UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

PEOPLE DETECTION ALGORITHMS BASED ON APPEARANCE AND MOTION INFORMATION

Alvaro García Martín
Supervisor: José María Martínez Sánchez

-TRABAJO FIN DE MASTER-

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
November 2009

PEOPLE DETECTION ALGORITHMS BASED ON APPEARANCE AND MOTION INFORMATION

Alvaro García Martín

Supervisor: José María Martínez Sánchez

email: {Alvaro.Garcia, JoseM.Martinez}@uam.es

Video Processing and Understanding Lab

Departamento de Ingeniería Informática

Escuela Politécnica Superior

Universidad Autónoma de Madrid

November 2009

Abstract

.The work presented in this document is focused on the improvement of the people detection task in visual analysis in video surveillance sequences. We have designed and implemented a people detection algorithm based on appearance and motion as discriminative information. Firstly, we have undertaken a study on the state of the art in people detection algorithms targeted to video surveillance scenarios and we have made a classification of the different algorithms studied. Within this classification, algorithms based on appearance use to obtain better results than motion based algorithms, and therefore we have designed and implement an algorithm based on appearance but adding robustness to the detection with motion information. As main result a new people detector have been implemented and integrated into the VPULab system: this new algorithm obtains more reliable results and less computational cost than the previous algorithm implemented by VPULab. Additionally, in order to provide a good performance evaluation of the proposed framework and a comparison with previous work: the design and annotation of a people video/image dataset have been done.

This work is supported by Cátedra Infoglobal-UAM para “Nuevas tecnologías de vídeo aplicadas a la seguridad”, the Spanish Government (TEC2007- 65400 SemanticVideo) and the Comunidad de Madrid (S-050/TIC-0223 - ProMultiDis-CM)

Acknowledgements

First I would like to thank my supervisor, Prof. Dr. José María Martínez Sánchez, for their support and advice during this work.

I also wish to thank all members of Video Processing & Understanding Lab for their support and friendship.

Finally, I would like to thank my family and friends for always being there supporting me.

Álvaro García Martín
November 2009

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the last years signal processing has been in constant evolution. In particular, it has been making big efforts and progress in digital image and video processing because of their utility in the information society that we are living in.

Considering the huge demand existing in the area of security systems, one of the biggest research lines is video surveillance. The need for providing security to people and their properties in a crowded world explains the huge development and expansion of video surveillance systems nowadays. Within the digital image and video processing research area there exists a rich variety of algorithms of motion detection, object detection, event detection, etc, which are being used in security. Automatic people detection in video sequences is one group of them. It is actually a complex problem with multiple applications, not only in video surveillance, but also different areas like intelligent systems (robotic), video games, etc.

The complexity of the problem is mainly based on the difficult of modeling a person and his activities because of their huge variability. People present a big variety of physical appearances, poses, movements and interactions between different people and objects. Currently many different systems exist which try to solve this problem. The state of art in people detection and tracking includes several successful solutions working in specific and constrained scenarios. In contrast the state of art in people activity detection is less explored because of its higher complexity and its strong dependency with previous phases (e.g., people detection, object detection and tracking).

1.2 Objectives

The work presented in this document has three main objectives, in first place to identify and study the main people detection algorithms that to date have been proposed in the literature. In second place to implement a people detection algorithm using people appearance and motion as discriminative information. And in third place to improve the results of previous people detector which has been implemented by the Video Processing Understanding Lab (VPULab).

To archive this objective, a new a video processing framework has to be implemented (based on VPULab system). Therefore, the presented work will have the following stages:

- To study the state of the art in people detection algorithms.
- To integrate common techniques for semantic information extraction or object recognition in video sequences.
- To integrate specific people detection techniques in video sequences.
- To design and develop people detection techniques based on people appearance and motion as discriminative information.
- To combine or fuse different people detection techniques.
- To evaluate and compare with previous people detector the quality of the final results using objective techniques, as well as its computational execution cost. Furthermore, this objective includes the design of an appropriate evaluation methodology and a data set to evaluate the techniques proposed.

The final objective of this system is the design and implementation of a framework which allows us to develop our research objectives. By means of video analysis tools, discriminative features between people and objects will be extracted:

- People detection algorithms based on people appearance will be studied because most of the papers related to people detection use appearance information; this is logical because the main features that discriminate objects, vehicles, animals, etc, are the features based on appearance (aspect ratio, color, silhouette, etc), which work better than motion related features.
- People detection algorithms based on motion will be studied. Although the motion information is not as discriminative as appearance information, in multiple scenarios it can be considered complementary information or even essential to obtain the correct people detection.

- The combination or fusion of multiple detection algorithms, adding robustness and effectiveness to the global system.

Finally, the results of this work will be evaluated in order to study the possibility of including the new people detection techniques into the VPULab system. In addition to this, the developed system will be able to be extended, that means, it has to be designed to introduce easily new improvements or algorithms without interfering with global analysis system.

1.3 Document Structure

The structure of the document is as follows:

- Chapter 1. This chapter presents the motivation and the objectives of the work presented.
- Chapter 2. This chapter presents an overview of the literature related to the work presented in this document.
- Chapter 3. This chapter presents the design and implementation of the proposed people detection algorithm.
- Chapter 4. This chapter presents the integration of the new detector in the system implemented in the VPULab.
- Chapter 5. This chapter presents the data set used, the evaluation process and some experimental results. Furthermore, it includes a detailed comparison of the new system results with the previous one.
- Chapter 6. This chapter summarizes the main achievements of the work, discusses the obtained results and provides suggestions for future work.

At the end, three appendixes list further details. They describe the GML AdaBoost Matlab Toolbox (Appendix A), the C++ classes used or created during this work (Appendix B) and the ViPER toolkit used for the evaluation process (Appendix C).

Chapter 2

State Of The Art

This chapter gives an overview of previous work that has been done in the scope of the work presented in this document. In the next sections, we describe briefly a "canonical" People Detection System in section 2.1 and the state of the art in people detection algorithms targeted to video surveillance scenarios in section 2.2.

2.1 People Detection System

Automatic people detection is a complex problem because of the huge variability in people appearance and motion. The people detection task consists of the design and training of a person model based on characteristic parameters (motion, dimensions, silhouette, etc). Then the detection consists of adjusting our person model to the objects' models in the scene. All objects that adjust to our model will be detected/classified as person, whilst all the others won't be detected/classified as person.

The processing stages of a "canonical" automated video analysis system for people detection include: background/foreground extraction, object extraction (object detection), object classification, object tracking, and event or action recognition [7, 8]. As we can see in Figure 2.1 our system comprises of four stages:

- Background/Foreground extraction: Nearly every visual surveillance system starts with motion detection. Motion detection aims at segmenting regions corresponding to moving objects from the rest of the image. The approaches range from simple (e.g., frame difference) to more sophisticated ones which try to estimate a more complex background model (e.g., Mixture of Gaussians, Kernel Density Estimators) [9]. The consecutive stages depend on the background accuracy obtained, that is, the rest of stages have a strong dependency with results obtained in this process: a bad background model could cause false object detections, missing objects or partial object detections.

- Object extraction: After background/foreground segmentation, morphological operations are typically applied to reduce the noise of the resulting image mask and improve the object extraction. Once the region boundaries have been detected, it is often useful to extract regions which are not separated by a boundary. In order to segment or extract objects, a connected component analysis is realized. Only objects extracted or detected in this stage are analyzed in following processes.
- Object tracking: After motion detection and object extraction, surveillance systems generally track moving objects. The aim of an object tracker is to generate the trajectory of an object over time by locating its position in every frame of a video sequence. Tracking over time typically involves matching objects in consecutive frames using features such as points, lines or blobs. Tracking methods could be divided into three major categories: point, kernel or silhouette tracking [10].
- Object classification: Object classification can be considered as a standard pattern recognition issue. This process compares our object classification models and generated object models from an image or sequence (tracking) and makes a final decision base on their similarity.

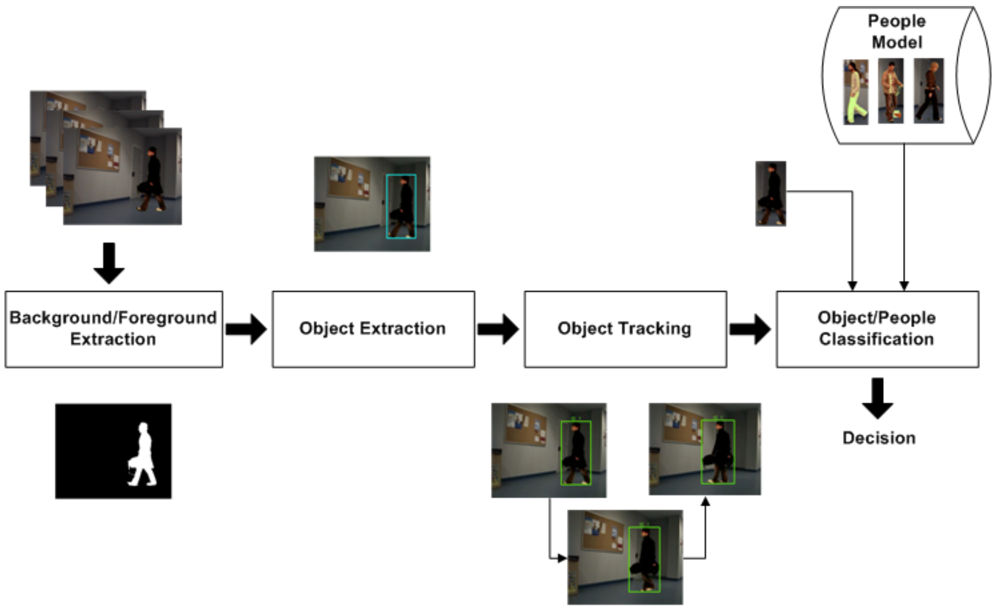

People Model
Background/Foreground Extraction
Object Extraction
Object Tracking
Object/People Classification
Decision

Figure 2.1: "Canonical" Video Analysis System for People Detection

2.2 Classification of People Detection Algorithms

This section describes the classification of people algorithms realized and enumerates the different people detection algorithms studied. During this study we have classified the different approaches to solve the people detection problem in video sequences [1].

Many criteria could be used to classify people detection algorithms; for example, the techniques used (e.g., background/foreground extraction, movement estimation/compensation), the type of models used (e.g., stick figure-based, statistical, movement), the use of 2D or 3D information, the sensor modality (e.g., visible light, infra-red), the sensor multiplicity (monocular, stereo, or multicamera), the sensor placement (centralized vs. distributed), the sensor mobility (stationary vs. moving), etc.. This work follows the second criteria: we have decided to realize the classification based on the features used to model a person (silhouette, height, color, movement pattern, etc), because it is more discriminative to classify the different proposed models. The techniques used to extract the information can vary from one system to another, but the people features are invariant to the techniques used to extract them.

To show our classification we have defined a hierarchical tree structure (see Fig 2.2) in which each branch excludes the others possibilities in order to avoid ambiguous situations. In the following sections, we follow this hierarchy for overviewing the studied state of art.

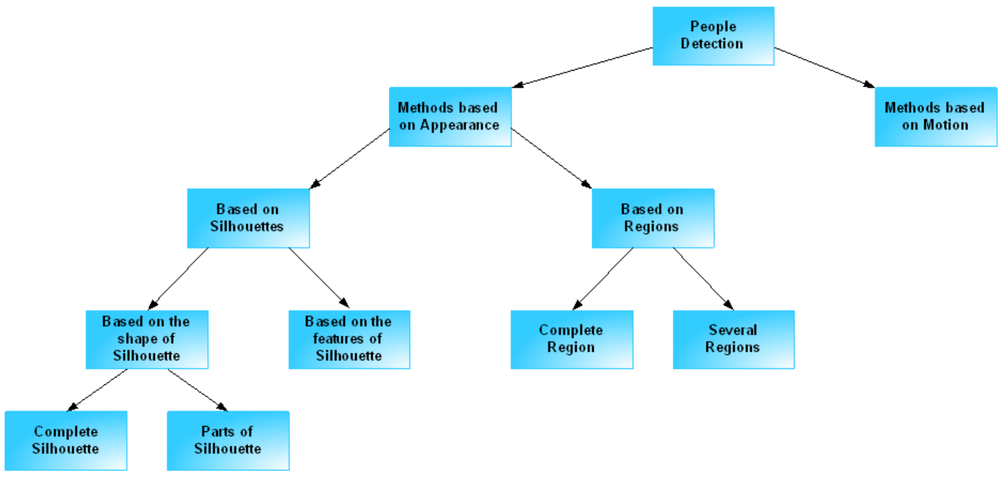


Figure 2.2: People Detection Classification

[1]Any classification system could be perfectly debated because it depends on the discriminative aspects on which its hierarchy is based.

2.2.1 Methods Based on Motion

As shown in Figure 2.2 the first people detection method classification consists of methods based on people appearance or methods based on people movement analysis. Nowadays in the existing literature, most methods are only based on appearance information or they add robustness to the detection with motion information through tracking algorithms. However, human appearance varies due to environmental factors such as light conditions, clothing, contrast, etc., apart from the huge intrinsic people variability such as different heights, widths, poses, etc. For these reasons, there are some approaches which try to avoid these factors and to realize the detection using only motion information.

Inside this classification, [1] proposes a people detection system based on detecting people motion patterns. For each object present in two consecutive images size normalization is performed and its flow pattern is calculated, that consists of dense optical horizontal and vertical flows (see Figure 2.3). The resulting pattern is then classified using a Support Vector Machine (SVM) [11].

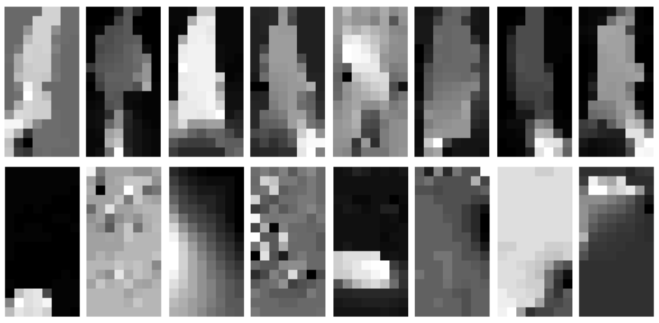


Figure 2.3: Human Motion Patterns and Non Human Motion Patterns (extracted from [1])

Another approach based on motion information [2] proposes an object classification system based on periodic motion analysis. The algorithm segments the motion, tracks objects in the foreground, aligns each object along time and finally computes the object's self-similarity and how it evolves in time. The periodicity of this measure is analyzed using Time-Frequency analysis (see Figure 2.4) and finally object classification is realized (e.g., people, running dogs, vehicles) using periodicity.

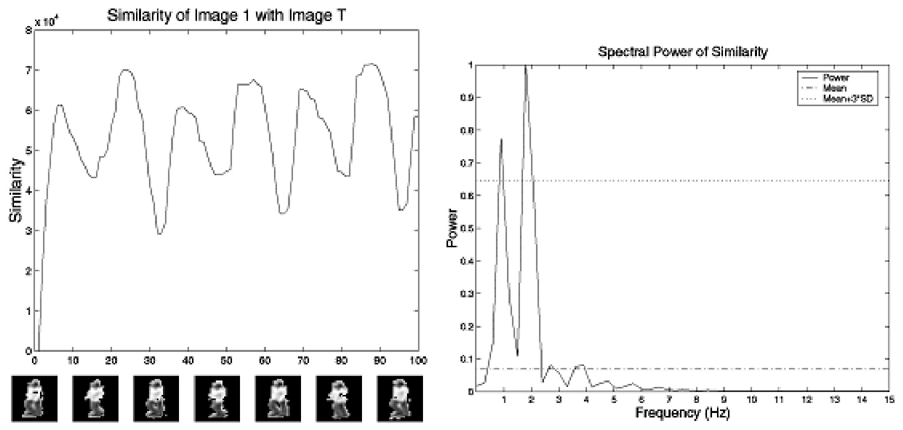


Figure 2.4: Object Self Similarity Time and Object Self Similarity Frequency (extracted from [2])

Methods based on motion use to obtain worse results than methods based on appearance but they are independent of appearance variability. They do not support partial occlusions because in this case we could not extract motion patterns correctly (see Table 2.1). For these reasons they could be considered either complementary information or essential in specific scenarios where methods based on appearance did not work (e.g., bad illumination, small objects).

Algorithm	Feature	Object Classification	Support Partial Occlusions	Complexity
[1]	Flow Pattern	Person/No person	No	Medium
[2]	Periodic Motion	People Running dogs Vehicles	No	Low

Table 2.1: Methods Based on Motion

2.2.2 Methods Based on Appearance

As we have already commented, most of the existing approaches use people appearance information to decide if the detected objects are people or not. Inside this classification we can see two easily differentiate branches: on the one hand algorithms based on people silhouettes, which are based on their contour, and on the other hand algorithms based on the regions that represent the person. This ramification splits the problem and even the approach used in each case in two systems clearly differentiated. On the one hand systems based on silhouettes are focused on extracting objects contours and over this processed image creating or adjusting their classification model. On the other hand systems based on regions do not need to extract contours: they look for regions over the image which could be objects and put into practice their classification model.

2.2.2.1 Methods Based on Silhouettes

The methods based on silhouettes can be classified in those which try to model the silhouette's shape and those which only extract discriminative features from the silhouette (height, aspect ratio, ellipse adjust, elliptical cylinders adjust, etc).

Methods Based on Features of Silhouette [3] proposes a surveillance system which makes use of a depth and gray scale sensor. This method does not use background subtraction, but uses depth information to segment objects. Afterwards, this method tries to fit an elliptical model to each of the detected objects (blobs). To fit an ellipse to the silhouette, it first defines the initial ellipse with the bounding box size of the silhouette and then shrinks it until it is totally inside the silhouette (see Figure 2.5). The final ellipse model is compared with a previous established model.

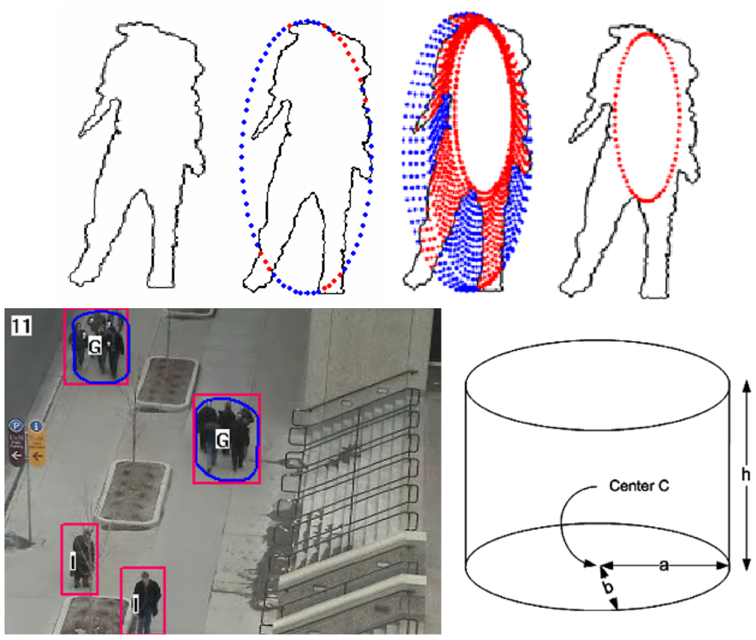


Figure 2.5: Ellipse Fitting and Elliptical Cylinders Fitting (extracted from [3, 4])

[4] considers the problem of counting the number of people in the scene for what people detection should be performed. First background subtraction is realized, then

the projected area on the ground in world coordinates and the velocity are estimated for each foreground blob. Finally each blob is classified as individual person, group or vehicle. In order to decide between individual person and group, the author proposes two methods based on features of the silhouette. The first method uses the projected area occupied and the average density of people and groups which is learned previously. The second method tries to fit an elliptical cylinder to the silhouette (see Figure 2.5). Finally with this information the number of people in each blob is estimated.

[12] proposes a people detection system based on the fusion of three simple independent people detectors. Each detector is separately applied to each detected blob and the final decision is a combination or fusion of those three evidences. Each detector uses features extracted from the silhouette. The first detector is based on the aspect ratio of the blob (ratio between its width and height), the second detector is based on the already explained ellipse fitting method and the third detector is based on the Ghost algorithm proposed in [6], which computes the convex and concave hulls of silhouette, detects the silhouette's vertexes and finally determines the possible positions of body parts.

Systems based on features of the silhouette use to obtain worse results than methods based on shape, but they have lower complexity (people simplified models). They do not need accurate silhouettes and therefore a good approximation is enough, whilst shape models need to obtain accurate silhouettes. And it could be so complicated on complex scenarios: light conditions changes, shadows, motion velocity, etc. For this reason these methods (see Table 2.2) use to be a simple and fast alternative.

Algorithm	Feature	Object Classification	Support Partial Occlusions	Complexity
[3]	Ellipse Fitting	Person/No person	No	Low
[4]	Elliptical Cylinder Fitting	Individual Group Vehicles	No	Low
[12]	Aspect Ratio Ellipse Fitting Convex-Concave Hulls	Person/No person	No	Low

Table 2.2: Methods Based on Features of Silhouette

Methods Based on Silhouette's Shape Methods based on the shape of the silhouette are divided in two branches. On the one hand those which are based on a model of person as a complete silhouette [5, 13, 14], and on the other hand those which are based on a model of person as the union of parts of the same silhouette [6, 15]. Whereas the first approaches are focused on finding the silhouette of objects, the second approaches are focused on finding the different parts of the silhouette (head, arms, legs, etc) and then they try to join them to obtain their person model.

Complete Silhouette [5] defines a people detection system using a trained codebook of people shapes in order to classify between humans and other objects. First a size normalization of the object is realized, then the shape of the object is extracted (see Figure 2.6), and finally the feature vector is coded as the distance from the boundary of the human body to the left side of bounding box. The final decision depends on the minimum distortion between the object signature and codebook.

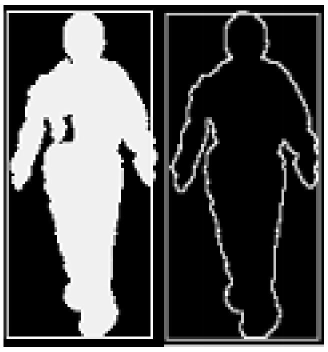


Figure 2.6: Silhouette Extraction (extracted from [5])

[13] presents a system for human detection in uncontrolled camera motion environments. The algorithm uses a log-polar transformation of the image pair to recover translational, rotational and scale misalignments. This algorithm searches for humans in the image by matching its edge features to a database of templates of human silhouettes using the Chamfer distance [16].

[14] proposes a people detection system using a 3D camera. This camera, mounted on a mobile robot, generates a dense 3D point cloud and afterwards a connected component algorithm is applied in order to generate compact objects. Each detected object is modeled as a feature vector consisting of human shape in the form of a row-oriented and column-oriented histogram.

Parts of Silhouette [6] describes an approach to locate body parts of people using silhouettes. The people model consists of six primary body parts (head, two hands, two feet and torso) and ten secondary parts (elbows, knees, shoulders, armpits, hip and upperback). Firstly, the algorithm estimates the human body posture (standing, sitting, crawling-bending or laying) using normalized horizontal and vertical projection histograms and posture models previously trained. Secondly, it computes the convex and concave hulls of the silhouette, and then it detects the silhouette's vertexes and the possible positions of the body parts. Finally, it estimates body parts locations with posture decision and silhouette's vertexes information (see Figure 2.7).

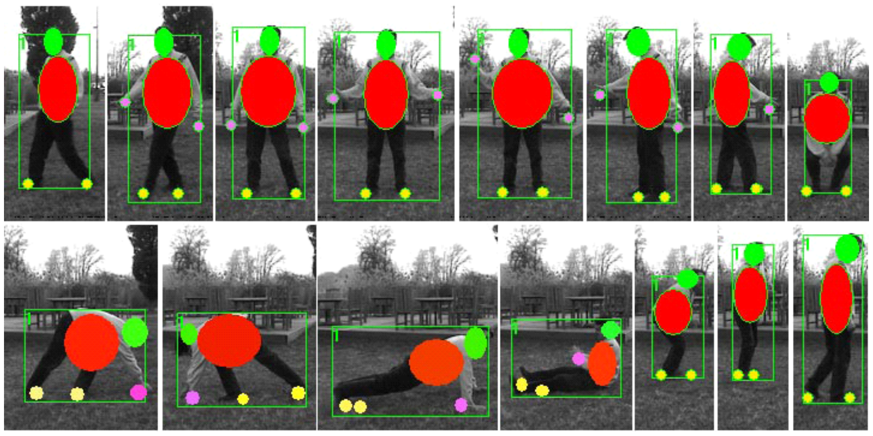


Figure 2.7: Parts of Silhouette (extracted from [6])

[15] proposes a people detection algorithm based on a human model as an assembly of natural body parts. Four independent edge feature detectors are trained (body, head, torso and legs), and afterwards the image is scanned with body parts detectors. Responses of body parts detectors are combined to form a joint likelihood model that includes cases of multiple and possibly inter-occluded humans.

The systems based on the complete silhouette's shape use to obtain good results but they depends only on the obtained silhouette with the corresponding partial occlusions (arms, legs, etc), whilst the systems based on parts of silhouette make the final decision as combination of multiple evidences (different found parts of silhouette) allowing them to support partial occlusions (the absence of any of them) but with a higher computational cost (see Table 2.3).

Algorithm	Feature	Object Classification	Support Partial Occlusions	Complexity
[5]	Silhouette Shape Codified Vector	Person/No person	No	Low
[13]	Silhouette Shape Templates	Person/No person	No	Low
[14]	Silhouette Shape Codified Vector	Person/No person	No	Low
[6]	Body Parts	Person/No person	Yes	Medium
[15]	Body Parts	Person/No person	Yes	Medium

Table 2.3: Methods Based on Silhouette's Shape

2.2.2.2 Methods Based on Regions

As in the previous division between complete silhouette and parts of silhouette, methods based on regions can be divided in those which model the person as one unique region [17, 18] and those which try to detect different regions in order to join them in a person model [19, 20]. This classification has got the same advantages and disadvantages than the previous division. Methods based on the combination of body parts make the final decision by combining multiple evidences so they use to be more reliable than methods based on simpler human models. However they have higher computational cost (see Table 2.4).

Complete Region [17] presents a pedestrian detection system that integrates image intensity information with motion information. The basis of the model is an extension of the rectangle filters from Viola and Jones to the motion domain, this means that the algorithm calculates Viola's filters responses over two consecutive frames of a video sequence instead of over a single frame. Each person is considered as a complete region and the human model is characterized with the responses estimated with each filter.

[18] proposes an extension of the previous algorithm. In this case the algorithm defines seven types of volume filters in the 3D space, instead of using rectangle filter in the 2D space. It extracts these features in a space-time volume and uses a Gaussian kernel SVM as classifier.

Several Regions [19] defines a people detection system based on color segmentation and cloth people detection. Firstly, the image is divided into regions using color-based segmentation; secondly, the body parts localization is initialized with a face detection phase; and finally, a tree structured human body model is built. This model represents the probability distribution of body parts and each leaf corresponds with a body part (hair, face ,arm , hand or leg) or an article of clothing (shirt, pants, etc) as shown in Figure 2.8.

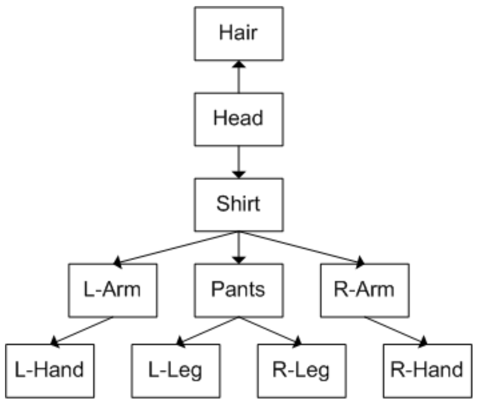


Figure 2.8: Tree Human Model

[20] proposes a people detection method based on human model of three body parts and color information to front and side views. Object segmentation is based on skin color and robust background modeling. Then, the people model is built with skin color probability distribution maps in each body part (face, head and torso).

Algorithm	Feature	Object Classification	Support Partial Occlusions	Complexity
[17]	Haar-like Features	Person/No person	No	Low
[18]	3D Haar-like Features	Person/No person	No	Low
[19]	Color Body Parts	Person/No person	Yes	Medium
[20]	Skin Color Body Parts	Person/No person	Yes	Medium

Table 2.4: Methods Based on Regions

2.2.3 Conclusions

During this section we have commented some advantages and disadvantages of different approaches to solve the people detection problem in video sequences. This subsection sums up some conclusions extracted from the realized study.

As we have already commented, there are few approaches based only on motion information. Their main advantages are that they are independent of appearance variability and use to have low complexity. However they use to have poor results and they do not support partial occlusions. Methods based on motion information could add robustness without adding too much complexity.

The methods based on people simplified models (only a region or shape) use to have lower complexity but they do not support partial occlusions neither pose variations. However, the methods based on more complex people models use to have higher complexity but they support partial occlusions and pose variations. Another advantage is that they made the final decision by combining multiple evidences, so they use to be more reliable than methods based on simpler human models. For these reasons they use to have better results.

Chapter 3

People Detector

3.1 Introduction

In this chapter we describe the design and implementation of the proposed people detection algorithm. As we have already commented, algorithms based on appearance use to obtain better results than motion based algorithms, and therefore we have decided to design and implement an algorithm based on appearance but adding robustness to the detection with motion information. In the next sections, we briefly describe previous people detection work (section 3.2), the proposed people detector (section 3.3) based on appearance, and the enhancement of it making use of motion information (section 3.4).

3.2 Previous Work

As we have already commented, one of the aims of this work consists of trying to improve the results of a previous people detection system implemented within the VPULab. This previous people detector [12] has been classified and briefly explained in section 2.2. This algorithm is a method based on simple features of the objects' silhouette; each feature constitutes an independent people detector:

- Aspect ratio: This first detector is the simplest and it is based on a single feature, the aspect ratio of the blob, which is defined as the quotient between its width and height.
- Ellipse fitting: The second people detector is based on the ellipse fitting method [3]. It makes use of three simple features: the ratio between the number of iterations and the maximum number of iterations allowed in the ellipse fitting

process, the percentage of the ellipse's sampled points that lie outside the silhouette at the end of fitting, and the final ellipse's aspect ratio.

- Convex-Concave Hulls: The third people detector is based on the Ghost algorithm proposed in [6] which approximates the contour of the blob's foreground with a closed polygon corresponding to the contour's convex and concave hulls. Three features are also evaluated for this detector based on that polygon: the number of points of the polygon, the ratio between the amount of convex and concave vertices, and the inverse of the number of vertices that are found to belong to the top of the polygon (head).

In order to generate a measure of evidence from a given people-related feature x, the latter is assumed to approximately follow a normal distribution of mean μ and standard deviation σ. Both parameters are experimentally determined for every defined feature by considering a training set with images of people in usual postures. The evidence of the given feature is then defined as a real value between zero and one.

Each detector is applied to every blob and they provide independent evidences. The final evidence about the analyzed blob being a person is obtained by averaging the evidences provided by the three detectors. The result of the fusion is significantly more efficient and stable than when the detectors are applied separately. Each detector defines a simplified people model (aspect ratio, ellipse or convex-concave hulls), and therefore the proposed algorithm is simple and fast. However the use of so simple people models use to provide worse results and more limitations, such as problems with partial occlusions, pose changes or camera point of view changes, because this kind of models do not support it.

3.3 Proposed People Detector

3.3.1 Introduction

Our people detector is based on the algorithm proposed in [15] but targeted to online video surveillance scenarios. This algorithm defines a more complex people model, and for this reason we expect better results than the algorithm described in previous section 3.2.

Original Algorithm [15] proposes a method for human detection in crowded scenes, but working only with static images (frames). People's silhouette is one of most discriminative features in order to classify objects and people. The main idea consists of identifying characteristic edges of each body part and generating four edge body parts models. Four independent edge feature detectors are trained (body, head, torso and legs), then the image is scanned with body parts detectors. Responses of each part detectors are combined to form a joint likelihood model that includes cases of multiple and possibly inter-occluded humans. This algorithm supports pose or camera point of view changes, but does only work for people with upright standing or walking pose. The main problem lies in correctly modeling the four body parts based on training silhouettes. However the final decision is a combination of four evidences and therefore modeling is not such a critical stage: even though the individual part detectors may have false alarms, they would be solved by the combined detector.

Proposed Algorithm The original algorithm is targeted to static image and scans the complete image, for this reason people models must be complex to classify correctly many negative examples. In addition computation time is not a main objective so then the training is focused in reducing false positives (complex people models), therefore greatly increases the processing time. In order to get a faster algorithm we propose not to scan the complete image, we only process detected objects during previous stages (see Figure 2.1) and we simplify the training process (simpler people models) in two ways: we use a ranking of the best edges of each body part model (see section 3.3.3.4) and the training is not focused in reducing false positive but getting good precision results.

In the next sections, we describe the training image dataset (section 3.3.2), proposed algorithm design (section 3.3.3) and final system design (section 3.3.4).

3.3.2 Image Dataset

The proposed algorithm consists of four edge body parts models. Each model has to be trained with an image collection with people and no people examples and therefore we need a complete image dataset with positive and negative examples `http://www-vpu.ii.uam.es/~agm2/dataset.html`.

Negative Images

Negative images have been chosen from the LabelMe dataset [21]. LabelMe is a database and an online annotation tool that allows the sharing of images and annotations. The online tool provides functionalities such as drawing polygons, querying images, and browsing the database. We have selected images with objects and scenarios without people. Each image has been cropped in small pieces in order to obtain a huge number of different negative images and has been normalized (four different body sizes and gray scaled).

Positive Images

Positive images have been chosen from the INRIA dataset [22]. The INRIA dataset contains images of humans cropped from a varied set of personal photos: the people are usually standing, but appear in any orientation and against a wide variety of background image including crowds. People body blobs have been extracted, normalized (58x24 pixels and gray scaled) and segmented in body parts (see Figure 3.2). We can see some negative and positive examples in Figure 3.1.

Finally our image dataset stores 3542 positive images and more than 40000 negative images.

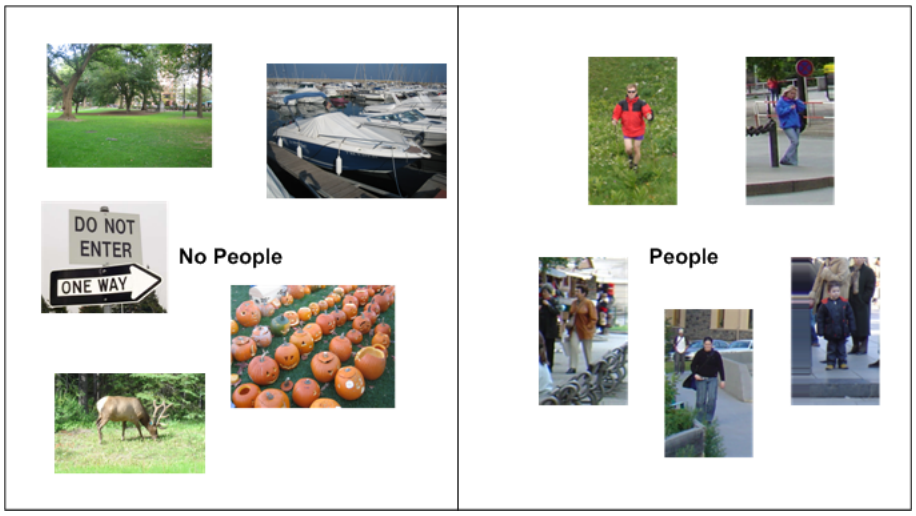


Figure 3.1: Negative and Positive Images

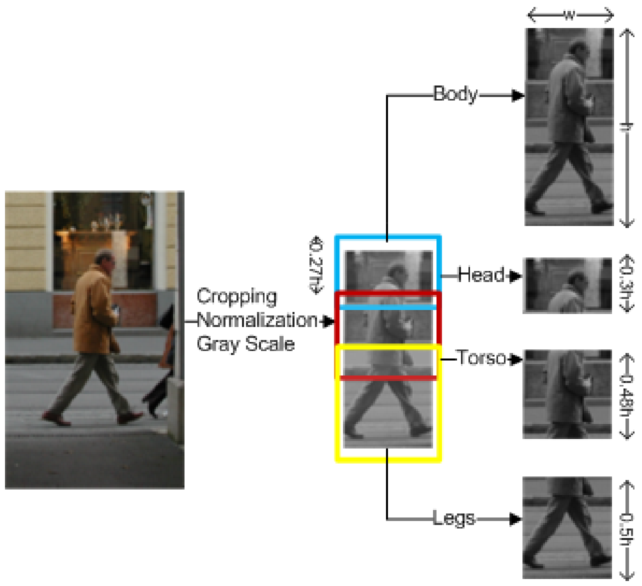


Figure 3.2: Body Parts Segmentation

3.3.3 Proposed Algorithm Design

The proposed algorithm extracts features based on edges. The main idea consists of identifying characteristic edges of each body part and generate four edge body parts models. In order to explain our design and implementation we have to describe four main concepts: edgelet, edge feature, weak classifier and body part model.

3.3.3.1 Edgelet

An edgelet is a sequence of orientations and associated movements which represent an edge's shape (see Figure 3.3). In this work according to the size of the images (58x24 pixels) and original algorithm [15], the possible length (k) of one single edgelet is from 4 pixels to 12 pixels. And the edgelet features we use consist of single shapes, including lines, 1/8 circles, 1/4 circles, and 1/2 circles. We use 36 types of lines (four orientations: 0,45,90 and 135º; and 9 dimensions :4-12 pixels; $4\,orientations \times 9\,dimensions = 36$). We generate arcs from 4 pixels to 12 pixels as described below:

Algorithm 3.1 Arcs Generation Algorithm

```
1. Circumferences are generated. Their perimeters ($P_{circumference}$) have to follow:
       $\left(\frac{1}{8}, \frac{1}{4}\, or\, \frac{1}{2}\right) \times P_{circumference} \geqslant k,\ k = 4, \ldots, 12,\ P_{circumference} \in \mathbb{N}$    (3.1)
2. Intensity and normal vector by 3×3 Sobel kernel convolutions are calculated.
3. The orientation of the normal vector is quantized into six discrete values (see Figure 3.3).
4. Using Freeman chain code [23], we follow circle contour and extract its sequence of quantized orientations and movements.
5. Finally, segments of length $k$ ($k = 4, \ldots, 12$) are extracted.
```

Finally, we have a total of 775 edgelets (36 lines and 739 arcs). For example, when the size of the body image is 24×58, the overall number of possible edgelet features is 1078800 ($24 \times 58 \times 775 = 1078800$).

3.3.3.2 Edge features

Edge features are extracted in order to generate each edge body part model. In the first place, the image intensity (M^I) and normal vector (V^I) are calculated by 3×3 Sobel kernel convolutions. Then the normal vector is simplified for computational efficiency by quantizing the orientation of the normal vector into six discrete values (see Figure 3.3).

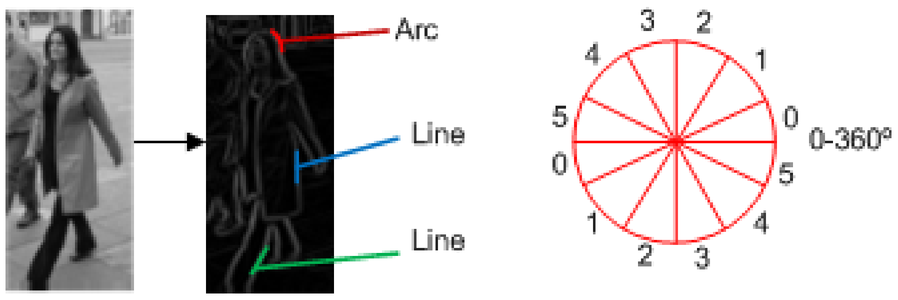


Figure 3.3: Sobel Convolution and Orientation Quantization

The dot product between two normal vectors is approximated [15] by the following function:

$$l\left[x\right]=\begin{cases}1 & x=0\\ \frac{4}{5} & x=\pm1,\pm5\\ \frac{1}{2} & x=\pm2,\pm4\\ 0 & x=\pm3\end{cases} \tag{3.2}$$

where the input x is the difference between two quantized orientations.

Then edge features ($f\left(E,I,w\right)$) are calculated by:

$$f\left(E,I,w\right)=\frac{1}{k}\sum_{i=1}^{k}M^{I}(u_i+w)\cdot l\left[V^{I}\left(u_i+w\right)-V_i^{E}\right] \tag{3.3}$$

Each feature is calculated for each position (w) of the image (I) along each kind of edgelet (E). $\left\{V_i^E\right\}_{i=1}^{k}$ denotes the quantized edge orientation of edgelet (E) on position i, being k the edgelet length and u_i the relative edgelet position (Freeman chain code) due to motion associated with the quantized orientation (V_i^E). Finally every edge feature is normalized to $[0,1]$ and quantized with $n=16$ bins resulting in ($\hat{f}\left(E,I,w\right)$).

3.3.3.3 Weak classifier

We use the boosting method to learn the body parts detectors. According to [24], boosting is a method for finding a highly accurate hypothesis (classification rule) by combining many "weak" hypotheses, what means that each of them is only moderately accurate. Typically, each weak hypothesis is a simple rule which can be used to generate a predicted classification for any instance. Then the weak classifier h^{weak} based on edgelets can be defined as:

$$h^{weak}\left(E,I,w\right)=\frac{1}{2}\sum_{j=1}^{n}ln\left(\frac{W_{+1}^{j}+\varepsilon}{W_{-1}^{j}+\varepsilon}\right)\cdot B_n^j\left(\hat{f}\left(E,I,w\right)\right),\ n=16\,bins \tag{3.4}$$

Where B_n^j is defined as:

$$B_n^j\left(u\right)=\begin{cases}1, & u\in\left[\frac{j-1}{n},\frac{j}{n}\right),\ j=1\ldots n\\ 0, & otherwise\end{cases} \tag{3.5}$$

W_c^j is defined as:

$$W_c^j = P\left(\hat{f}\,(E,w) \in bin_j,\, y=c\right),\ c=\pm 1,\ j=1\ldots n \tag{3.6}$$

And ε is a smoothing factor [24]:

$$\varepsilon = \frac{1}{m} \tag{3.7}$$

Where m is the number of training examples.

A weak classifier is calculated for each position (w) of the image (I) and each kind of edgelet (E). W_c^j consists of the probability of edgelet (E) occurrence for each position and each class (positive $+1$ or negative -1). Then the Adaboost algorithm [24] is used to train and learn a strong classifier as a linear combination of a series of weak classifiers as explained below.

3.3.3.4 Body part model

After calculating every weak classifier, we have to train each edge body part model. We have defined two different training approaches: a monolithic classifier and a cascade of classifiers. As we have already described in section 3.3.3.1 we have 775 different edgelets, in order to reduce computational cost and to identify the most characteristic edgelets of each body part we have made a top-100 edgelet ranking as described below:

Algorithm 3.2 Top-100 Edgelet Ranking

```
1. For $iteracion = 1\,to\,N_{iterations}$
   (a) Collect positive and negative examples in a bootstrap way.
   (b) For $n = 1\,to\,N_{edgelets}$
       i. Select the best classifier of $h^{weak}\,(E(n), w)$ (minimum error decision).
       ii. Evaluate precision. $Precision\,(n) = \frac{True\,Positive\,Rate}{True\,Positive\,Rate + False\,Positive\,Rate}$
   (c) Select the best 100 edgelets and actualize edgelet repetitions
       $Edgelet_{repetitions}(e) = Edgelet_{repetitions}(e) + 1,\ e \in 100\,best\ edgelets.$
2. Select 100 most repeated edgelets.
```

We use the Top-100 edgelet ranking of each body part model and, as we have already commented, the training is not only focused in reducing false positive but in getting good precision results. The final people model will be less complex (using a smaller number of classifiers but those which belong to Top-100 edgelet ranking) and the completed system will be faster (not scanning the entire image and using a simplified model), whilst trying to maintain good precision results. Figure 3.4 shows an example of using Top-100 edgelet ranking. We have trained a body part with 25000 negative examples and 2416 positive examples, with and without using Top-100 edgelet ranking. Logically by using only 100 of the 775 possible edges we increase the error rate, but in the worst cases the error is only 0.008% higher.

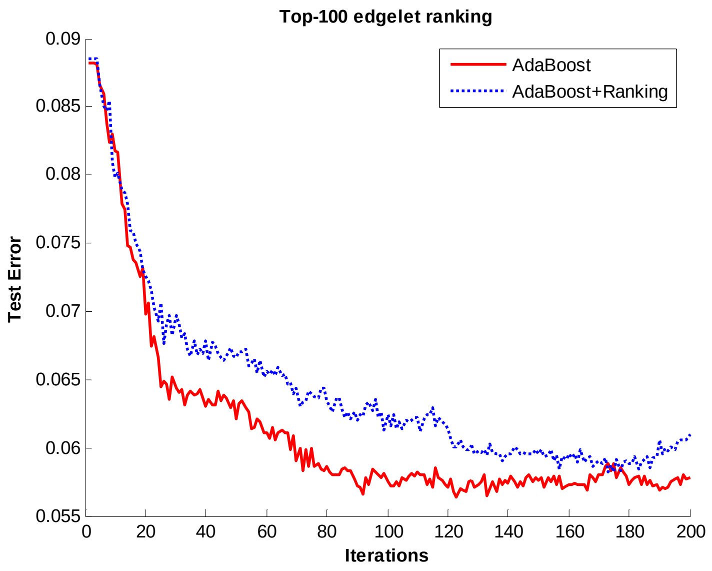


Figure 3.4: Top-100 Edgelet Ranking

Monolithic classifier We use the Top-100 edge ranking obtained previously to train and learn a strong classifier as a linear combination of a series of weak classifiers. Adaboost is used to learn strong classifiers, called layers, from the Top-100 weak classifiers' pool. The strong classifier H is a linear combination of a series of weak classifiers:

$$H = \sum_{j=1}^{T} \alpha_j h^{weak} (E, w) \tag{3.8}$$

Where α is defined as:

$$\alpha_j = \frac{1}{2} ln \left(\frac{1 - \epsilon_j}{\epsilon_j} \right) \tag{3.9}$$

Where ϵ is the error decision [24] and T is the number of weak classifiers in H.

The final model is defined by the number of features T, Adaboost model (decision rules h^{weak} and learned weights α_j). During the training phase we have used 2416 positive examples and 25000 negative examples, the final edge part models are strong classifiers as a linear combination of 200 weak classifiers. During the testing phase we have used 1126 positive and 8000 negatives examples. The training dataset does not need to be balanced because AdaBoost works properly with redundant (majority vs. minority class) data, reducing your chances of being selected because they are easily well classified, and thus forced to reduce their impact on the construction of the classifier. Some testing results can be seen in Figure 3.5. According with these results, we can see positive and negative distributions of each body part. All body parts follow similar distributions and there is overlap between positive and negative distributions in all cases.

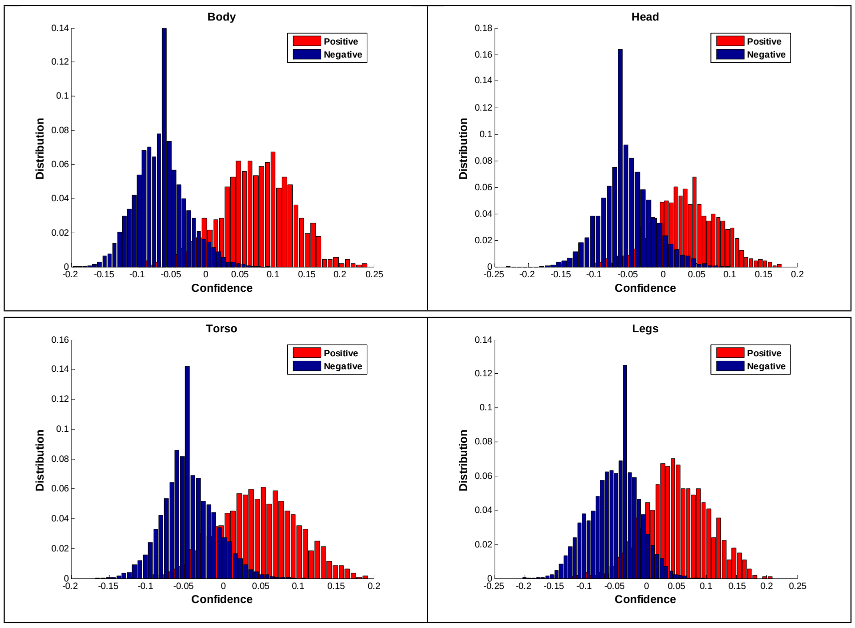


Figure 3.5: Testing Monolithic Results

Cascade classifier The training algorithm is a cascade Adaboost [25]. Using the Top-100 edge ranking information a cascading structure is built (each cascade stage C is composed of one strong classifier H -see equation 3.8- and a threshold b).

$$C = H - b \tag{3.10}$$

The learning procedure of one layer is referred to as a boosting stage. At the end of each boosting stage, the threshold b is tuned so that C has a high detection rate and new negative samples for the next stage are collected in a bootstrap way.

Algorithm 3.3 Cascade Adaboost Algorithm

```
1. Define the maximum acceptable false positive rate per layer $F_i^{max}$ and the minimum acceptable detection rate per layer $D_i^{min}$.
2. Define a target overall false positive rate $F_{target}$.
3. Select an initial train and validation set of positive ($P$) and negative $N$ examples in a bootstrap way.
4. Initialize false positive rate, detection rate and counter: $F_0 = 1.0$; $D_0 = 1.0$; $i = 0$.
5. While $F_i > F_{target}$
   (a) Initialize actual false positive rate, detection rate and counter: $f = 1$; $d = 1$; $i = i + 1$.
   (b) Initialize number of weak classifiers and false positive rate per layer: $n_i = 0$; $F_i = F_{i-1}$.
   (c) While $f > F_i^{max}$
       i. Add another weak classifier: $n_i = n_i + 1$.
       ii. Use train set to train a strong classifier $H_i$ with $n_i$ features using Adaboost algorithm and top-100 edgelet ranking.
       iii. Use validation set to find threshold $b_i$ (see equation 3.10) which satisfied: $D_i^{min} \geq d \times D_{i-1}$.
       iv. Revalue using $b_i$ current cascade ($C_0 ... C_i$) classifier on validation set to determinate $f$ and $d$.
   (d) Actualize $D_i = D_{i-1} \times d$ and $F_i = F_{i-1} \times f$ .
   (e) If $F_i \geq F_{target}$
       i. Evaluate the current cascade detector on the set of non people images and create a new training and validation set of negative examples with any false detections.
```

The final model ($C_0 ... C_i$) is defined by the number of stages i, number of features of each stage n_i, Adaboost model H_i (decision rules, learned weights) of each stage, and threshold of each stage b_i. During the training phase we have used 2416 positive examples and 5000 negative examples per Adaboost stage. Each final edge part models is a 4 layer cascade of classifiers which include between 100-300 features. During the testing phase we have used 1126 positive and 8000 negatives examples. Some testing results can be seen in Figure 3.6. According to these results, we can see positive and negative distributions of each body part. As in the monolithic case, there is an overlap between positive and negative distributions. However, in this case the negative distribution is more concentrated and in some body parts the overlap is clearly smaller (Body and Torso).

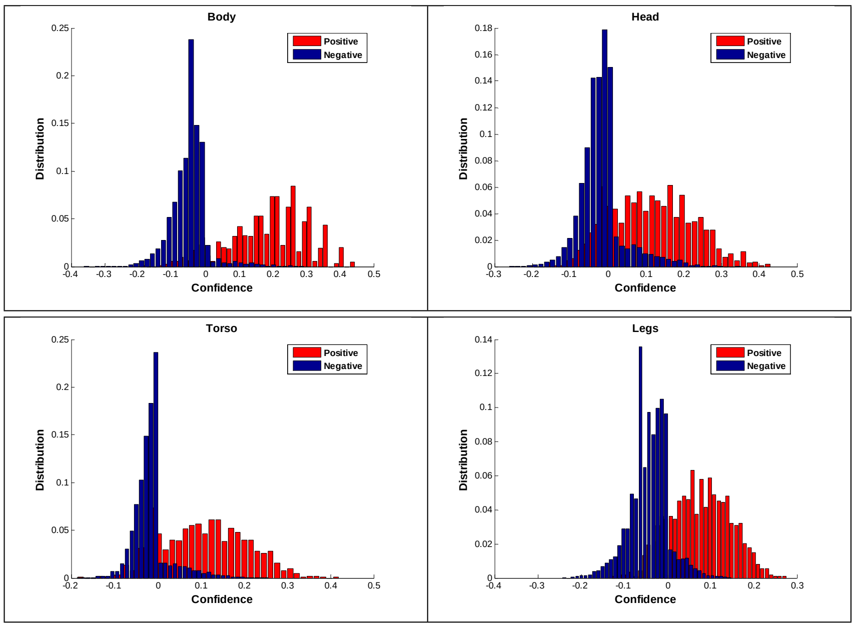


Figure 3.6: Testing Cascade Results

Adaboost implementation In order to implement both algorithms we have used a Matlab toolbox with Adaboost algorithm implementations (GML `http://graphics.cs.msu.ru/en/science/research/machinelearning/adaboosttoolbox`, more information can be found in appendix A). This toolbox has three different boosting schemes: Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. According to authors Real AdaBoost is the generalization of a basic AdaBoost algorithm first introduced by Fruend and Schapire [26], Gentle AdaBoost is a more robust and stable version of real AdaBoost [27] and Modest AdaBoost [28] has a better generalization capability and resistance to overfitting. We have used the Gentle AdaBoost version because it performs better than the Modest AdaBoost and slightly better than Real AdaBoost (see figure 3.7) on regular data, according to authors it is considerably better on noisy data, it is much more resistant to outliers, and the overfitting problem is solved using validation sets during the training phase. In the example1 (see figure 3.7) we have trained a body part with 14000 negative examples and 2416 positive examples, we can see as long-term (200 iterations) GentleAdaBoost performs slightly better results than RealAdaBoost (lower test error) whilst ModestAdaBoost remains constant, learns new weaks classifiers but without improving the global error. In the example2 (see figure 3.7) we have trained a body part with 5000 negative

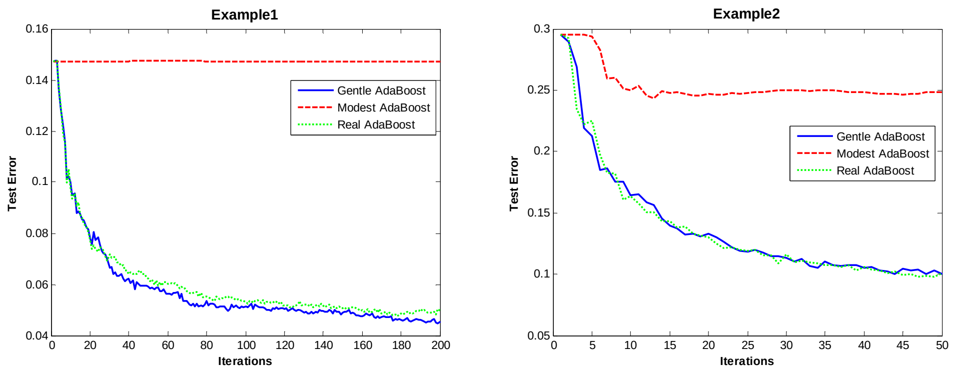


Figure 3.7: Real Gentle Modest

examples and 2416 positive examples and as the previous example we can see how short-term (50 iterations) GentleAdaBoost and RealAdaBoost perform better results than ModestAdaBoost. ModestAdaBoost gradually reduces the error rate, but after 15-20 iteration remains almost constant.

3.3.4 Proposed Final System Design

In previous sections we have explained the design and implementation of the training stages of our people detector based on edges. The edgelets generation and cascade Adaboost algorithm have been implemented in Matlab. However, our final system consist of four main running tasks (image normalization, feature extraction, weak classifiers generation and classification) that have been implemented in C++ and integrated into the VPULab system (see more information in chapter 4). The image normalization (section 3.3.2), feature extraction (section 3.3.3.2) and weak classifiers generation (section 3.3.3.3) have been already explained: in this section we describe briefly the final system and the classification stage, the last running task.

3.3.4.1 Final system overview

In Figure 3.8 we can see the system functionoal blocks. In first place, a frame acquisition is realized and then motion segmentation is performed to detect moving pixels (Background/Foreground Extraction Module). Subsequently, the Object Extraction Module analyzes the connected regions of the binary foreground mask (obtained in previous stage) to detect blobs. Then, the Object Tracking Module generates the trajectories of the blobs between consecutive frames using color (current image) and position information (blob). Afterwards a classification stage is applied to the blobs for distinguishing between people and non-people classes using current image, blob information, blob trajectories and body parts models learned during training phase.

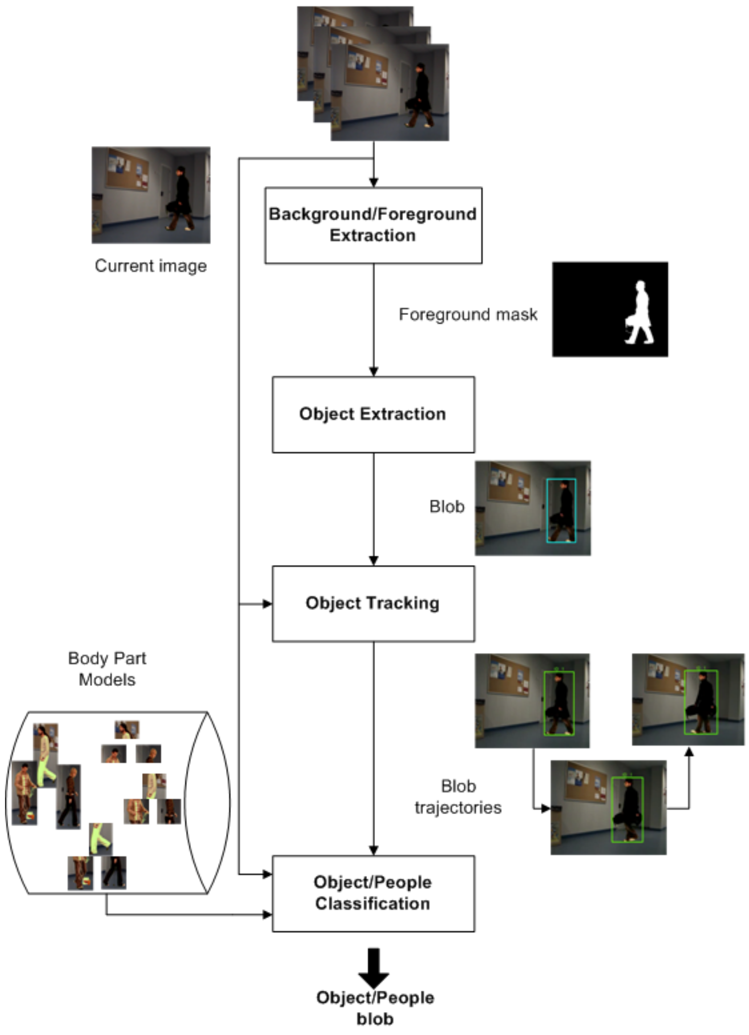


Figure 3.8: System Overview

3.3.4.2 Classification

Each blob's image is normalized, edge features are extracted, and weak classifiers are generated. Then we generate the four body parts models (monolithic or cascade classifiers). The classification consists of comparing the new models with the models trained previously. In the case of a monolithic classifier we just evaluate the model with equation 3.8: the signum of strong classifier H represents the class (positive/person $+1$ or negative/no person -1), and its absolute magnitude is the "confidence" of the decision. In the case of the cascade classifier the classification is a sequence of monolithic classifiers one after other, each blob is evaluated at each stage (monolithic classifiers) and a threshold (see equation 3.10). If the blob verifies the whole process (signum of each cascade stage C_i is positive), it is classified as person, otherwise, it is considered as no person. In both cases, the “confidence” of the decision is the last stage output absolute magnitude.

The final classification process consists of evaluating the four body parts models (monolithic or cascade classifiers) providing four independent evidences. The final evidence about the analyzed blob being a person is obtained by averaging the evidences provided by the four body parts detectors.

3.4 Motion Information

As we have already commented our people detector makes people/no people classification using appearance information. However we also want to add robustness to the detection with motion information through a tracking algorithm. Tracking refers to predicting and establishing the position and orientation of an object in an image sequence; it allows us to identify (blob identification) and monitor object movements over time. In this section we describe the tracking algorithm used in this work (section 3.4.1) and the use of motion information to add robustness to the people detection (section 3.4.2).

3.4.1 Tracking

Video tracking is the process of locating a moving object (or several ones) in time using a camera. The tracking algorithm implemented within the VPULab is based on the Kalman Filter [29]. The Kalman filter [30] is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown. The tracking based on Kalman filter provide accurate continuously updated information about the position and velocity of an object given only a sequence of observations about its position (blob information), each of which could includes some errors.

The Kalman filter has two distinct phases: Predict and Update. The predict phase uses the state estimate from the previous timestep to produce an estimate of the state at the current timestep. This predicted state estimate is also known as the a priori state estimate because, although it is an estimate of the state at the current timestep, it does not include observation information from the current timestep. In the update phase, the current a priori prediction is combined with current observation information to refine the state estimate. This improved estimate is named the a posteriori state estimate.

Once the Kalman filter defines the new positions of the objects in the scene, the correspondences between the blobs detected in the current frame and the blobs detected in the previous frames (tracks) are realized according to these new estimated positions and color similarity.

3.4.2 Improving People Detection by Using Motion Information

As we have already commented the tracking allows us to identify and monitor object movements over time. For this reason we are able to extract motion information of objects present in the scene; we want to use this information in order to add robustness to the people detection. We propose two simple uses of the motion information provided by the tracking: People classification over time and Static object analysis.

3.4.2.1 People classification over time

The first use of the motion information provided by the tracking is based on the simple idea that objects/people are always objects/people over time, it means, if one blob was classified as object during some frames and suddenly in a frame this blob was classified as people, it would be probably a mistake; and in the same way in the opposite case (a blob was classified as people during some frames).

We propose a simple weighted decision over time, it means, the final decision will be a weighted sum of current evidence and accumulated over time. Thus the decision is averaged over time, eliminating possible errors, but allowing transitions between detected blobs as people to objects and vice versa (e.g., when a person is occluded by an object (table, door, car, etc) or vice versa).

$$Final\ Decision = \alpha Current\ Evidence + (1-\alpha) Accumulated\ Evidence,\ 0 \leq a \leq 1 \tag{3.11}$$

where α is the accumulated evidence weight.

3.4.2.2 Static object analysis

The second use of the motion information provided by the tracking is focused on the critical classification of static objects. In video surveillance, moving blobs tend to be vehicles, people or objects carried by people. However, static blobs tend to be static humans or static objects. Trying to improve this critical classification between static human and static object we expect to realize more accurate people detection over static blobs.

We define as static blobs, the ones for which their speed do not vary over time or vary below a threshold. On this class of blobs, we perform a more exhaustive classification process to reduce the false positive rate without reducing the true positive rate. The process involves evaluating each of the models of body parts around a blob's area (search area) and the end result is a weighted average (see Figure 3.9). The weighted average is linearly decreasing with distance. In this way, we obtain more reliable evidences but increasing the computational cost, each detector must be evaluated $(2p + 1)(2q + 1)$ times. For example, in the same figure $p = 3$ and $q = 4$, so we have to evaluate each part detector $(2 \cdot 3 + 1)(2 \cdot 4 + 1) = 63$ times.

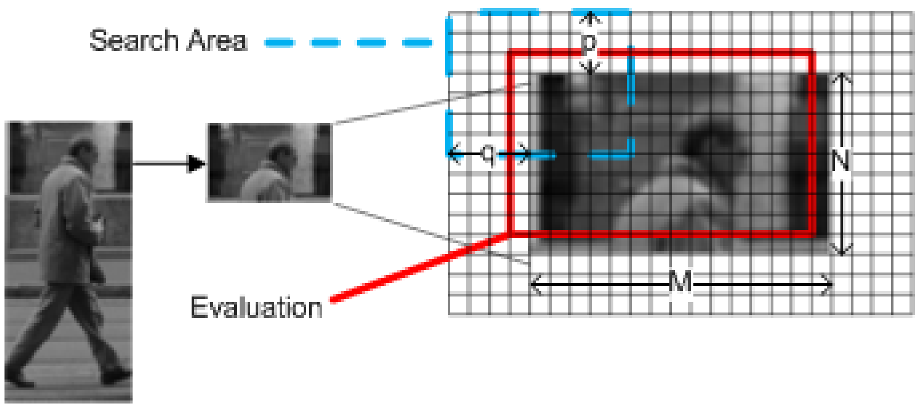


Figure 3.9: Search Area

Chapter 4

Integration

4.1 Introduction

In this chapter we describe the integration of the new detector in the system implemented in the VPULab . We first describe the external basic structure and interface that the detectot must have (see section 4.2) and then we describe the internal structure of the detector (see section 4.3).

4.2 External Structure

The video processing framework implemented in the VPULab includes a class in C++ "PeopleDetector" (more information about c++ classes in appendix B) to make it easy to include new detectors of people. Each new person detector corresponds to an extension of the parent class. This parent class "PeopleDetector" defines a basic structure and methods for detection. Each new people detector must include at least three lists of blobs: “listNewBlobs”, “listBlobs” and “outputListBlobs”, and two methods: “checkObject” and “PeopleLikelihoodOfBlobs”.

To integrate the new detector in the system, we must be able to replace the previous module to our module, without affecting the overall system. For this reason, the external structure (interface input / output) should be maintained (see Figure 4.1). The people detector module must accept as input a list of “PeoleBlobs” (listNewBlobs) and must generate as output a list of “ObjectBlobs” (outputListBlobs). A “BasicBlob” is the C++ class to describe a basic blob, it includes blob’s dimensions (width and height), position (x and y) and identification (ID). A “PeopleBlob” is an extension of “BasicBlob” class, and it also includes a score attribute. The score will include the evidence found during the detection process. An “ObjectBlob” is an extension of “BasicBlob” class, it also includes the probability of being a person “peopleLikelihood”.

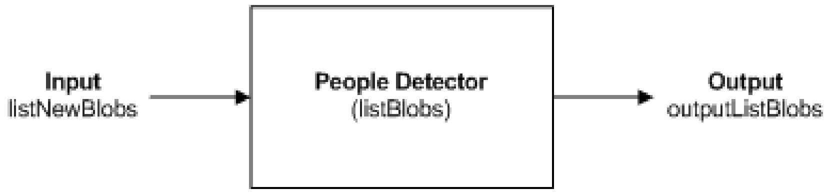


Figure 4.1: External Structure

The people detector at each iteration, the method “PeopleLikelihoodOfBlobs” updates the list of blobs to be processed “listBlobs” basing on the ID of the input list of blobs “listNewBlobs” and then each blob is processed with the method “checkObject”. This second method executes the people detection and generates a people evidence. As a result, each blob is processed and the obtained evidences are stored (score attribute). At the end, the final evidence is obtained and the output list of blobs “outputListBlobs” generated with the final evidence (peopleLikelihood attribute).

4.3 Internal Structure

To integrate our people detector, we have implemented two new C++ classes: “EdgePeopleDetector” and “FusionEdgePeopleDetector”. Both classes are an extension of the PeopleDetector class. The first class allows the execution of independent body parts detectors (see Figure 4.2) and the second one allows the execution and combination of all body parts detectors (see Figure 4.3).

In both cases the external structure (interface input / output) of "PeopleDetector" class is maintained, the detector module accepts as input a list of “PeoleBlobs” (listNewBlobs) and generates as output a list of “ObjectBlobs” (outputListBlobs). And both detectors include the methods “checkObject” and “PeopleLikelihoodOfBlobs”.

4.3.1 Edge People Detector

If we run each detector independently, the method “PeopleLikelihoodOfBlobs” is in charge of updating the list of blobs to be processed “listBlobs” based on the ID (assigned during the tracking) of the input list of blobs “listNewBlobs”, normalizing the blob’s image and executing the “checkObject” method for each blob. The updating stage allows us to monitor object’s classification over time (see section 3.4.1). The image normalization is common to all body parts detectors (resize and gray scaled, see section 3.3.2). Then the method “checkObject” executes the people detection stages: blob's image segmentation, edge features extraction, weak classifiers generation and people classification. The method "PeopleLikelihoodOfBlobs" finally collects the results of method "checkObject" and generates the output list of blobs "outputListBlobs" with the classification.

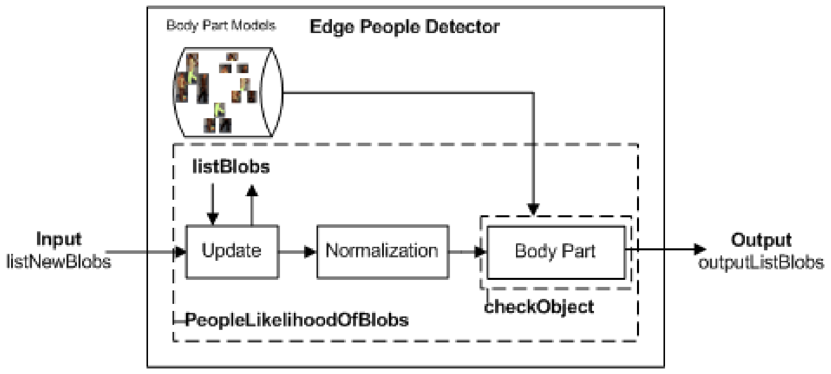


Figure 4.2: Edge People Detector

4.3.2 Fusion Edge People Detector

If we run the detector based on fusion, the overall operation is almost the same (update, normalization and checking). The main difference is that the method "checkObject" executes independently the method "checkObject" of each body part detector (see Figure 4.3). Then the method "calculateFusionEvidence" performs the final classification as a fusion of the four evidences and finally the method "PeopleLikelihoodOfBlobs" collects the results of method "checkObject" and generates the output list of blobs "outputListBlobs" with the classification.

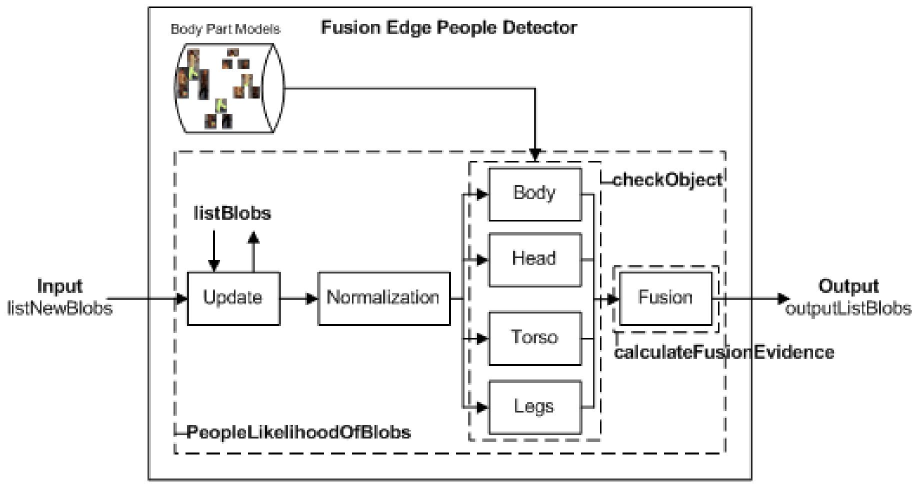


Figure 4.3: Fusion Edge People Detector

Chapter 5

Experimental Work

5.1 Introduction

In this chapter, we describe the experiments carried out for testing the proposed people detector (described in section 3.3) without and with motion information (described in section 3.4), and compare them with a previous people detector (described in section 3.2). In the next sections, we describe the dataset used to test the methods (section 5.2), the metrics (section 5.3) and the evaluation results comparing methods (section 5.4).

5.2 Experimental data

5.2.1 Dataset Used

Experiments were carried out on several test sequences from public datasets related with the people detection/object classification task. They are:

- **PETS2006**: Ninth IEEE International Workshop on Performance Evaluations of Tracking and Surveillance, June 2006.URL `http://pets2006.net/`

- **i-LIDS dataset for AVSS2007**: Fourth IEEE International Conference on Advanced Video and Signal based Surveillance, September 2007. URL: `http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_d.html`

- **WCAM**: Video Surveillance video sequences (The test visual material used in this work has been provided with courtesy of Thales Security Systems within the scope of the IST FP6 WCAM project). URL: `http://wcam.epfl.ch/`

- **VISOR**: Video Surveillance Online Repository.
  URL: `http://imagelab.ing.unimore.it/visor`
- **CVSG**: A Chroma-based Video Segmentation Ground-truth. URL: `http://www-vpu.ii.uam.es/CVSG/`
- The well known **"Hall_monitor"** sequence.

In order to evaluate the performance of the proposed approach, we have selected sequences where there are people and other objects. We define as object every blob detected in the sequence that is not a person, and therefore no-people blobs detected during background extraction will be classified as objects. We are going to compare two different people detectors: the previous detector uses extracted blobs and foreground mask obtained during background extraction, whilst our proposed algorithm does not depend on foreground mask but on extracted blobs and the image. The people classification will be defined over each detected blob.

We want to compare algorithms with and without background subtraction because the previous detector depends very much on the foreground mask obtained. For this reason we have grouped all the test sequences into different categories depending on

1. Availability of background/foreground masks:
    - Gamma background extraction sequences: It is related with sequences which background/foreground ground truth is not available. Then the background subtraction is realized.
    - Background/Foreground ground truth sequences: It is related with sequences whose background/foreground ground truth is available. Then the blob extraction is realized directly over ground truth masks.
2. Scene complexity, which is related with the number of objects, their velocity, partial occlusions and the people classification difficulty.
    - Low: Simple scenarios, few blobs and without occlusions.
    - Medium: More complex scenarios, more blobs and occlusions.
    - High: Complex scenarios, many blobs and occlusions.

A description of categorization is shown in Table 5.1, whilst Figure 5.1 shows an example of each category with a description of the content:

Background/Scene complexity	Background Extraction	Background Ground Truth
Low	BE-1	GT-1
Medium	BE-2	GT-2
High	BE-3	GT-3

Table 5.1: Test Sequence Categorization

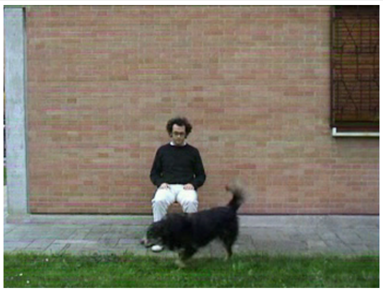
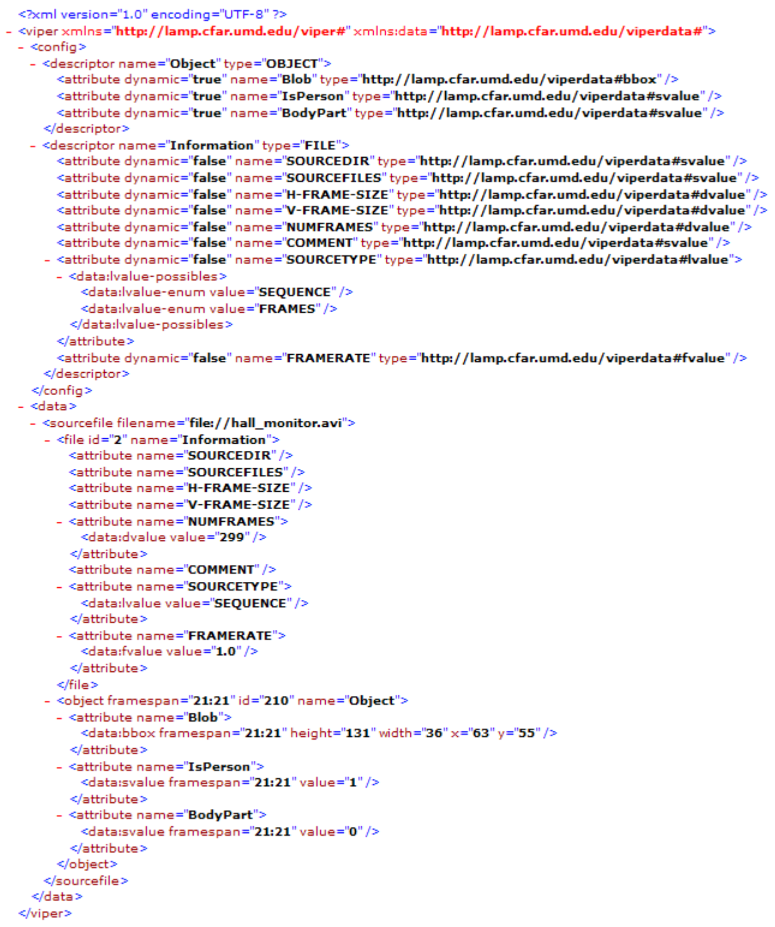
Background Extraction	Background Ground Truth
Name: Hall_monitor. Resolution: 352x240. Number of frames: 300. Number of Blobs (Static): 491(39). Scene complexity: Low. Category: BE-1. Description: Two people who take or leave an object.	Name:CVSG_S1. Resolution: 352x288. Number of frames: 193. Number of Blobs (Static): 359(190). Scene complexity: Low. Category: GT-1. Description: Single person that leaves an object.
	
Name: VISOR_S1. Resolution: 320x240. Number of frames: 534. Number of Blobs (Static): 443(177). Scene complexity: Medium. Category: BE-2. Description: One person sitting and a dog. With occlusions.	Name:CVSG_S2. Resolution: 352x288. Number of frames: 349. Number of Blobs (Static): 617(104). Scene complexity: Medium. Category: GT-2. Description: Two people who take an object. With occlusions.
Name: PETS_S1. Resolution: 750x576. Number of frames: 1221. Number of Blobs (Static): 3675(518). Scene complexity: High. Category: BE-3. Description: Crowded scenario that contains a single person leaving an object.	Name:CVSG_S3. Resolution: 352x288. Number of frames: 897. Number of Blobs (Static): 2406(1626). Scene complexity: High. Category: GT-3. Description: Three people who take and leave objects. With occlusions.

Figure 5.1: Video Categories Examples

5.2.2 Ground Truth

For each video file, we have created a ground truth description of the detected blobs, using two attributes: IsPerson (People/No People) and BodyPart (Head/Torso/Legs). The first attribute "IsPerson" is people classification (1/0) and the second attribute "BodyPart" is annotated just in case the blob constitutes an unique body part (1-head, 2-torso or 3-leg, the body is considered a complete person: IsPerson=1). The "BodyPart" attribute is not used in this work, but it has been included as we have identified this annotation as a requirement for future work (see section 6.3). There are some tools available to annotate video files: Anvil (`http://www.anvil-software.de/`), IBM Annotation Tool (`http://www.research.ibm.com/VideoAnnEx/`) and Viper Annotation Toolkit (`http://viper-toolkit.sourceforge.net/`). We have decided to use the Viper tool because it is the most popular in the research community, it is easy to manage and it has associated performance evaluation tools (for more information see appendix C). An example of a ground-truth annotation file is shown in Figure 5.2.



```xml
  <?xml version="1.0" encoding="UTF-8" ?>
- <viper xmlns="http://lamp.cfar.umd.edu/viper#" xmlns:data="http://lamp.cfar.umd.edu/viperdata#">
  - <config>
    - <descriptor name="Object" type="OBJECT">
        <attribute dynamic="true" name="Blob" type="http://lamp.cfar.umd.edu/viperdata#bbox" />
        <attribute dynamic="true" name="IsPerson" type="http://lamp.cfar.umd.edu/viperdata#svalue" />
        <attribute dynamic="true" name="BodyPart" type="http://lamp.cfar.umd.edu/viperdata#svalue" />
      </descriptor>
    - <descriptor name="Information" type="FILE">
        <attribute dynamic="false" name="SOURCEDIR" type="http://lamp.cfar.umd.edu/viperdata#svalue" />
        <attribute dynamic="false" name="SOURCEFILES" type="http://lamp.cfar.umd.edu/viperdata#svalue" />
        <attribute dynamic="false" name="H-FRAME-SIZE" type="http://lamp.cfar.umd.edu/viperdata#dvalue" />
        <attribute dynamic="false" name="V-FRAME-SIZE" type="http://lamp.cfar.umd.edu/viperdata#dvalue" />
        <attribute dynamic="false" name="NUMFRAMES" type="http://lamp.cfar.umd.edu/viperdata#dvalue" />
        <attribute dynamic="false" name="COMMENT" type="http://lamp.cfar.umd.edu/viperdata#svalue" />
      - <attribute dynamic="false" name="SOURCETYPE" type="http://lamp.cfar.umd.edu/viperdata#lvalue">
        - <data:lvalue-possibles>
            <data:lvalue-enum value="SEQUENCE" />
            <data:lvalue-enum value="FRAMES" />
          </data:lvalue-possibles>
        </attribute>
        <attribute dynamic="false" name="FRAMERATE" type="http://lamp.cfar.umd.edu/viperdata#fvalue" />
      </descriptor>
    </config>
  - <data>
    - <sourcefile filename="file://hall_monitor.avi">
      - <file id="2" name="Information">
          <attribute name="SOURCEDIR" />
          <attribute name="SOURCEFILES" />
          <attribute name="H-FRAME-SIZE" />
          <attribute name="V-FRAME-SIZE" />
        - <attribute name="NUMFRAMES">
            <data:dvalue value="299" />
          </attribute>
          <attribute name="COMMENT" />
        - <attribute name="SOURCETYPE">
            <data:lvalue value="SEQUENCE" />
          </attribute>
        - <attribute name="FRAMERATE">
            <data:fvalue value="1.0" />
          </attribute>
        </file>
      - <object framespan="21:21" id="210" name="Object">
        - <attribute name="Blob">
            <data:bbox framespan="21:21" height="131" width="36" x="63" y="55" />
          </attribute>
        - <attribute name="IsPerson">
            <data:svalue framespan="21:21" value="1" />
          </attribute>
        - <attribute name="BodyPart">
            <data:svalue framespan="21:21" value="0" />
          </attribute>
        </object>
      </sourcefile>
    </data>
  </viper>
```

Figure 5.2: Example Ground Truth Viper File

5.3 Metrics

The performance of the detected events on a video sequence is evaluated in terms of Precision (P), Recall (R) and ROC curve. Precision represents the ratio between the true alarms (that is, they are in the ground truth) and the total number of alarms detected by the module. Recall represents the ratio between alarms that correspond to real alarms in the ground truth and the total number of alarms in the ground truth. "A receiver operating characteristic (ROC), or simply ROC curve, is a graphical plot of the sensitivity vs. (1 − specificity) for a binary classifier system as its discrimination threshold is varied. The ROC can also be represented equivalently by plotting the fraction of true positives (TPR = true positive rate) vs. the fraction of false positives (FPR = false positive rate)"[1]. S (Sensitivity) = TPR and E (Specificity) = 1-FPR.

[1]Fragment extracted from http://en.wikipedia.org

$$P = \frac{TP}{TP + FP} \tag{5.1}$$

$$R = \frac{TP}{TP + FN} \tag{5.2}$$

$$S = \frac{TP}{TP + FN} \tag{5.3}$$

$$E = \frac{TN}{TN + FP} \tag{5.4}$$

where TP (True Positive or true alarms) are the people blobs detected from the ground truth, FN (False Negatives or missed events) are the people blobs not detected from the ground truth and FP (False Positive or false alarms) are blobs detected as people that do not appear in the ground truth.

5.4 Performance evaluation comparison

In this section, experimental results of the proposed people detection system are presented. These results include an evaluation of people detection and computational cost. The system has been implemented in C++, using the OpenCV image processing library `http://www.intel.com/technology/computing/opencv/`. The tests have been performed on a Pentium IV with a CPU frequency of 1.79 GHz and 2GB RAM. Additional results can be found at `http://www-vpu.ii.uam.es/~agm2/people_edge_detector_demo.html`.

5.4.1 People Detection Results

In this section we have evaluated people detection results on several public video sequences manually annotated. Two different training approaches (a monolithic classifier and a cascade of classifiers) have been compared, we have compared the fusion detector and each body part detector independently, our proposal with previous detector have been compared, and the improving people detection by using motion information has been evaluated.

5.4.1.1 Monolithic classifier vs. cascade of classifiers

We have tested the performance of two different training approaches: a monolithic classifier and a cascade of classifiers. We present the results of one video of each category. Figure 5.3 shows the ROC curves (see section 5.3, Sensitivity vs. (1-Specificity)) corresponding to the application of both approaches. We can see that in most cases the results of the cascade approach are better or similar than the monolithic approach. It is generally observed that for the same value of sensitivity the value of specificity is worse. Besides the cascade computational cost will be equal to or less than monolithic approach. Although both approaches have the same number of weak classifiers, the monolithic approach always has to evaluate all the weaks classifiers, while the cascade approach has to evaluate only those which verify each stage of the cascade.

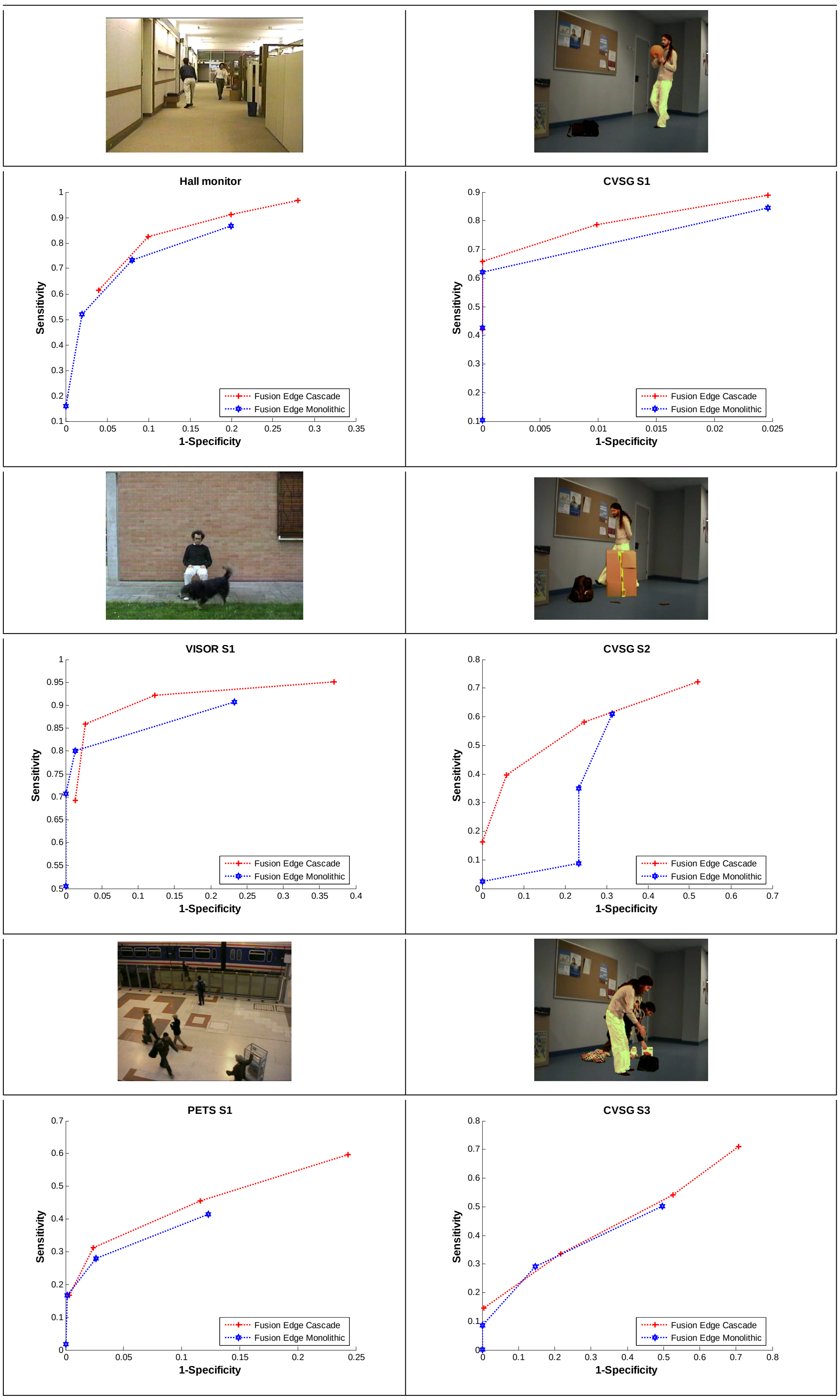


Figure 5.3: Cascade Approach vs. Monolithic Approach

5.4.1.2 Fusion vs. Body Part Detector

We have tested the performance of each body part detector independently and their fusion to compare results. Figure 5.4 shows the ROC curves corresponding to the application of the each body parts detectors and their fusion. We can see that in most cases the results of fusion are better or more stable than the detectors independently. For example, in the video PETS_S1, the head detector is better than the fusion. However, in the video CVSG_S1, this detector (Head) has the worst outcome of all. Even though some individual parts detectors have poor results, they are solved by the combined detector. The worst results were observed in the most complex sequences (PETS-S1 and CVSG-S3).

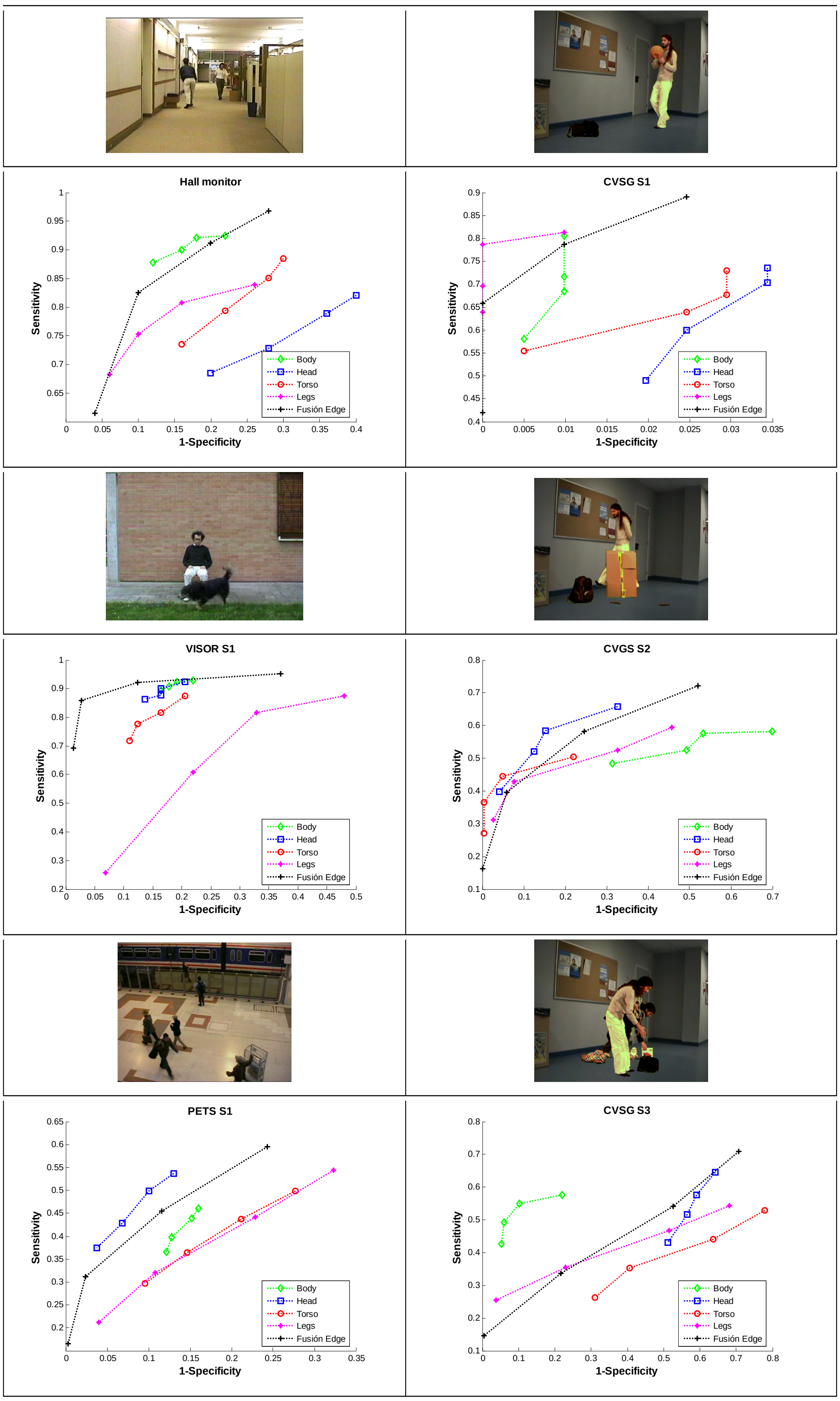


Figure 5.4: Body Parts Detectors vs. Fusion Detector

5.4.1.3 Fusion edge detector vs. Previous detector

We have tested the performance of the previous and the proposed algorithms to compare their results. We present the results of one video of each category to see the overall performance of both detectors. Figure 5.5 shows the ROC curves corresponding to the application of the two detectors. We see a relevant improvement of the results with respecto to the previous detector. We can see the improved results in the two types of sequences: BE and GT. As in the previous case the worst improvement was seen in one of the most complex sequences (CVSG-S3). The main problem is that objects are detected as divided in multiple parts due to partial occlusions or background extraction difficulty (e.g., people divided into several blobs) and therefore this effect reduces the performance of people detection. However, our proposed algorithm gets better results because if individual part detectors have low confidence (the absence of any body part), they would be enhanced by the combined detector. Another problem is when static objects are classified as people: we will try to solve these errors through using motion information.

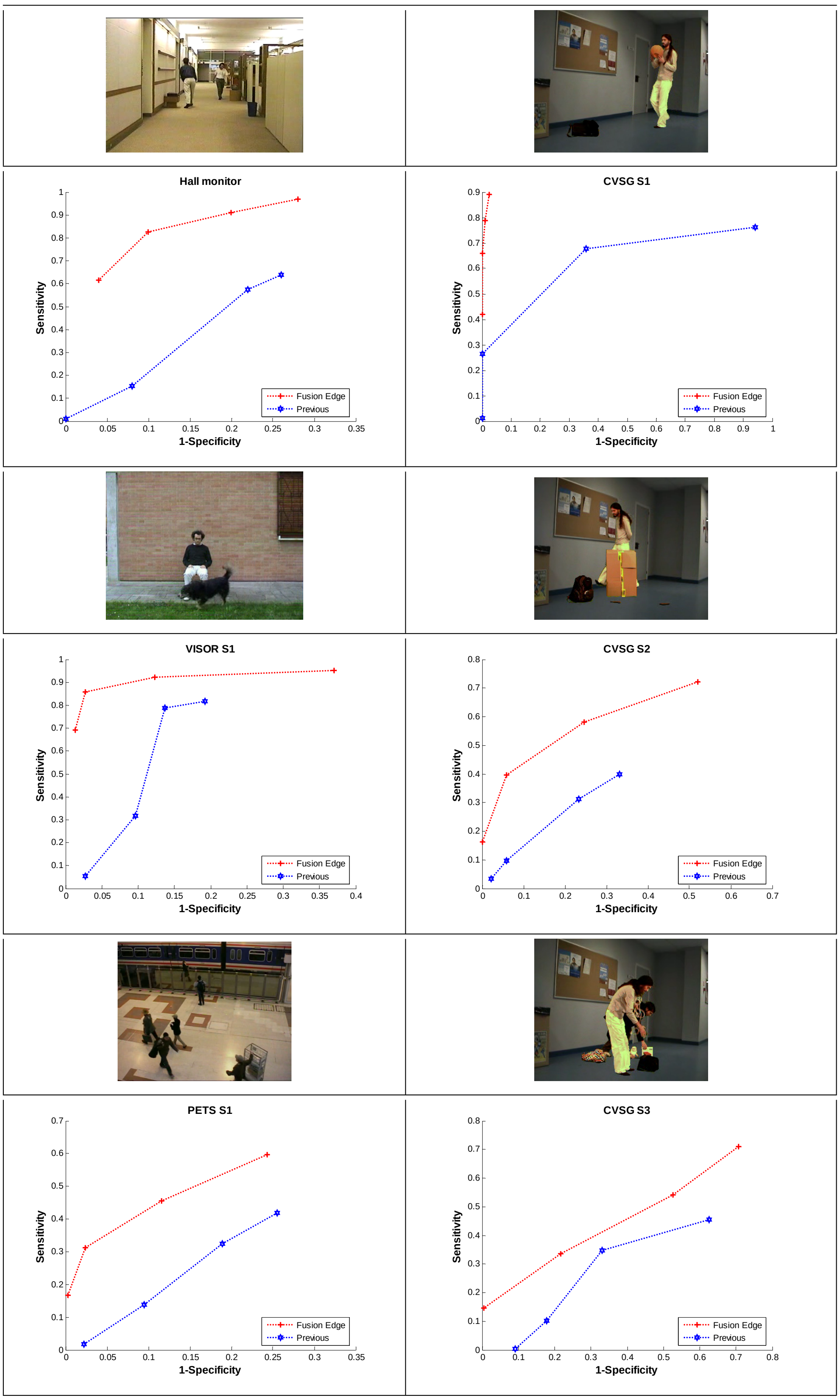


Figure 5.5: Fusion Edge Detector vs. Previous Detector

5.4.1.4 Improving people detection by using motion information

We have tested the performance of the proposed algorithm using motion information. On one hand, we present the results of using tracking information to make a people classification over time. On the other hand, we present the results of static object analysis.

People classification over time We use a simple weighted classification over time, what means that the final classification will be a weighted sum of current evidence and the one accumulated over time (more information in section 3.4.2.1). We have tested different accumulated evidence weights (from 0.6 to 0.9) with very similar results. Figure 5.6 shows the ROC curves corresponding to the application of the detector with an accumulated evidence weight of 0.8, it means, 80% current evidence + 20% accumulated evidence. We can see how the use of people classification over time does not provide significant improvements in any of the examples.

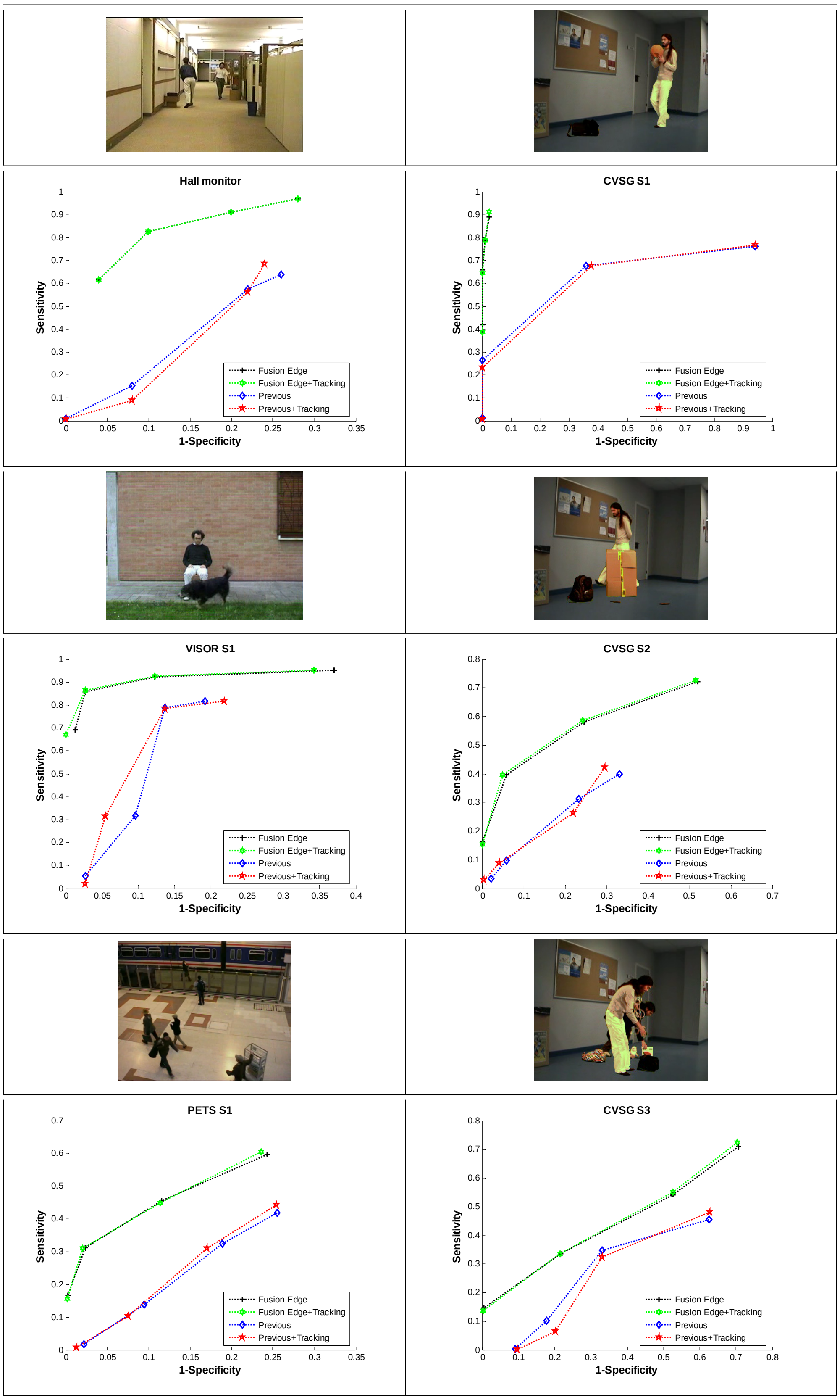


Figure 5.6: People Classification Over Time

Static object analysis We define as static blobs, the ones for which their speed do not vary over time or vary below a threshold. On this class of blobs, we perform a more exhaustive classification process to reduce the false positive rate without reducing the true positive rate (see section 3.4.2.2). We have tested different search areas (3x3, 5x5, 7x7..): the two first areas obtained very similar results, but as we keep increasing the search area the detection rate reduces. Figure 5.7 shows the ROC curves corresponding to the application of the detector with a search area of 3x3 pixels. On one hand, we can see how the analysis of static object has little effect over those objects which were previously classified correctly, even with static people (VISOR_S1 and PETS_S1), for this reason the detection rate remains almost constant. On the other hand, the analysis of static object has effect over those objects which were previously classified incorrectly. In videos PETS_S1 and CVSG_S3 we can see how the false positive rate is reduced, so there is an improvement.

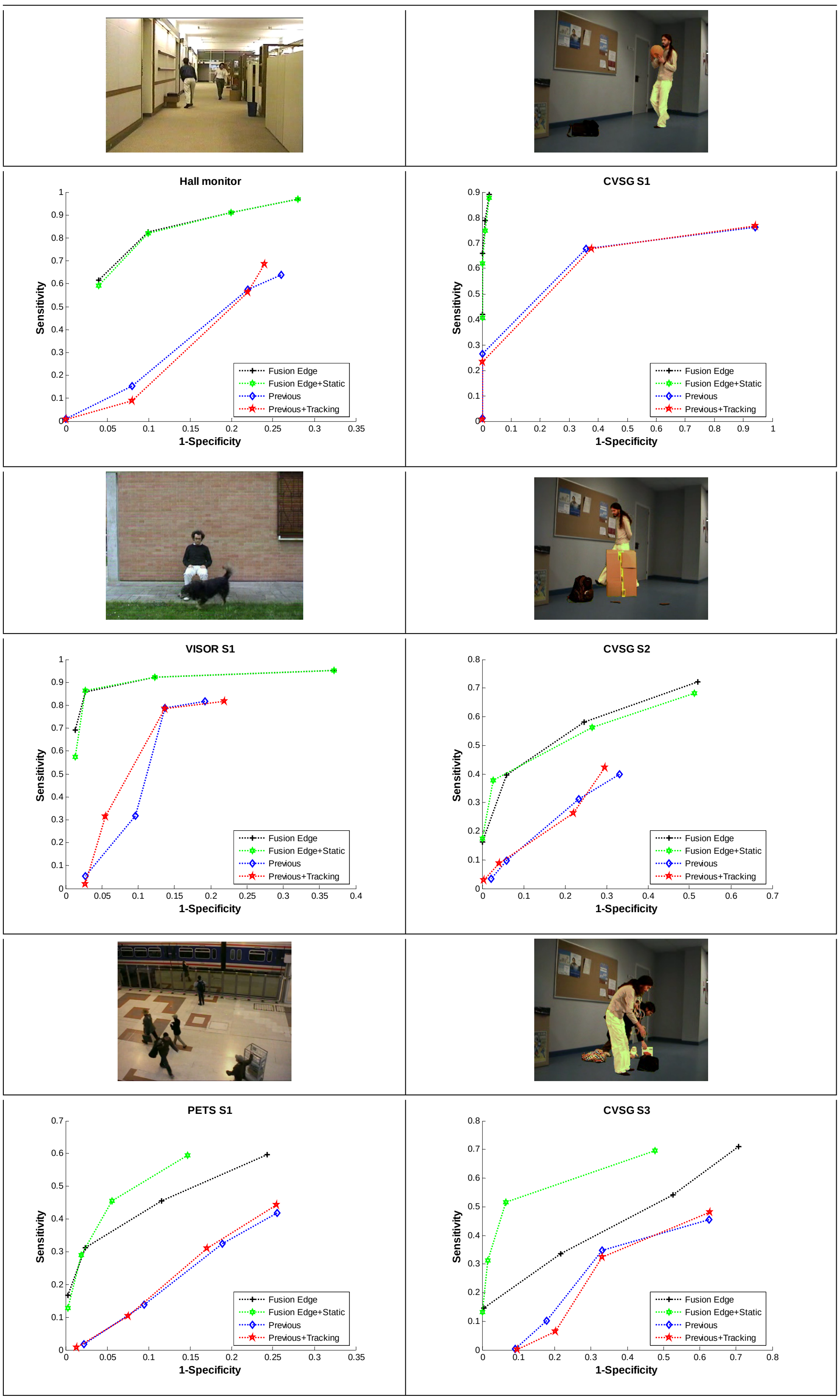


Figure 5.7: Static Object Analysis

5.4.2 Computational Cost

In this section, the computational performance of the previous and proposed algorithm without/with motion information is discussed. To evaluate the computational cost of people detection, we must clearly differentiate the sequences with background extraction from sequences with ground truth. In the first case, we can evaluate a real system with all stages. However, in the second case we can only evaluate the performance of the people detection stage.

5.4.2.1 Background Extraction

In Table 5.3a we can see three examples of computational costs (BE-1, BE-2 and BE-3). We can compare the previous detector, the fusion edge detector and the static object analysis. The people classification over time has not been included because the computational cost is the same with or without it. The results show clearly how the proposed detector significantly reduces the processing time, and therefore the rate of frames per second (fps) increases. Obviously, the videos with lower resolution obtain a higher frame rate (Hall_ monitor 40.42fps and VISOR_S1 50.82fps). The example with biggest resolution obtains a frame rate of 10.23fps, but we see that the 79.62% of time is dedicated to background extraction whilst just 13.09% to people detection.

It is noteworthy that in the previous algorithm the processing time per blob depends on the blob's size. Instead our algorithm does not depend on blob's size because of image normalization, so then the processing time per blob remains almost constant in different videos (4.00-4.25ms). The people detector with the analysis of static objects has a higher computational cost than the fusion edge detector, but it still remains lower than the previous detector. The processing time depends on the number of detected static blobs. For example VISOR_S1 contains almost a 40% of static blobs (177/443) and it is still significantly faster than previous detector.

5.4.2.2 Ground Truth

As mentioned above, in this case the background extraction is not performed, so we can not evaluate the global system performance. Even so we can see how the proposed algorithm is much faster than the previous detector. As in the previous case, the version with the analysis of static objects has a higher cost, but it is still smaller than the previous algorithm. For example CVSG_S3 with a 70% of static blobs (1626/2406) is still faster than previous detector.

	Algorithm	Fps	ms/blob	Time (ms)	Background (%)	Extraction (%)	Tracking (%)	Static (%)	People (%)	Extra (%)
Hall monitor	Previous	19.72	19.78	15213.30	33.17	1.26	1.08	0.00	63.85	0.64
	Fusion Edge	40.42	4.00	7421.61	66.22	2.42	2.42	0.00	26.46	2.48
	Fusion Edge+Static	33.98	5.88	8828.24	53.45	2.06	1.80	7.56	32.7	2.43
VISOR S1	Previous	25.00	29.48	21363.32	36.06	1.30	0.91	0.00	61.13	0.6
	Fusion Edge	50.82	4.15	10508.58	75.17	2.63	1.94	0.00	17.51	2.75
	Fusion Edge+Static	34.47	13.81	15490.13	48.59	1.76	1.28	6.79	39.49	2.08
PETS S1	Previous	5.79	28.28	210734.44	46.48	1.53	1.62	0.00	49.31	1.06
	Fusion Edge	10.23	4.25	119345.27	79.62	2.62	2.78	0.00	13.09	1.89
	Fusion Edge+Static	8.54	7.39	142919.06	65.09	2.17	2.32	9.86	19.00	1.57

(a) Background Extraction Sequences Computational Cost

	Algorithm	Fps	ms/blob	Time (ms)	Background (%)	Extraction (%)	Tracking (%)	Static (%)	People (%)	Extra (%)
CVSG S1	Previous	18.06	29.03	10685.79	0.00	1.01	0.94	0.00	97.55	0.5
	Fusion Edge	114.62	3.91	1683.77	0.00	6.77	6.49	0.00	83.36	3.38
	Fusion Edge+Static	32.27	14.07	5980.59	0.00	1.81	1.72	10.38	84.45	1.64
CVSG S2	Previous	10.07	55.37	34647.60	0.00	0.59	0.54	0.00	98.61	0.27
	Fusion Edge	108.92	4.28	3204.17	0.00	6.56	5.96	0.00	82.38	5.09
	Fusion Edge+Static	56.39	8.00	6189.51	0.00	3.45	3.10	11.37	79.71	2.37
CVSG S3	Previous	16.23	22.11	55267.63	0.00	0.95	0.89	0.00	96.25	1.91
	Fusion Edge	67.02	4.30	13384.06	0.00	4.89	5.69	0.00	86.10	3.32
	Fusion Edge+Static	19.46	18.05	46091.12	0.00	1.11	1.05	2.98	94.23	0.63

(b) Ground Truth Sequences Computational Cost

Table 5.3: Computational Cost

Chapter 6

Conclusions and future work

6.1 Summary of Work

In this document, three main contributions have been presented:

- **A people detection algorithm based on people appearance as discriminative information**. We have designed and implemented a people detection algorithm based on appearance. The main idea consists of identifying characteristic edges of each body part and generate four edge body parts models. In order to get a fast algorithm but getting good precision result, we only process detected objects during previous stages and we have simplified the training process.

- **Improving people detection by using motion information**. In order to add robustness to the detection, we have designed two approaches for using motion information through a tracking algorithm. The fist approach consists of a simple weighted decision over time and the second approach consists of performing a more exhaustive classification over static objects.

- **The design and annotation of a people video/image dataset.** The proposed algorithm consists of four edge body parts models. Each model has to be trained with an image collection with people and no people examples, and therefore we have designed a complete image dataset composed of several images from public datasets. We have grouped images into train and test image sets. Moreover, each positive example has been segmented into four body parts. Besides, in order to provide a good performance evaluation of the proposed framework and a comparison with previous work, we have designed a dataset composed of several sequences from public datasets. We have grouped the test sequences into six different complexity categories depending on the background extraction and scene complexity. Then we have annotated these sequences using the Viper-GT tool [31] describing each object in terms of its location in space and time, and people classification. Finally, the dataset and its ground-truth are used to evaluate our algorithm.

6.2 Conclusions

The work presented in this document is focused on the improvement of people detection task in video surveillance. We have designed and implemented a people detection algorithm based on appearance and motion as discriminative information. Firstly, we have undertaken a study on the state of the art in people detection algorithms targeted to video surveillance scenarios, and we have made a classification of the different algorithms studied. Within this classification, algorithms based on appearance use to obtain better results than motion based algorithms, and therefore we have designed and implemented an algorithm based on appearance but adding robustness to the detection with motion information.

We have evaluated people detection results on several public video sequences manually annotated (see chapter 5). Two different training approaches (a monolithic classifier and a cascade of classifiers) have been evaluated, we have compared the fusion detector and each body part detector independently, our proposal with previous detector have been compared, and the improving people detection by using motion information has been evaluated.

Some conclusions have been extracted: The cascade approach works better than monolithic approach. Our proposed algorithm is more robust (body parts fusion), stable and faster than the previous algorithm. Although people classification over time does not provide significant improvements, the analysis of static object improves the results considerably.

Therefore, the main objective of this work has been completed: a new people detector have been implemented and integrated into the VPULab system. This new algorithm obtains more reliable results and less computational cost than the previous algorithm.

6.3 Future Work

The work described in this document is a step in the direction of the combination of people appearance and motion information to improve the results obtained in the people detection task. The proposed problem (people detection) is certainly not fully solved: the state of art in people detection and tracking includes several successful solutions only working in specific and constrained scenarios. We identify four main areas for future work:

- **Improvement of analysis components.** In the proposed framework, the background extraction process fails in some cases (shadows, reflections, noise, etc), and therefore the object extraction does not work properly. Thus, we propose to study new algorithms and post-processing techniques to increase the performance of the object extraction process. The object extraction process is one of the critical points in any visual analysis system. We propose the study of techniques for noise removal, shadows detection, etc, in order to refine the background extraction.

- **Study of more robust tracking systems.** In the state of the art, several robust trackers have been developed which can track objects in real time in simple scenarios. However, it is clear that the assumptions used to make the tracking problem tractable (e.g., smoothness of motion, minimal amount of occlusion, illumination constancy, high contrast with respect to background) are violated in many realistic scenarios. In particular, in our case we propose to study tracking algorithms which are more robust to occlusions. And that way, we could use the motion information better.

- **Improvement of body parts detectors.** We propose to improve each body part detector. We propose to eliminate the restriction that the person has to be in standing position, for example by a previous analysis of the pose. We propose not to assume that a blob is a whole person, but also a unique part in order to support larger occlusions. For this reason the sequences annotations include the attribute "IsPerson" and also the attribute "BodyPart".

- **Motion information.** We intend to explore in more depth the use of motion information, for example flow pattern, periodic motion, etc. Motion could be considered either complementary information or essential in specific scenarios where methods based on appearance do not work.

Bibliography

[1] H. Sidenbladh, "Detecting human motion with support vector machines," in *International Conference on Pattern Recognition (ICPR)*, 2004.

[2] R. Cutler and L. S. Davis, "Robust real-time periodic motion detection, analysis, and applications," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22(8), pp. 781–796, 2000.

[3] F. Xu and K. Fujimura, "Human detection using depth and gray images," in *Conference on Advanced Video and Signal Base Surveillance (AVSS)*, pp. 115–121, 2003.

[4] P. Kilambi, E. Ribnick, A. J. Joshi, O. Masoud, and N. Papanikolopoulos, "Estimating pedestrian counts in groups," in *Computer Vision and Image Understarding*, vol. 110 of *2008*, pp. 43–59, 2008.

[5] J. Zhou and J. Hoang, "Real time robust human detection and tracking system," in *Computer Vision and Patter Recognition (CVPR)*, p. 149, 2005.

[6] I. Haritaoglu, D. Harwood, and L. S. Davis, "Ghost: A human body part labelling system using silhouettes," in *International Conference on Pattern Recognition (ICPR)*, pp. 77–82, 1998.

[7] M. Valera and S. A. Velastin, "Intelligent distributed surveillance systems: a review," in *IEE Proceedings on Visual Image Signal Processing*, vol. 152, pp. 192–204, 8 April 2005.

[8] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," in *IEEE Transactions on Systems, Man and Cybernetics*, vol. 34, pp. 334–352, 2004.

[9] M. Piccardi, "Background subtraction techniques: a review," in *Proc. IEEE Intl. Conf. on Systems, Man and Cybernetics*, 2004.

[10] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," in *ACM*, vol. 38, p. 13, 2006.

[11] H. Drucker, C. J. Burges, L. Kaufman, C. J. C, B. L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems*, pp. 155–161, 1997.

[12] V. Fernández-Carbajales, M. A. García, and J. M. Martínez, "Robust people detection by fusion of evidence from multiple methods," in *WIAMIS '08: Proceedings of the 2008 Ninth International Workshop on Image Analysis for Multimedia Interactive Services*, pp. 55–58, 2008.

[13] M. Hussein, W. Abd-Almageed, Y. Ran, and L. Davis, "Real-time human detection, tracking, and verification in uncontrolled camera motion environments," in *International Conference on Vision Systems (ICVS)*, p. 41, 2006.

[14] N. Koenig, "Toward real-time human detection and tracking in diverse environments," in *International Conference on Development and Learning (ICDL)*, pp. 94–98, 2007.

[15] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detector," in *International Conference on Computer Vision (ICCV)*, pp. 90–97., 2005.

[16] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: two new techniques for image matching," in *International Joint Conference on Artificial Intelligence*, 1977.

[17] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *International Conference on Computer Vision (ICCV)*, 2003.

[18] X. Cui, Y. Liu, S. Shan, X. Chen, and W. Gao, "3d haar-like features for pedestrian detection," in *International Conference on Multimedia & Expo (ICME)*, pp. 1263–1266, 2007.

[19] N. Sprague and J. Luo, "Clothed people detection in still images," in *International Conference on Pattern Recognition (ICPR)*, 2002.

[20] S. Harasse, L. Bonnaud, and M. Desvignes, "Human model for people detection in dynamic scenes," in *Computer Vision on Pattern Recognition (CVPR)*, pp. 335–354, 2006.

[21] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," in *International journal of computer vision*, vol. 77 of *1-3*, pp. 157–173, 2008.

[22] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1, pp. 886–893, 2005.

[23] H. Freeman, "Computer processing of line-drawing images," in *Computing Surveys*, vol. 6, pp. 57–97, 1974.

[24] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," in *Machine Learning*, vol. 37, pp. 297–336, 1999.

[25] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition*, vol. 1, pp. 511–518, 2001.

[26] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*, pp. 23–37, 1995.

[27] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting,” in *The Annals of Statistics*, vol. 2, pp. 337–374, April 2000.

[28] A. Vezhnevets and V. Vezhnevets, “Modest adaboost :teaching adaboost to generalize better,” in *Graphicon*, 2005.

[29] R. Kalman, “A new approach to linear filtering and prediction problems,” in *Journal of Basic Engineering*, vol. 1, pp. 35–45, 1960.

[30] G. Welch and G. Bishop, “An introduction to the kalman filter,” 1995.

[31] D. Doermann and D. Mihalcik, “Tools and techniques for video performances evaluation,” in *IEEE Int. Conference on Pattern Recognition*, vol. 1(1), pp. 167–1709, 2000.

XList of Abbreviations

BG	BackGround
Blob	Binary Large Object
CCA	Connected Component Analysis
FG	ForeGround
GM	Gaussian Model
GMM	Gaussian Mixture Model
HTPP	HyperText Transport Protocol
MoG	Mixture of Gaussians
SVM	Support Vector Machine
URL	Universal Resource Locator
VPULab	Video Processing Understanding Lab
XML	eXtensible Markup Language

Appendix A

GML AdaBoost Matlab Toolbox

GML AdaBoost Matlab Toolbox `http://graphics.cs.msu.ru/en/science/research/machinelearning/adaboosttoolbox` is a set of Matlab functions and classes implementing a family of classification algorithms, known as Boosting. This toolbox has three different boosting schemes: Real AdaBoost, Gentle AdaBoost and Modest AdaBoost.

Real AdaBoost is the generalization of a basic AdaBoost algorithm first introduced by Schapire [24]. Real AdaBoost should be treated as a basic "hardcore" boosting algorithm.

Gentle AdaBoost is a more robust and stable version of real AdaBoost [27]. So far, it has been the most practically efficient boosting algorithm, used, for example, in the Viola-Jones object detector [25]. Authors' experiments show, that Gentle AdaBoost performs slightly better than Real AdaBoost on regular data, but it is considerably better on noisy data, and much more resistant to outliers.

Modest AdaBoost [28] is mostly aimed for a better generalization capability and resistance to overfitting. Authors' experiments show, that in terms of test error and overfitting this algorithm outperforms both Real and Gentle AdaBoost.

A.1 Functions and Classes

In this section we describe some Matlab functions and C++ classes used during this work.

A.1.1 GML Functions and Classes

In this section we describe some Matlab functions or methods and C++ classes implemented in the GML AdaBoost Matlab Toolbox.

RealAdaBoost/GentleAdaBoost/ModestAdaBoost *function [Learners, Weights, {final_hyp}] = RealAdaBoost/GentleAdaBoost/ModestAdaBoost (WeakLrn, Data, Labels, Max_Iter, {OldW, OldLrn, final_hyp})*

Matlab function to boost a weak learner *WeakLrn* using *Real AdaBoost/Gentle Adaboost/Modest Adaboost* algorithm with *Max_Iter* iterations on dataset given in *Data* and *Labels.*

- Arguments: *WeakLrn* - weak learner. *Data* - training data. Should be DxN matrix, where D is the dimensionality of data, and N is the number of training samples. *Labels* - training labels. Should be 1xN matrix, where N is the number of training samples, any label is either +1 or –1. *Max_Iter* - number of iterations. *OldW* - weights of already built committee (used for further training of already built commitee. Optional parameter). *OldLrn* - learners of already built committee (used for further training of already built committee. Optional parameter). *final_hyp* - output for training data of already built committee (used to speed up further training of already built committee. Optional parameter).

- Return: *Learners* - cell array of constructed learners. *Weights* - weights of learners. This vector has the same size as *Learners* and represents weight of each learner in final committee. *final_hyp* - output for training data.

Classify *function Result = Classify (Learners, Weights, Data)*

Matlab function to classify *Data* using boosted assembly of *Learners* with respective *Weights.* Result will contain real numbers; the signum of those numbers represents the class, and its absolute magnitude is the "confidence" of the decision. To obtain classification one should take signum of *Result.* To regulate the rate of false positive / false negative *Result* could be compared with some threshold. Increasing threshold would reduce false positive rate, but will also increase false negative.

- Arguments: *Learners* - cell array of constructed learners. *Weights* - weights of learners. *Data* - training data.

- Return: *Result - Data prediction.*

TranslateToC *function code = TranslateToC (Learners, Weights, fid)*

Matlab function to save constructed classifiers for further use in C++ applications. File has the following format:

<TN>

<W> <N > <D> <T> <Ts> {<D> <T> <Ts>}

<W> <N > <D> <T> <Ts> {<D> <T> <Ts>}

...

<end>

Where: TN – total number of weak classifiers. W – weight of weak classifier. N – number of thresholds representing weak classifier. D – thresholds dimension. T – threshold value. Ts – threshold sing; it's either –1 or +1, resembling if the sample should be greater or lesser than threshold to be classified positive.

- Arguments: *Learners* - learners of committee to be saved. *Weights* - weights of committee to be saved. *fid* - opened file id (use *fopen* to make one).
- Return: *code* - equals 1 if everything was alright.

CBoostedCommittee C++ class to load constructed classifiers previously saved using "TranslateToC" function. It contains the attributes that define a classifier: Learners (thresholds dimension, threshold value and threshold sing) and Weights. This class also includes the method which loads constructed classifiers from a file "LoadFromFile" and the method which predicts a sample "Predict".

A.1.2 Functions and Classes Implemented

In this section we describe some Matlab functions or methods and C++ classes implemented in this work.

ClassifyCascade *function Result = ClassifyCascade(Learners, Weights, Thresholds, Data)*

Matlab function implemented in this work to classify *Data* using constructed cascade classifiers (*Learners*, *Weights* and *Thresholds*). Result will contain real numbers; the signum of those numbers represents the class, and its absolute magnitude is the "confidence" of the decision. To obtain classification one should take signum of *Result.* This method is valid for simple constructed classifiers or cascade structures.

- Arguments: *Learners* - array of learners of each cascade stage. *Weights* - array of weights of each cascade stage. *Thresholds* - thresholds of each cascade stage. *Data* - training data.
- Return: *Result - Data prediction.*

WriteWeaksClasifiersCascade *function code=WriteWeaksClasifiers (Learners, Weights, fid, height, width, num_edges)*

Matlab function implemented in this work to save the list of classifiers used in the global classifier (edgelet's number and edgelet's position, see equation 3.8). This method is necessary because during the training phase we generate all possible classifiers but during the running phase we only need the classifiers chosen for the model. This method is valid for simple constructed classifiers or cascade structures.

- Arguments: *Learners* - array of learners of each cascade stage to be saved. *Weights* - array of weights of each cascade stage to be saved. *fid* - opened file id (use *fopen* to make one). *height* - height of body part model. *width* - width of body part model. *num_edges* - number of edgelets used (775 or 100 using top-100 ranking).
- Return: *code* - equals 1 if everything was alright.

TranslateToCCascade *function code = TranslateToCCascade (Learners, Weights, fid)*

Matlab function implemented in this work to save constructed cascade classifiers for further use in C++ applications. This method is valid for simple constructed classifiers or cascade structures.

- Arguments: *Learners* - array of learners of each cascade stage to be saved. *Weights* - array of weights of each cascade stage to be saved. *fid* - opened file id (use *fopen* to make one).
- Return: *code* - equals 1 if everything was alright.

CBoostedCommitteeCascade C++ class implemented in this work to load constructed cascade classifiers previously saved using "TranslateToCCascade" function. It contains the attributes that define a cascade classifier: the number of stages and the monolithic classifiers of each stage (CBoostedCommittee). This class also includes the method which loads constructed cascade classifiers from a file "LoadFromFileCascade" and the method which predicts a sample "PredictCascade".

Appendix B

C++ classes

In this section we describe some C++ classes used or created during this work.

B.1 VPULab Classes

In this section we describe some classes already implemented in VPULab system.

BasicBlob Class to describe a basic blob. It contains three main private attributes: An integer "FrameStart" to store the number of frame when appeared this blob, an OpenCV CvBlob "blob" (to store width, height, x and y position and ID of blob) and a string "format" to store extra information about blob's format.

PeopleBlob Class to describe a blob with information of people classification. It is an extension of BasicBlob class, it also includes five new attributes: Two OpenCV images IplImage ("image" and "mask" cropped from original frame and foreground/background mask) of the blob, two integers "centerX" and "centerY" with information about the blob's center coordinate and a score attribute which will include the evidence found during the detection process.

ObjectBlob Class to describe a static object or people with information to describe if it is stolen or abandoned object or people. It is an extension of BasicBlob class; it also includes seven new attributes: An integer "count" with the life count of the blob, an integer "staticCount" with the static life count of the blob, an integer "lastFrame" with the last frame of the static life count of the blob, a boolean "act" with a flag of used blob or not, a boolean "peopleLikelihoodComputed" with a flag of people classification computed or not, a double "peopleLikelihood" with the likelihood of being people and an integer “numberOfCheckingSteps” with the number of times that the checking process (check if it is stolen or abandoned) has been performed.

BlobList Class to describe a list of blobs. It contains a template “pBlob” with a list of any kind blobs.

PeopleDetector Class to describe detectors of people. It contains four main private attributes: A BlobList of PeopleBlobs “listNewBlobs” to store the input blobs, a BlobList of PeopleBlobs “listBlobs” to store the blobs to be processed, a BlobList of ObjectBlobs “outputListBlobs” to store the output blobs with the final likelihood of being people and an IplImage “resultImage” to represent the people classification results.

The PeopleDetector class also defines two main methods: “PeopleLikelihoodOfBlobs” and “checkObject”. The first method must be implemented to process the BlobLists of PeopleBlobs “listBlobs” and the second method must be implemented to process a single PeopleBlob.

B.2 Classes Implemented

In this section we describe some C++ classes implemented in this work.

EdgePeopleDetector Class to describe detector based on edges. It is an extension of PeopleDetector class; it also includes the attributes required for people detection based on edges, the most important are: An integer “body_part” with the body part identification and a string “Dir” with the directory where we can find all you need to realize the detection (trained models and learned edge probability distributions).

The EdgePeopleDetector class also includes the implementation of methods “PeopleLikelihoodOfBlobs” and “checkObject”. The first method, at each iteration, updates the list of blobs to be processed “listBlobs” basing on the ID of the input list of blobs “listNewBlobs” and then each blob is processed with the method “checkObject”. This second method executes the people detection stages: blob’s image segmentation, edge features extraction, weak classifiers generation and people classification.

FusionEdgePeopleDetector Class to describe people detector based on fusion of body parts detectors (EdgePeopleDetector). It is an extension of PeopleDetector class; it also includes five new attributes: A double “cutValue” with the decision threshold (if the evidence is greater than the threshold the blob is classified as a person) and four EdgePeopleDetectors “bodyAlg, headAlg, torsoAlg and legsAlg” (a detector for each body part).

The FusionEdgePeopleDetector class also includes the implementation of methods “PeopleLikelihoodOfBlobs”, “checkObject” and “calculateFusionEvidence”. The first method, at each iteration, updates the list of blobs to be processed “listBlobs” basing on the ID of the input list of blobs “listNewBlobs”, and then each blob is processed four times (one for each body part) with the method “checkObject” and finally performs the final classification as a fusion of the four evidences “calculateFusionEvidence”.

Edge Class to describe an edgelet (see section 3.3.3.1). It contains information of length, angles and movements associated with those angles: An integer with the “length” and two array of integers with information of “angles” and “movements”. This class also includes the method which performs the dot product approximation between two normal vectors: “EdgeAngleMultNorm” (see equation 3.2).

EdgeFeature Class to describe an edge feature (see section 3.3.3.2). An edge feature is defined with three attributes: Two integers “height” and “width”, and an IplImage “data” according to equation 3.3.

EdgeFeatureExtractor Class to implement the edge features extraction. It contains the images (IplImage) necessary for the generation of each edge feature (see section 3.3.3.2): Sobel intensity, normal vector and orientation quantization. It also includes the method which extracts all the edge features and stores it during the training phase: “generateEdgeFeatures” and the method which extracts only the necessary edge features in order to generate the classification model during the running phase: “generateEdgeFeature”.

WeakClassifier Class to describe an weak classifier (see section 3.3.3.3). An edge feature is defined with three attributes: Two integers “height” and “width”, and an IplImage “data” according to equation 3.4.

WeakClassifierExtractor Class to implement the weak classifiers extraction. It contains the images (IplImage) necessary for the generation of each weak classifier (see section 3.3.3.3): epsilon image, logarithm of the image, etc. It includes the method which extracts all the weak classifiers and stores it during the training phase: “generateWeakClassifiers” and the method which extracts only the necessary weaks classifiers in order to generate the classification model during the running phase:

“generateWeakClassifier”. This class also includes the method which performs the B function: “functionB” (see equation 3.5).

Appendix C

ViPER Ground Truth Tool

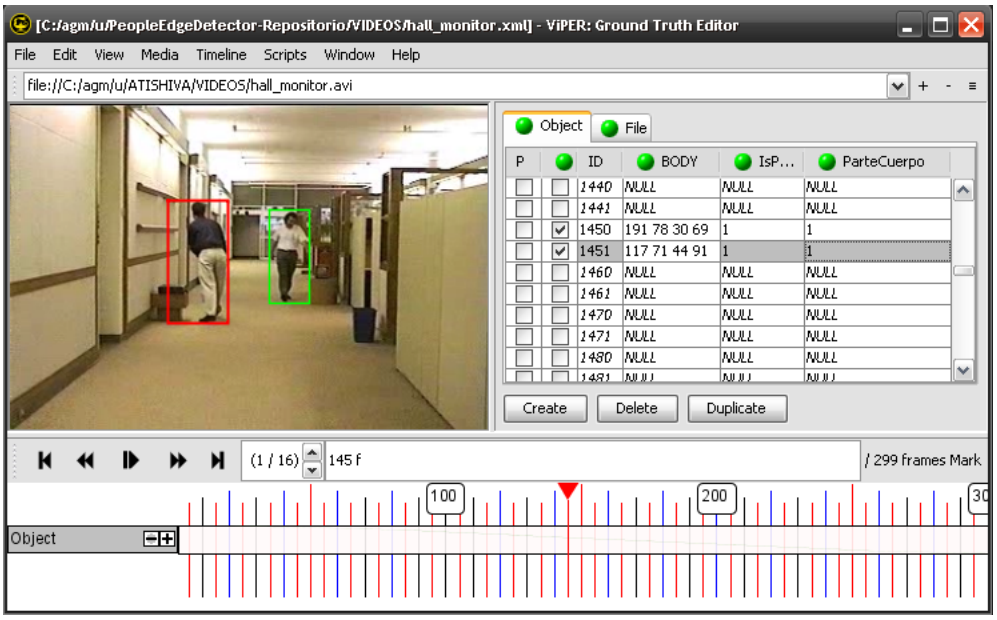
In order to evaluate a video analysis algorithm, or a set of algorithms, it is necessary to define a methodology. For video processing, it is very common for a human to define the ground truth for each video clip. In order to ensure that researchers may repeat and verify evaluations, it is important to make the ground truth metadata is available to other researchers in a documented format. It is very useful to have methods of qualitatively verifying the ground truth, as well, with metadata browsers and editors. ViPER-GT provides tools for creation and editing video metadata. ViPER `http://viper-toolkit.sourceforge.net` also provides a general framework for evaluation. In this appendix, we focus on ground truth annotation because the evaluation task is only the number of people blobs correctly or incorrectly detected.

The Video Performance Evaluation Resource Kit's Ground Truth tool, or ViPER-GT, allows someone to annotate a video with metadata, mainly for use as ground truth for performance evaluation. This includes information describing the file, such as date of filming and keywords about its content. It also includes concrete features, such as scene breaks and bounding boxes around people.

C.1 ViPER-GT Interface

ViPER-GT is the tool for creating and editing metadata using a Java graphical user interface. It is designed to allow frame-by-frame markup of video metadata stored in the Viper format. It is also useful for visualization. For more information, see the ViPER-GT product page `http://viper-toolkit.sourceforge.net/products/gt/`.

In Figure C.1 we can see an example of the ViPER-GT user interface. At the top of the frame is a pull-down menu that shows the name of the currently loaded video file; this panel, the source media selector, also allows the user to edit the list of described media files. The video frame view is in the upper-left quadrant of the screen; this displays the video with spatial annotations. To the right of the video frame is the spreadsheet view, which displays the annotations as a table. Beneath these two views of the data is the time line view, which displays summary of the video annotation, indicating when descriptors are marked as valid.

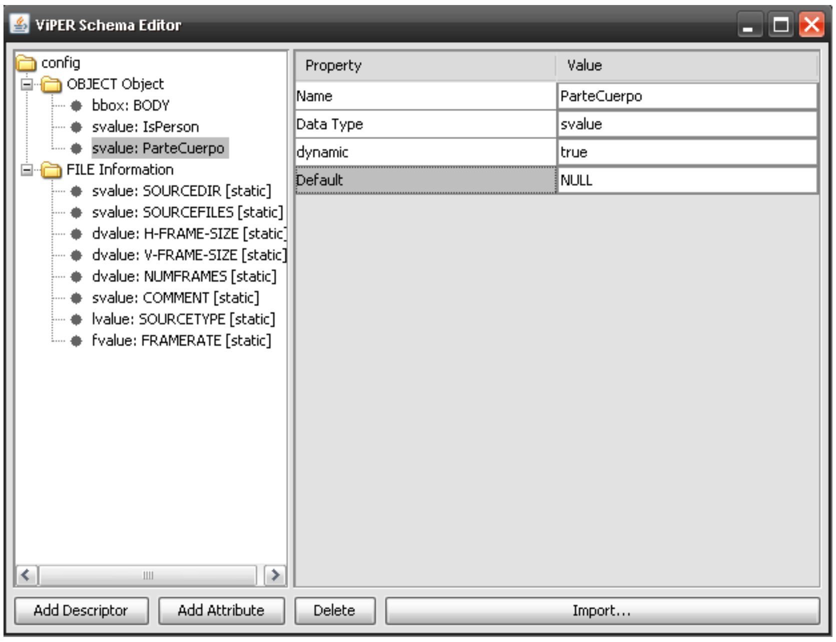

[C:/agm/u/PeopleEdgeDetector-Repositorio/VIDEOS/hall_monitor.xml] - ViPER: Ground Truth Editor
File Edit View Media Timeline Scripts Window Help
file://C:/agm/u/ATISHIVA/VIDEOS/hall_monitor.avi
Object
File
Create
Delete
Duplicate
(1 / 16)
145 f
/ 299 frames Mark
100
200
30
Object

P	●	ID	● BODY	● IsP...	● ParteCuerpo
☐	☐	1440	NULL	NULL	NULL
☐	☐	1441	NULL	NULL	NULL
☐	✓	1450	191 78 30 69	1	1
☐	✓	1451	117 71 44 91	1	1
☐	☐	1460	NULL	NULL	NULL
☐	☐	1461	NULL	NULL	NULL
☐	☐	1470	NULL	NULL	NULL
☐	☐	1471	NULL	NULL	NULL
☐	☐	1480	NULL	NULL	NULL
☐	☐	1481	NULL	NULL	NULL

Figure C.1: ViPER-GT Interface

C.2 Creating Schema Descriptions

ViPER-GT also provides a graphical tool to create descriptions. A descriptor can be: a record describing some element of the video, an object that conforms to a user defined schema and it is composed of attributes. We have different types of descriptors (*File, Content,* and *Object). File* descriptor reflects the video as a whole, or other metadata about the video, such as file format and frame rate. *Content* descriptions describe metadata that may only occur one at a time. Each instance of this type has a time span and a set of attributes. *Object* description refers to an object that may have many instances at any given time, and whose instances may change over time. (e.g., the events to detect). In Figure C.2 we can see an example of the interface.



```
config
OBJECT Object
    bbox: BODY
    svalue: IsPerson
    svalue: ParteCuerpo
FILE Information
    svalue: SOURCEDIR [static]
    svalue: SOURCEFILES [static]
    dvalue: H-FRAME-SIZE [static]
    dvalue: V-FRAME-SIZE [static]
    dvalue: NUMFRAMES [static]
    svalue: COMMENT [static]
    lvalue: SOURCETYPE [static]
    fvalue: FRAMERATE [static]
```

Property	Value
Name	ParteCuerpo
Data Type	svalue
dynamic	true
Default	NULL

Figure C.2: ViperGT Schema

C.2.1 Descriptor Designed for this Work

In Figure C.3 we can see the descriptor for representing the blobs annotated in the ground truth and detected by the proposed framework. This descriptor is composed of three parts: a bounding box (Blob) with the blob's area, a boolean people classification (IsPerson) and a body part identification (BodyPart: 0-Body, 1-Head, 2-Torso and 3-Legs).

```
- <descriptor name="Object" type="OBJECT">
    <attribute dynamic="true" name="Blob" type="http://lamp.cfar.umd.edu/viperdata#bbox" />
    <attribute dynamic="true" name="IsPerson" type="http://lamp.cfar.umd.edu/viperdata#svalue" />
    <attribute dynamic="true" name="BodyPart" type="http://lamp.cfar.umd.edu/viperdata#svalue" />
  </descriptor>
```

Figure C.3: Object Descriptor Designed