15: Output tracking after "simplifysplitmerge".



The *EMSG* of the blob (bottom right) is 0.69 and not classified as phantom.

Image with a left object (bottom middle), the *EMSG* of it is 0.06 (it is classified as phantom).

Fig 16: Representation of the *EMSG* of a phantom blob.

Number of sequences	Ground truth	
	Positive	Negative
217	26	191

Table 1: Description of ground truth of sequences.

Nb sequences	Ground truth		System observation			
	Positive	Negative	False positive	False positive rate	True positive	Detection rate
217	26	191	34	18%	26	100%

Table 2: Results of the experimentation.

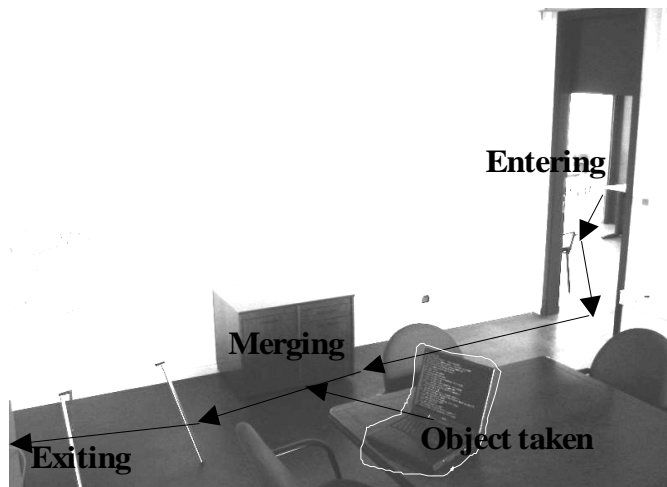


Fig 17: Tracking description pattern for "stolen object".