CLOVIS - A generic framework for general purpose visual surveillance applications

Raffi Enficiaud, Bruno Lienard, Nicolas Allezard, Raphaël Sebbe
Serge Beucher, Xavier Desurmont, Patrick Sayd, JF Delaigle

Abstract

Today's video-surveillance software are often based upon monolitic software running on PCs or embedded systems also called intelligent sensors; but, no real interactions exists between elements of a network dedicated to a video-surveillance scenario. The new framework, named "Clovis" (which stands for 'Composant LOgiciel pour la VIdeo-Surveillance' - Software Component for Video-Surveillance) proposes, for the video-analysis world, a new approach to develop and deploy modular software for and in those sensors on scalable network. Through three sample applications, we present in this article, how the underlined framework can be used to easily develop software on a stand-alone manner or using distributed computing to enhance video-analysis.

1 Introduction

The rapidly growing computational power and communication capabilities of new surveillance sensors on the market, open the way to brand new applications of visual surveillance. The former classical centralized architecture becomes obsolete since the analysis of a scene can be performed in those sensors. Authors of [7] review required needs and skills by these kind of sensors. In this article, we present the "CLOVIS" framework which is a generic platform that allows rapid development of surveillance applications on a stand-alone manner or in a distributed environment through a people counting and a pedestrian detection stand-alone applications and a multi-cameras tracking with no overlapping application. In Section 2, we present framework architecture choices and demonstrate how it can cope with application of very different behaviours. The first application sample is described in Section 3 through the people counting application and Section 4 describes the second stand-alone application. Section 5 presents an application of tracking of people within a network of sensors. We end this article with some concluding remarks and future developments.

2 The architecture

In this section, we present the work achieved toward the design of CLOVIS's framework architecture for hosting video-surveillance software. During its design, we kept in mind that third party surveillance algorithms can be easily integrateted as modules by using the framework APIs. In other words, some services, such as network capabilities, must be available for the application developer to allow construction of various kinds of solutions. Figure 1 shows a typical network of sensors that must be realisable with the CLOVIS platform. Thus the whole system proposes a way to assemble a lot of processing modules in a runtime environment capable of networking interactions.

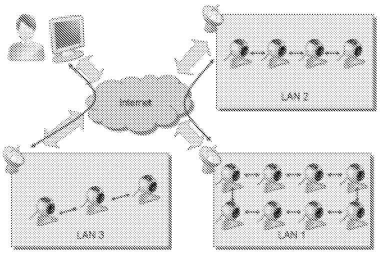


Figure 1: Wide area surveillance issue

2.1 Software issue

CLOVIS provides an Interface pattern oriented *glue* library which provides services for a third party integrator. Used as it, an application developer can use available vision algorithms to create a solution, and if their features are not efficient, he can provides his own solution by providing his own implementation for a given Interface such as processing call, image conversion and so once. Internally, as other development frameworks, CLOVIS also provides event oriented mechanisms that allow catching messages coming from other modules in the same process. When messages is received from another process, events are transparently handled by the communication layer of the framework, as classical third-party architecture.

Once all required modules of surveillance application are selected and implemented, a binary library (compiled application) is produced and deployed on an intelligent sensor running a *plugin host* system which is able to dynamically load (or unload) a surveillance application as shown in 2. Moreover the *host* system is able to send to this library grabbed frames and network events (which are handled in the surveillance library through the framework APIs).

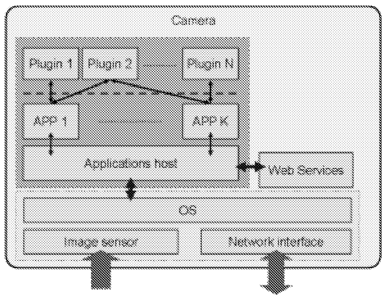


Figure 2: Insight view of the embedded application part

The actual implementation of APIs already provides a comfortable list of image processing functions such as fast background subtraction module based on [18] widely used for tracking purposes.

2.2 Network communications issue

A key issue in the network is the possible amount of exchanged data between network's entities. Since video streaming is a well known problem in this kind of application, we prefer to focus on analysis information exchanged between sensors across the network. Video analysis produces a lot of extracted and interpreted data from the image and often formatted in plain text, called meta-language, such as the nature, the location, the size, ... As today's representation of meta-language is the *eXtensible Markup Language* (XML), we have chosen this formalism to represent descriptions of objects extracted from the scene analysis as shown in 3. Commonly used in IT world for marshalling (serialization of binary object for transportation), we can easily couple our descriptions with a dedicated transport protocol to exchange information between sensors.

The CLOVIS framework is constructed to allow the marshalling of events containing analysis descriptions and send these to other sensors in the network using a *Remote Procedure Call* (RPC) like system. As transport protocol, since a HTTP server is often available on sensors, we have chosen *Simple Object Access Protocol* (SOAP), the protocol part of the Web-Service third party architecture which is often used in distributed computing. Thus, to use this feature, the application developer can embed his analysis description in a CLOVIS event (he can also embed a binary description since *base64* encoding is also supported) and "stream" data

```
<object>
     <name>object 1</name>
     <velocity>
         <vector>
            <c>3.0</c>
            <c>45.57521</c>
         </vector>
     </velocity>
</object>
```

Figure 3: XML description of objects

to other sensors such as he sent data to other modules in the same executable. The gSOAP implementation is used due to his highest performance as explained in [10].

Another issue of network implementation is the way to know the topology of the network, in other words, how the runtime environment embedded in the sensor can know where it must send messages to other sensors. Of course, many systems exist to discover the network's topology, but to handle correctly events between detections we also need more information like geographical location of the sensor and so on (see 4). In our development we are based on a augmented topology server which has a well-known address. Once a newly active sensor registers itself to the server, it informs adjacent sensors that they can be triggered by this new sensor, for instance when a tracked object left its point of view to enter in the area of another sensor.

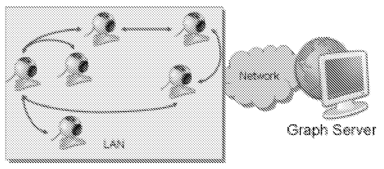


Figure 4: The topology server

In the registering procedure, a scheme is sent to the registered module and defines which kind of information can be sent through the network to another registered module. By this mechanism we can avoid overload of the network with unwanted information and increase the overall performance of the system.

Another important issue pointed out by [7] concerns the security (identity, encryption...) of the transmitted information. Since our architecture is based upon 3rd-Party architectures such as Web-Service, we can take benefit from developments of these technologies. Indeed, since SOAP is over HTTP, the usage of SSL for instance is very easy and in the same manner, all other web-oriented security technologies can be easily used.

3 Counting people

The goal of the counting method is to count people crossing a region, maintaining two counters, one for each direction, positive and negative. The positive direction is provided as an input parameter, as are a number of detection lines. Other parameters of the methods are the expected mean size of people along detection lines, a sensibility threshold, as well as other timing and luminance thresholds.

The main concept of the method is to detect moving people on a set of detection lines, once it is possible to group a detection on each line (depending on direction), the algorithm consider that a person can be counted.

3.1 The algorithm

Detection Lines

Detection of people is done individually on each detection line. A number of 1D processing step and one 2D processing step (step 5) are performed in the following order:

1. Background estimation using a recursive filtering or by using Gaussian mixture modeling
2. Automatic thresholding of the difference image using Otsu method [12]. Then filtering the thresholded image by use of mathematical morphology operators (hole filling and area filtering according to a given people size)
3. Shadow removal, following [13]: the shadow blob are filled and combined with a threshold on the standard deviation of the difference images pixels
4. Split/merge of blobs according to the size. Prior knowledge of expected people size is used to split the blobs along the detection lines
5. Evaluate the speed of candidate
6. Perform tracking and collision detection from previous frames

Shadows are generally correctly evaluated, except for the case of people having a texture similar to the background but constantly darker (which is quite rare). To perform step 4, we do not consider occlusion at time, although a more complete method is currently being developed, accounting for camera perspective and occlusions.

In step 5, the system needs the walking direction to increment the corresponding counter, positive or negative. The speed of the blobs, that we name candidates, is computed using a block-matching algorithm on 8x8 blocks on a subsampled image, and only in the neighborhood of the detection lines. The reason of subsampling is to capture details that are specific to the walking persons into the 8x8 window; the factor thus depends on people size in the original image. Mean candidate speed is evaluated by taking the mean of the speeds provided by the block-matching algorithm in the blob region.

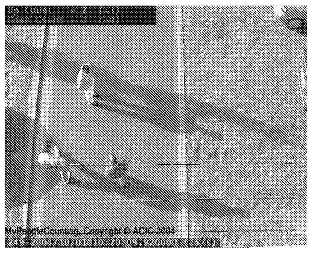
Finally, tracking is performed in step 6, where candidates from successive frames are matched against each other. This permits to differentiate between distinct people. This is accomplished by comparing the candidates of the current frame to the ones of previous frames, in terms of spatial distance. New candidates replace old ones if the corresponding spatial region stays inactive for a given duration (we generally use a value between $[0.2, 1]$ s.).

Combining Multiple Detection Lines

Using a single counting line is generally too much sensitive, and over counts people. To enhance the robustness of the detection, the system use detections from multiple counting lines. The scheme for combining that information is to consider that a person going in one direction should cross the lines in a specific order, compatible to the estimated speed. This is illustrated on figure 5.

N lines are used, and that correspondence (speed / cross time) is searched in their sets of candidates. Additional criteria, such as compatibility between speed and covered distance across lines, could also be used.

In practice, using 3 lines leads to more robust results than when using a single one. Using more than 3 lines does not improve the results quantitatively, although the processing time is directly proportional to the number of lines.


(a) Ex 1

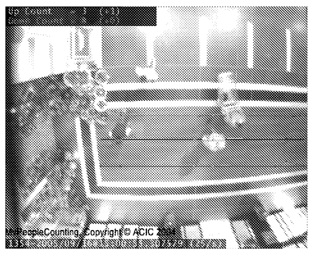

(b) Ex 2

Figure 5: View of the counting configuration

Integration within "CLOVIS"

The algorithm is developed in C++ using Multitel MVision2 libraries which are integrated in the CLOVIS APIs and bundled in one dynamic library as explained in [5] . The CLOVIS development methodology is used to assemble all required modules and the runtime environment is used to execute the application. Since both source code of Multitel's MVision2 and Clovis are "cross-compilable", it is possible to plug directly the library on the Clovis runtime platform and execute the application on both Linux or Windows OSs.

3.2 Constraints and evaluation

As said before, an estimate size of people is required to be able to split blobs on detection line, it means that, in this case, camera orientation must be chosen to minimize occlusion possibilities. Lines setup must be also chosen to avoid incomplete paths between lines, in other words, they must be selected to avoid that a person can cross a first line and avoid one of others. Typically, the ideal location of camera is an overhead camera and lines must be selected between two walls. An evaluation was made in some cases (shopping center entrance and shopping center corridor). The mean error was estimated in these test at 8% for fine tuned algorithm (adapted to the context), including false detections and non detections.

4 Detecting pedestrians

The person detection is an indispensable function in a visual surveillance software library. This is a very challenging issue since the detector must accommodate the wide variety of human appearances, complex backgrounds, possible occlusions, multiple scales. Moreover, a highly desirable feature is the algorithm capability to robustly detect people independently from their motion. The CLOVISlibrary proposes a detection module which enables to detect persons in static images even if they are immobile. It is based on shape recognition by silhouette statistical modeling.

4.1 Descriptors of human shapes

The human shape is captured by a set of local descriptors based on histograms of the gradient orientation, as illustrated in figure 6. These descriptors have already proven to be efficient for shape recognition tasks such as hand gesture recognition [8] and more recently for human detection [4] because of their stability. They are in particular less sensitive to lighting conditions than intensity or color-based methods. To increase their performance, the gradient orientation histograms can also be weighted by the local gradient magnitude.

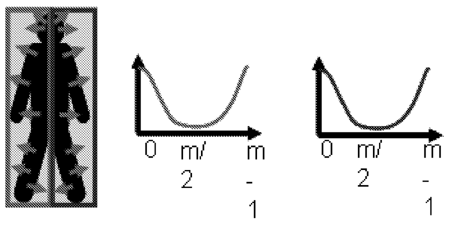


Figure 6: Histograms of oriented gradients on two sub-images

4.2 Learning stage

The descriptors are used as classifiers discriminating human shapes from other objects. They are obtained by training with a learning machine composed of two classes of objects ("person" and "non-person") and is fed by several positive samples (persons images) and some negative ones extracted from various background... A decision stump is associated to each histogram component and classifies in a very simple manner with one split. To determine the spatial support and the histogram components of the most discriminative classifiers, we used the Adaboost learning algorithm which enables to build a strong classifier from a set of weak classifiers -the decision stumps in our case - by focusing on misclassified samples at each round of the boosting process. The performance of Adaboost for searching out a small number of relevant features from a large set of possible ones has been shown in various works [9] [14] and the algorithm has been successfully applied to object recognition problems such as face detection [17].

4.3 Detection strategy

During the detection phases, the analyze window screens the image and each position is classified person or non-person. Several window sizes are used to take into account potential scale variation. To speed up this process, a cascade of base classifiers is implemented . The candidate sub-images pass through a pipeline of filters and only the images passing all these filters which are classified "person". The key idea of a cascaded classifier is to progressively reject sub-images at each stage of the cascade (figure 7). Thus only a small number of feature evaluations are required on average.

Viola and Jones [17], [16] proved that this approach speeds up drastically the detection without decreasing the level of performances. The detection rate of the cascade is given by the product of results of each stage. Each stage is trained to perform 99.9% of positive detection with only 50% of false alarm, which ensure for a cascade with 30 classifiers the following performance:

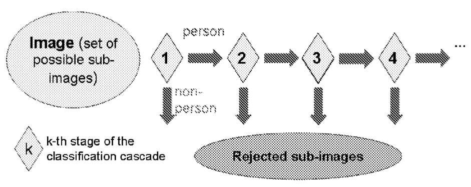


Image (set of possible sub-images)
person
1
2
3
4
...
non-person
k
k-th stage of the classification cascade
Rejected sub-images

Figure 7: Cascade of classifiers

- Rate of positive detection : $0.999^{30} = 97\%$
- Rate of false alarm : $0.5^{30} = 1e - 9$

4.4 Detection results

The learning database is composed of nearly 3000 positive examples. For each stage, we randomly draw 10000 negative examples among the previous stage false positive errors. The resulting cascade contains 16 stages. The false positive rate after training is 3.10e-9 say 6 errors on 2,000,142,144 sub images tested. The figure 8 shows the false positive rate during the training stage. We can see that more than 98% of the sub images are rejected after only 3 stages.

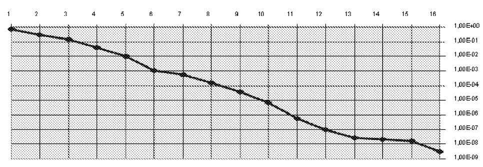
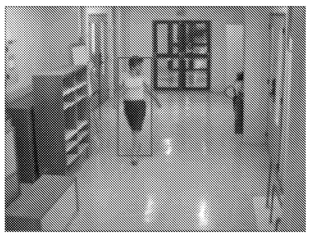
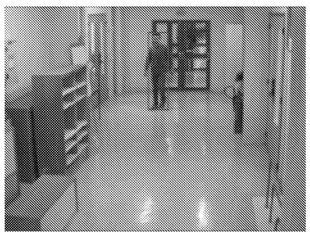


Figure 8: False positive rate during training (stage on abscissa, rate on ordinate)

Then, the detection method was tested on a new video sequence, which was not used during the training stage (fig. 9). This sequence was acquired in similar conditions (location and lighting) than some training sequences (half of the training sequences comes from various images collected on the web). This sequence contains 487 images, 2 persons are present in this scene, where there are seen 399 times. We obtain 14 false detections on 81,159,524 scanned positions (the false positive rate is 1.72.10e-7). The detection rate is 94.2% (376 detections on 399). The pedestrians scale varies from 100 to 370 pixels. The obtained detector performance is comparable to the most recent results in human detection [4] and could be improved by coupling the detection to tracking techniques.



(a) Ex 1



(b) Ex 2

Figure 9: Result of the classifier

5 Tracking using multiple cameras with non-overlapping views

Wide area visual surveillance rises the problem of a distributed intelligent network of sensors. “intelligent” here is considered as being the communication skills of relevant information between some involved sensors. One of the direct application of this kind of network is certainly the tracking of people using multiple cameras, with or without overlapping views. In the sequel, we place ourself in the second category of applications, and restrict the framework to stationary cameras.

Several points of different nature should be taken into account in order to resolve this application. Firsts things first, moving objects should be detected: this is achieved through a background subtraction techniques of our owns described in 5.1.1. For each of the identified moving targets, a relevant signature should be extracted and our approach will be explained in 5.1.2. Finally, for identification issue, the signatures should be sent across the network, and the method will be explained in 5.1.3.

5.1 Design

5.1.1 Intra-scene detection and tracking

One of the widest technique for detecting moving objects is background subtraction. Models for learning the background may be extremely simple, such as pixel-wise gaussian models, or more robust to cope with non-stationary backgrounds (illumination changes, non-rigid object subject to wind, etc.) relying on *kernel density estimation* techniques [6]. However, these work may be computationnaly expensive and are mainly used for segmentation tasks. Besides the convergence and classification problems, work has been achieved to cope with the initialization problem when foreground objects are presents during the training/initialization of the background [2].

Concerning the background initialization issue, and following [2], the aim is to improve the convergence of the background model by avoiding the use of discovered moving parts. To achieve this, we convolve the frame differences by a gaussian kernel in order to have closed boundaries, the size of the kernel is part of the configuration and is scene dependent. We then fill the holes and threshold the result: we obtain a very simple motion detector for a correct initialization.

More robust models of the background cope with non-stationarity by means of learning techniques. Here we feed back the background with the false alarms zones issued from the tracker: an alarm is considered as false when it is completely still during a certain amount of time, say 2 secs. A false alarm probability map is constructed along with the background, and is used when classifying the pixels of a new incoming frame. We obtain a simple background model technique without the use of expensive models and thus suitable for embedding.

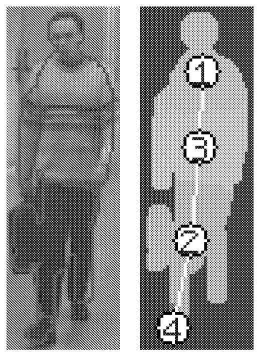
5.1.2 Signatures extraction

Once the blobs were detected, accurate identification across the network should be achieved. In the particular case of distributed computing, the identification should be both compact and robust to change of views and sensor response. One of the most used technique considers the color content of the detected object. This is the approach taken in [3], where three different color space are investigated to build up color histograms. According to the authors, HLS and Munsell color space provide comparable results for inter-scene matching.

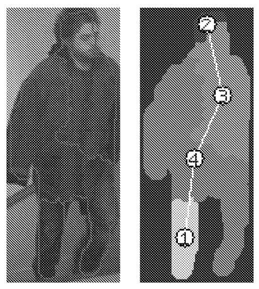
Several histogram metrics does exist and we use the well-known χ^2 distance for histogram comparison [15]. Our experiments show that a bi-dimensional histogram on hue and saturation channels, using HLS with L_1 norm [1], with around 15 bins for each channel, led to results that are correct even under partial occlusions.

In order to increase the accuracy, we use a new method derived from this one: we further segment the blob detected into N regions - N being small - using a hierarchical fusion of region as explained in [11]. First, the watershed transforms leads to an adjacency graph, on which the minimum spanning tree is computed. A new graph is built by merging the closest nodes and in order to have N nodes.

Each nodes of the newly obtained graph corresponds to a particular region. Results of blob segmentation are shown in figure 10. We assign to the nodes the bi-dimensional histogram as explained previously, computed over the underlying region. When the graph presents cycles, we break the weaker connection in term of χ^2 distance; on the contrary, when disjunction occur (for instance the trunk and the feet of a body), we merge the similar nodes by merging the corresponding histograms. A blob is thus explained by a *chain* which nodes are valuated by a bi-dimensional histogram.



(a) Ex 1. (b) Adjcacency graph of (a)



(c) Ex 2. (d) Adjcacency graph of (c)

Figure 10: Segmentation of blobs into 4 regions

5.1.3 Identification across the network

Two major issues arise to cope with the identification problem over the network. The former is rather an engineering issue, and concerns the representation of the data. The latter concerns the metrics used for comparing the blobs to achieve the a consistent labeling.

For the first point, we take real advantage of the CLOVIS framework and it capabilities of marshaling complex and structured data: the chain representation of the detected blobs are transmitted over the network, without the use of any specific development.

In order to take advantage of the camera network topology, lines representing entering/exiting regions in the scene are bound to the edge of the topology graph. When a blob disappears from a camera, it first crosses one of these lines, and its signature is sent to the correct adjacent camera. On the other part, when a camera receives an upcoming blob event, the blob's signature is pooled and compared to each newly detected blobs crossing the line specific to the camera that sent the event. The comparison uses a chain edit distance technique, and we choose the maximum a posteriori over the score extracted from the first frames after the blob appearance and within a time window starting from the upcoming event.

6 Conclusion and perspectives

The three applications have shown that development of specific video-surveillance applications can be done by partners in parallel without inferring others. The CLOVIS framework allow each third-party integrator to develop his own specific modules and exploit these in the same runtime environment. Moreover, basic image processing operators (acquisition, low-pass filter, morphological filters) can be used directly from the built-in image processing library and modules sequence execution is performed thanks to the *plugin host* feature of the platform.

Video-analysis applications developed in the scope of the project may be enhanced in the future such as the blob signature, metrics and network layer part. Another planned improvement of the platform is the development of a complete user interface for assemblage of modules, for the deployment on a network of sensors and for the management of the system.

References

[1] J. Angulo-López. *Morphologie mathématique et indexation couleur. Application à la microscopie en biomédecine*. PhD thesis, Centre de morphologie mathématique, École des mines de Paris, 2003.

[2] A. Bevilacqua. A novel background initialization method in visual surveillance. In *Proceedings of IAPR Workshop on Machine Vision Applications (MVA 2002)*, pages 614–617, December 2002.

[3] R. Bowden and P. KaewTraKulPong. Towards automated wide area visual surveillance: tracking objects between spatially-separated, uncalibrated views. In *IEE Proceedings - Vision, Image and Signal Processing*, volume 152, pages 213 – 223, April 2005.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In C. Schmid, S. Soatto, and C. Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[5] X. Desurmont, B. Lienard, J. Meessen, and J.-F. Delaigle. Real-time optimizations for smart network camera. In *Proceedings of SPIE - Real-Time Imaging IX*, January 2005.

[6] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using non-parametric kernel density estimation for visual surveillance. In *Proceedings of the IEEE*, July 2002.

[7] M. V. Espina and S. A. Velastin. Intelligent distributed surveillance systems: a review. In *IEE Proceedings - Vision, Image, and Signal Processing*, volume 152, pages 192–204, April 2005.

[8] W. Freeman and M. Roth. Orientation histogram for hand gesture recognition. In *Int'l Workshop on Automatic Face- and Gesture-Recognition*, 1995.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Dept. of Statistics, Stanford University, 1998.

[10] M. Govindaraju, A. Slominski, K. Chiu, P. Liu, R. van Engelan, and M. J. Lewis. Toward characterizing the performance of soap toolkits. In *Proceedings of 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, November 2004.

[11] F. Meyer. Minimal spanning forests for morphological segmentation. In J. Serra and P. Soille, editors, *Mathematical Morphology and its applications to signal processing (Proceedings ISMM'94)*, pages 13–14, Fontainebleau, France, September 1994. Kluwer Academic Publishers.

[12] N. Otsu. A threshold selection method from gray level histograms. *IEEE Trans. Systems, Man and Cybernetics*, 9:62–66, threshold selection 1979.

[13] E. Salvador, A. Cavallaro, and T. Ebrahimi. Cast shadow segmentation using invariant color features. *Computer Vision and Image Understanding*, 95(2):238–259, August 2004.

[14] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[15] B. Schiele. *Reconnaissance d'objets utilisant des histogrammes multidimensionnels de champs réceptifs*. PhD thesis, Institut National Polytechnique de Grenoble, 1997.

[16] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003.

[17] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR (1)*, pages 511–518. IEEE Computer Society, 2001.

[18] P. P. Wayne and J. Schoonees. Understanding background mixture models for foreground segmentation. In *Image and Vision Computing New Zealand (IVCNZ 2002)*, pages 267–271, November 2002.