Department of Electrical and Computer Engineering
Faculty of Sciences and Technology
University of Coimbra

Traffic Video Surveillance for Automatic Incident Detection on Highways

Gonçalo Luis Mateus Vaz Monteiro
ggmonteiro@gmail.com

Submitted in partial fulfillment of the
requirements for the degree of Master of Science

November 30, 2008

Supervisor: Professor Jorge Batista
Department of Electrical and Computer Engineering
University of Coimbra

Abstract

In order to ensure a safe and efficient driving, it is important to classify the behaviors of the vehicles and to understand their interactions in typical traffic scenarios. Not long ago, this burdensome task was performed only by human operators at traffic control centers. However, the increasing number of available cameras dictated the need for automatic traffic surveillance systems. In this thesis, one specific roadway safety system will be addressed, focusing on the monitoring of drivers' behaviors through the analysis of the CCTV signal.

The presented work focus on the detection of dangerous behaviors in highways in order to improve road safety. Two distinct solutions are proposed to detect two of the most dangerous situations in highways: wrong way vehicles and vehicles stopped on the road or on the hard shoulder. An immediate detection of these dangerous behaviors could help prevent serious accidents by warning the oncoming vehicles and the police. The proposed system aims at automatically detecting these events, triggering an alarm on the highway traffic telematic system, and, consequently, warning the human operator, who will handle the situation.

In order to detect the vehicles in the image a segmentation process for outdoor scenarios based on background subtraction is presented. The proposed system aims at robustly adapting itself to lighting variations and to work twenty four hours a day. For this purpose, three different background models and dynamic thresholds update are used. It was also presented a shadow/highlight detection algorithm based on normalized color cross-correlation to discard shadow areas and blobs generated by lighting variations (moving clouds, artificial light changes, etc.).

The stopped vehicle detection system proposed on this thesis has three main

phases. First, the segmentation process previously referred is used to identify all the vehicles in the scene. Second, it is verified if there are *static* pixels, i.e., pixels segmented during a certain period of time with the same color. A pixel color history cache analysis is used to identify the static pixels. Finally, a temporal validation is applied to the blobs formed with those static pixels. If the validation succeeds, an alarm is triggered in the traffic telematic system.

The wrong way vehicle detection system proposed on this thesis uses information provided by an optical flow algorithm to estimate the motion direction of the vehicles. Firstly, the orientation pattern of the vehicle's motion flow is modeled by a mixture of Gaussians. Then, there is a Detection Phase: it is verified, in each frame, if the motion direction of the vehicle matches the learned motion direction model. On both phases, a Block Median Filtering is applied to remove noisy flow vectors. Finally, a temporal validation is used to validate the wrong way vehicle. If the validation succeeds, an alarm is triggered in the traffic telematic system.

Several tests were performed on the proposed solutions in order to validate its effectiveness. Tests performed in public datasets and several datasets from Portuguese highways with different weather conditions and illumination changes proved the robustness and the accuracy of the presented methodologies in detecting anomalous situations in highways. Several tests were also performed on site in some Portuguese highways during several months and good results were obtained.

A real automatic traffic surveillance system was developed using the research work presented on this thesis. This automatic traffic surveillance system is currently being used in several Portuguese highways to automatically detect vehicles circulating on the wrong direction and vehicles that pull over on the road or on the hard shoulder.

Acknowledgements

To begin with, I would like to thank some people for the support they provided, without which I would not have been able to complete this work.

First of all, I would like to address a word of appreciation to Professor Jorge Batista, the supervisor of this dissertation, not only for having suggested the topic of this work but also for his expertise, technical support, useful comments, which were essential throughout all stages of the project and helped me improve my work.

I must also thank all my colleagues at the Institute of Systems and Robotics for the exchange of ideas and knowledge. Moreover, I would like to thank all my friends for the encouragement and for the help at the most difficult moments of this project.

I would like to address a very special word of appreciation to my family - to my parents José Monteiro and Maria Vaz - for having always provided me with all the types of support a son can wish.

And, finally, I would like to thank my girlfriend Cristina for helping me revise the previous versions of the thesis and, above all, for her incredible patience and endless support during these years of our life.

Thank you to you all!

Contents

1 Introduction **18**

1.1 Thesis Overview 21

I Robust Outdoor Vehicle Detection **24**

2 Foreground/Background Segmentation **26**

2.1 Introduction 26

2.1.1 Related Work 28

2.2 Segmentation Process 39

2.2.1 Background Modeling 40

2.2.2 Online Background Model Update 42

2.2.3 Pixel-level Analysis 46

2.2.4 Region-level Analysis 48

2.2.5 Frame-level Analysis 50

2.2.6 System Thresholds 51

2.3 Shadow and Highlight Detection 56

2.3.1 Shadow Detection Algorithms 57

2.3.2 Experimental Results 64

2.4 Image Flow Estimation 68

2.4.1 Optical Flow Estimation 69

2.4.2 Features to Track Selection 71

2.4.3 Experimental Results and Conclusions 72

2.5 Experimental Results 75

2.5.1 Segmentation Process Robustness and Accuracy 76

2.6 Conclusions 78

II Automatic Incident Detection 84

3 Wrong Way Vehicle Detection 86

3.1 Introduction 86

3.1.1 Related Work 87

3.2 Proposed Methodology 87

3.3 Vehicle's Motion Estimation 88

3.3.1 Block Median Filtering 88

3.4 Traffic Flow Direction Learning 90

3.5 Wrong Way Vehicle Detection 93

3.5.1 Temporal Validation 94

3.6 Experimental Results 95

3.6.1 Tests Performed On-Site 95

3.6.2 Tests Performed in Datasets of Highway Scenarios 97

3.7 Conclusions 98

4 Stopped Vehicle Detection 106

4.1 Introduction 106

4.1.1 Related Work 107

4.2 Proposed Methodology 108

4.3 Static Pixels Identification 109

4.3.1 Cache Update 110

4.3.2 Static Pixels Detection 111

4.4 Stopped Vehicle Validation 112

4.5 Experimental Results 113

4.5.1 Tests Performed in Public Datasets 113

4.5.2 Tests Performed On-Site 114

4.6 Conclusions 116

III Automatic Traffic Surveillance System 122

5 AVISAR Project 124

5.1 Introduction 124
5.2 System Description 125
5.3 System Integration 128
5.4 Scene Masks 128
5.5 Online Analysis 131

IV Final Notes **134**

6 Conclusion **136**
6.1 Achieved Objectives 136
6.2 Publications 138

List of Tables

2.1 Comparative results of the tests performed with texture-based algorithm and the color-based algorithm to detect shadowed pixels. 67
2.2 Composition of the challenging datasets available at VSSN 2006 website and used to validate the proposed segmentation algorithm. 77

3.1 Performance of the proposed wrong way vehicle detection system in some surveillance cameras in Portuguese highways. 97
3.2 Datasets used to validate the proposed wrong way vehicle detection system. 103
3.3 Performance of the proposed wrong way vehicle detection system in test datasets in Portuguese highways. 104

4.1 Public datasets used to validate the stopped vehicle detection system. 115
4.2 Performance of the proposed stopped vehicle detection system in some public datasets. 116
4.3 Scenarios used to perform the on-site tests to the proposed stopped vehicle detection system. 117
4.4 Performance of the proposed stopped vehicle detection system tested on-site in some outdoor scenarios. 118

List of Figures

1.1 Traffic Control Center at Carcavelos, Portugal. 20

2.1 Challenging situations for a segmentation process. 29
2.2 Adopted color model for segmentation. 35
2.3 Flowchart of the proposed segmentation process – pixel-level analysis. 40
2.4 Initial background model estimation. 43
2.5 Initial background model estimation with traffic jam. 44
2.6 Background model adaptation over time. 47
2.7 A sudden illumination change in a tunnel scenario. 51
2.8 Three consecutive frames of an sudden lighting change in a tunnel scenario. 51
2.9 The camera's AGC changes the image brightness because of the truck that is coming towards the camera. 52
2.10 Segmentation process in a typical outdoor scenario. 53
2.11 Example of a shadow problem in an outdoor scenario, the vehicle's shape and area change significantly. a) Bounding box of the vehicles detected with the proposed segmentation system. b) Foreground pixels detected with the proposed segmentation system (green pixels). 57
2.12 The geometric relationship of a moving cast shadow. 58
2.13 The conical representation of the HSV color space. 62
2.14 The representation of the chromatic plane of the hsL color space. 63
2.15 Scenarios used to compare the texture-based algorithm with the color-based algorithm to detect shadowed pixels. 65

2.16 Shadow detection in outdoor scenarios. Left column – Captured frame; Middle column – Color-Based shadow detection; Right column – Texture-Based shadow detection. The shadowed pixels are identified with the red color. 66
2.17 Sudden illumination change detection in a tunnel scenario. a) two consecutive frames of the sudden illumination change; c) Color-Based shadow detection algorithm; d) Texture-Based shadow detection algorithm. 67
2.18 Optical flow estimation with the presented methodology. 74
2.19 Segmentation process in a typical outdoor scenario. 75
2.20 Segmentation process in a typical outdoor scenario. 79
2.21 Segmentation process in a typical outdoor scenario. 79
2.22 Segmentation process in a tunnel scenario. 80
2.23 Segmentation process in a rainy situation. 80
2.24 Segmentation process with rain drops in the camera's lens 81
2.25 Comparative results of the proposed segmentation process. . . . 82

3.1 Results of the Block Median Filtering to reduce the disturbance in the optical flow estimation. a), c) e), and g) output of the estimated optical flow. b), d), f), and h) result of the block median filtering. Note that in h) not all the wrongly detected vectors are filtered. 89
3.2 Flowchart of the traffic flow direction learning process. 90
3.3 Traffic flow direction learning process: a), b) and c) show the evolution of the orientation pattern modeled by the first Gaussian of the MoG on a highway scenario. d) the result of the learning process after the model weight filtering. 91
3.4 Traffic flow direction learning process: a), b) and c) show the evolution of the orientation pattern modeled by the first Gaussian of the MoG on a highway scenario. d) the result of the learning process after the model weight filtering. 92
3.5 Flowchart of the proposed wrong way drivers detection system. . 93
3.6 Some examples of false optical flow estimation. 95

3.7 Sites in Portuguese highways used to validate the proposed wrong way vehicle detection system. 96
3.8 Wrong way vehicle detection in a typical outdoor scenario. The vehicle is reversing in a lane disabled for maintenance. 97
3.9 Wrong way vehicle detection in a typical outdoor scenario. The vehicle is reversing in the hard shoulder. 98
3.10 Wrong way vehicle detection in a tunnel scenario. The vehicle is reversing in the hard shoulder. 99
3.11 Wrong way vehicle detection at night. The vehicle is reversing in a lane disabled for maintenance. 100
3.12 Real wrong way vehicle detection at night. 100
3.13 Real wrong way vehicle detection. 101
3.14 Some situations of false wrong way vehicles detected in the performed tests. 102
3.15 Wrong way vehicle detection under occlusion in a simulated event. 104
3.16 Miss detection of wrong way vehicle due to image noise. 104

4.1 Flowchart of the proposed stopped vehicles detection process. . . 109
4.2 Flowchart of the pixel history cache data structure used to store the pixels color history and subsequently used to identify the static pixels. 110
4.3 Some examples of stopped vehicle detection in some Portuguese highways scenarios. a) tunnel entrance; b) stopped vehicle with the hazard warning lights turned on; c) tunnel scenario; d) stopped vehicle on the left side road (the one on the right was validated before); and e) bridge scenario. 119
4.4 Example of stopped vehicle detection with occlusions. 120
4.5 False stopped vehicle detection. The areas where the vehicle's lighting is projected are segmented as foreground. For each pixel of those areas, the most frequent color is white and those pixels are classified as static. 121

5.1 Flow chart of the developed Automatic Traffic Surveillance System prototype. 126

5.2 Integration of the ATS System library with ATS Adapter. 128
5.3 Integration of the ATS System in the Brisa surveillance cameras network and communication with iBrisa. 129
5.4 Two examples of scenarios with non wanted roads to analyze. a) Two way road on the left side of the highway; b) Bridge over the highway. 130
5.5 Two examples of scene processing mask. 130
5.6 Two examples of scene lanes identification mask. 131
5.7 Two examples of scene learned flow patterns mask. 131
5.8 Real-time processing images of a set of sites. 132
5.9 Real-time processing images of a site with detailed information. . 132
5.10 List of wrong way vehicle events detected for a site. 133
5.11 Stored data of a wrong way vehicle event. 133

Chapter 1

Introduction

Over the last decades, a special attention has been given to Intelligent Transportation Systems (ITS). The increasing research and development in this area has been strongly motivated by urbanization, changes in population density, and by the significant increase in the number of vehicles. Within the scope of ITS technologies are topics like enhancing safety, efficiency of the surface transportation system, increasing mobility, and environmental sustainability. The associated technologies can vary from basic management systems - such as car navigation; traffic signal control systems; variable message signs; automatic number plate recognition or speed cameras - to monitoring applications, such as security CCTV systems; weather information; etc.

The increase in the number of vehicles circulating on the roads and the subsequent increase of the number of roads led to the necessity of creating active systems of roadway safety. In this thesis, one specific roadway safety system will be addressed, focusing on the monitoring of drivers' behaviors through the analysis of the CCTV signal.

Lately, one of the most significant efforts in ITS research has been the development of visual surveillance systems that could help reduce the number of traffic incidents and traffic jams in urban and highway scenarios. Several different types of devices, including loop detectors, sensors, and cameras, have been employed in traffic monitoring systems. Vision-based analysis systems have become popular in transportation management due to their capability to extract

very rich information on road traffic compared to the sensor-based systems.

Typically, the detection of abnormal situations based on video surveillance in highways is performed by human operators at traffic control centers (see Fig. 1.1). Tickner et al. [3] demonstrated that the level of attention of a human operator and his accuracy of incident detection decreases over time. Nowadays, this problem has become more complex because of the significant increase in the number of surveillance cameras along the highways. Therefore, robust automatic traffic surveillance systems are required to accurately detect abnormal events in highways. In order to create such system, it is necessary to classify vehicles' behaviors and to understand their interactions in typical traffic scenarios. Several vision-based automatic traffic surveillance systems have been proposed [27, 20, 11, 47, 45, 54, 18, 19, 73] and these demonstrate a good potential for highway surveillance applications. Nowadays, vision-based traffic surveillance systems are widely used in ITS. The goal of a traffic surveillance system is to extract traffic information, such as vehicle count, traffic events, and traffic flow, which play an important role in traffic analysis and traffic management. However, it is challenging to maintain detection accuracy at all time since vision-based processing is sensitive to environmental factors such as weather conditions, shadows, lighting variation, etc.

In the present thesis, some vision-based solutions are proposed to help human operators in the hard task of detecting abnormal situations in highways. The presented solutions are part of an Automatic Traffic Surveillance (ATS) system that is being developed to be applied to the surveillance cameras of Brisa's highways[1]. This ATS system is divided into two main parts: a) an Automatic Incident Detection (AID) system and b) an Automatic Traffic Monitoring (ATM) system. The aim of the AID system is to detect and track potentially anomalous traffic events in highways. By anomalous events it is meant the detection of vehicles that stop on highways or on hard shoulders, vehicles driving on the wrong direction, vehicles constantly switching between lanes, low speed vehicles, and also high speed vehicles. The aim of the ATM system is to provide some useful statistical information about the traffic. Some of the statistical features to be

[1]Brisa – Auto-estradas de Portugal S.A. (www.brisa.pt) is the largest transport infrastructure company in Portugal. It has about 550 surveillance cameras along the 1,100-kilometer highway network.

Figure 1.1: Traffic Control Center at Carcavelos, Portugal. The surveillance of the highways operated by Brisa all over the country is performed in this Traffic Control Center.

retained are: the number of vehicles circulating on each of the lanes, vehicles' average speed, road occupancy rate, etc.

The presented work focus on the detection of dangerous behaviors in highways in order to improve road safety. Two distinct solutions are proposed to detect two dangerous situations in highways: a) wrong way vehicles; and b) stopped vehicles on the road or on the hard shoulder. An immediate detection of these dangerous behaviors could help prevent serious accidents by warning the oncoming vehicles (via traffic telematic systems or radio announcements) and by warning the police. The proposed system aims at automatically detecting these events, triggering an alarm on the highway traffic telematic system, and consequently warning the human operator, who will handle the situation. The system must be robust to illumination changes, adverse weather conditions, small camera movements, being able to robustly track vehicles with anomalous behaviors against occlusions and crowded situations.

1.1 Thesis Overview

This thesis is divided into four main parts:

Part I: Robust Outdoor Vehicle Detection – Presentation of a robust segmentation process for outdoor scenarios based on background subtraction. Three different background models are used to model the scene variations and to deal with illumination changes. In order to minimize false positive detections in the the segmentation process, two different shadow detection algorithms are presented and tested. An optical flow algorithm is also used to estimate the vehicle's motion and to validate blobs previously detected in the segmentation process. Several results of validation experiments performed in outdoor scenarios are shown.

Part II: Automatic Incident Detection – Two different solutions are presented to detect dangerous behaviors on highways. These dangerous behaviors can be either vehicles stopped in the road or in the hard shoulder and/or wrong way vehicles.

The proposed methodology to detect stopped vehicles has three main phases. Firstly, there is the segmentation of all vehicles in the scene. Secondly, it is verified if there are any static pixels segmented over a certain period of time. Those static pixels are then grouped into blobs. The static pixels identification is based on a pixel history cache analysis. Finally, a blob temporal validation is applied to discard false positives. If the validation succeeds, an alarm is triggered in the traffic telematic system.

The solution presented to detect wrong way drivers in highways has three main stages. Firstly, the orientation pattern of vehicles motion flow is learned and modeled by a mixture of Gaussians (Learning Phase). Then, there is a Detection Phase, where it is verified, in each frame, if the direction of the areas where movement was detected matches the learned direction model. On both phases, a Block Median Filtering is applied to the motion flow in order to remove noisy flow vectors. Finally, a temporal validation is used to validate the object as a real wrong way vehicle. If the validation succeeds, an alarm is triggered.

Several detection events in outdoor scenarios and validation results are presented.

Part III: Automatic Traffic Surveillance System – An Automatic

Traffic Surveillance system prototype based on algorithms developed on this thesis is presented. This system was integrated in the Telematic Management System of Brisa. Mechanisms of online analysis of false positives and miss detections were developed using web technologies. This web tool is also used to perform real-time analysis of the ATS system.

Part IV: Final Notes – The last part of the thesis contains a general conclusion and discussion. List of publications made during the research work on this thesis is also presented.

Part I

Robust Outdoor Vehicle Detection

Chapter 2

Foreground/Background Segmentation

2.1 Introduction

Background/foreground segmentation is an essential component of most video surveillance systems involving object detection and tracking. Such systems require both robustness to lighting variation and computer feasibility. Some examples of applications that use foreground segmentation techniques are video surveillance, gesture recognition, object tracking, pattern classification, quality control, object-based video encoding, satellite images analysis, medical images analysis, etc.

This chapter is dedicated to the foreground segmentation problem applied to robust outdoor vehicles detection. It is important to remember that motion segmentation is never meant to be a goal on its own. The output of the segmentation process is the basis for subsequent systems like traffic monitoring and automatic incident detection systems. Therefore, the segmentation process has a very serious responsibility: the performance of the overall system is directly affected by the performance of the segmentation process.

Although a lot of research has been done in the field of foreground segmentation, a lot of difficulties have still to be considered in this area, especially how to obtain good results under dynamic environments, changing situations,

and different weather conditions. Some challenging situations are illustrated in Fig. 2.1. In outdoor scenes, the main challenges faced by the foreground segmentation system are:

Quick illumination changes Quick illumination changes completely alter the color characteristics of the background, thus increasing the deviation between background pixels and the background model in color or intensity. This results in a drastic increase in the number of falsely detected foreground regions and, in a worst case scenario, the whole image appears as foreground. This shortcoming makes surveillance under partially cloudy days almost impossible with high false positive rates.

Initialization with moving objects If moving objects are present during initialization, then part of the background is occluded by moving objects. Therefore, many algorithms require a scene with no moving objects during initialization. This situation imposes serious limitations on systems to be used in high traffic density areas.

Shadows Objects cast shadows that might also be classified as foreground due to the illumination change in the shadow region. This kind of problem can change the vehicle shape or, in a worst scenario, vehicles can be merged into a single region.

Moving background objects Background objects like swaying trees or fluttering flags are very difficult to include in the background model since each background pixel can be modeled by many different colors in this situation. Camera vibration can also be associated with this problem.

Weather conditions A surveillance system working in an outdoor scenario has to deal with different weather conditions like rain, fog, dense moving clouds, snow, etc. Adverse weather conditions can generate low contrast images and moving background objects like raindrops or snowflakes. The wet road can also pose a problem for the segmentation process, because of the vehicles' lights reflection.

Image quality The system should be able to work with every type of surveillance cameras available, adapting itself to the camera acquisition condi-

tions.

Automatic gain control With the adjustment of the camera's AGC, the brightness of the whole image changes and the background pixels are classified as foreground.

Work 24/7 It is demanding for an outdoor segmentation system to be able to work 24 hours a day, 7 days a week. The scene model should robustly adapt itself to variations throughout time.

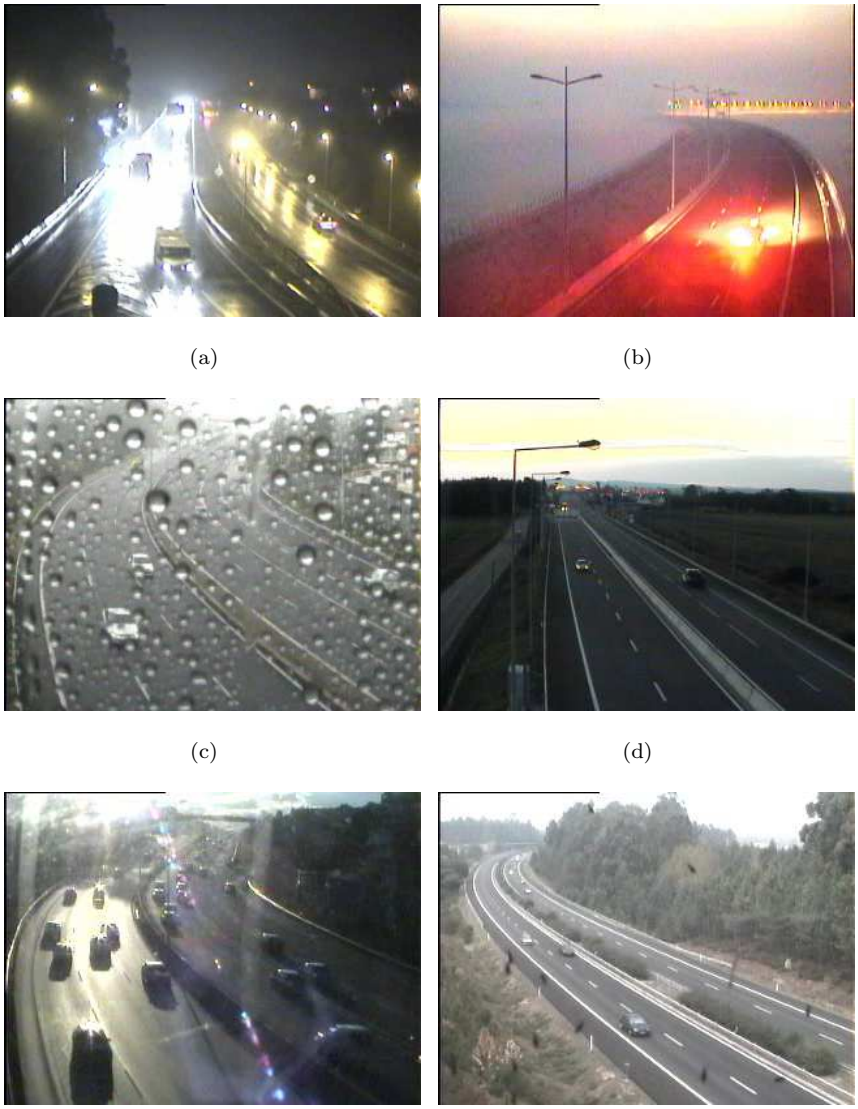
Computational cost A segmentation process can not demand a high computational power, because this would pose a problem for the integration into a real-time tracking system and/or for its use as an event detection system.

Several algorithms have been proposed to deal with the above mentioned problems in outdoor scenarios. However, finding a solution that robustly deals with all of these situations still a demanding and hard task.

2.1.1 Related Work

Over the last decades, both foreground analysis and foreground segmentation have become very important for a wide range of applications. This was only possible because of recent developments, such as the ability to process complex algorithms in real-time due to the computational power, and the available memory of recent computers. Most of the existing solutions either make strict assumptions about the scene or simply fail in outdoor scenarios when handling moving background regions and abrupt lighting variations resulting from moving clouds or camera's automatic gain control.

Several different segmentation approaches have been proposed to deal with problematic situations. Background subtraction based approaches are the most discussed pixel-wise techniques for foreground segmentation. Even though they are different, most background subtraction methodologies share common procedures: it is assumed that the observed image sequence, $I^t(\phi)$, is made of a fixed background model, $B(\phi)$, in front of which moving objects are observed. If one assumes that a moving object at frame t has a color distribution different from



(a) (b)

(c) (d)

(e) (f)

Figure 2.1: Some challenging situations for a segmentation process in outdoor scenarios: a) Vehicles' lights reflected off the wet road; b) Fog over a bridge; c) Rain drops in the camera lens; d) Low contrast image; e) Sun light in the direction of the camera; and f) Insects in the camera lens

the one observed in $B(\phi)$, the principle of background subtraction methods can be defined by Eq. 2.1, where M_F is the foreground mask at frame t, $diff$ is a distance between $I^t(\phi)$ and $B(\phi)$ at the pixel ϕ, and τ is a threshold.

$$M_F = \begin{cases} 1 & if\ diff(I^t(\phi),\ B(\phi)) > \tau \\ 0 & otherwise \end{cases} \tag{2.1}$$

These methodologies can be parametric or non-parametric, and the analysis can be pixel-based, region-level, or a combination of these different analysis methods. These methodologies have four main discussion points: a) $B(\phi)$ modulation; b) the procedure to update $B(\phi)$; c) determining the thresholds (τ) used to classify a pixel as foreground or background; d) and the foreground pixel validation criteria. Some state-of-the-art foreground segmentation algorithms will be described below.

The method proposed by Haritaoglu et al. [33] uses only grayscale information to detect foreground pixels. Firstly, a background model is generated with N frames, in which a pixel can have three values: minimal intensity (m), maximal intensity (M) and maximal difference value of two successive frames. The difference images are calculated with $I^t(\phi)$ and both the $I^t_m(\phi)$ image and $I^t_M(\phi)$ image. These images are used for the classification in which the foreground pixels are given if the difference values are higher than the values of the maximal inter frame difference. After the segmentation procedure, some post processing steps are also applied to reduce noise. Furthermore, the classified background pixel are then used to update the background model.

In [38], Horprasert et al. assume that luminance and chrominance have to be separated from each other on the RGB color space by generating a new color model. Thus, there is an expected chromaticity line in which the pixel value should be kept. The expected chromaticity is obtained by the arithmetic means of each pixel RGB values calculated over a number of background images. The distortion from this line is given as both chromaticity and brightness distortion being generated by standard deviation. With these distortions, several thresholds are determined to classify the pixel in one of the four possible classes: foreground, background, shadowed, and highlighted background.

In [28], François et al. assume that in the background only very slow global

changes can occur and, therefore, the color values of each pixel build a sphere cluster in the RGB color space. With this assumption, a background model as a Gaussian distribution is generated by considering the mean value and standard deviation for each pixel. In this method, the HSV color space is used instead of the RGB. The current image,$I^t(\phi)$, is subtracted from the mean value model and the obtained difference values of each pixel are then thresholded with a value proportional to the standard deviation model. Moreover, an update of the background model is also given.

The system proposed by Jabri et al. [41] uses both color and edge information to perform the foreground segmentation. The background model is trained for both mentioned parts by calculating the mean and standard deviation for each pixel, ϕ, of each color channel. With the subtraction of the incoming current frame, $I^t(\phi)$, for each channel, confidence maps are generated for both color and edge information. After that, a combination of the two maps are utilized by taking its maximum values. Finally, an hysteresis thresholding is used to obtain the foreground mask.

Cavallaro et al. [16, 17] proposed a methodology to detect foreground pixels based on $YCbCr$ color space. For each channel of $YCbCr$ color space an image differencing between the background, $B^t(\phi)$, and the current image, $I^t(\phi)$, is performed. An edge detection algorithm based on Sobel operator is computed using each partial result. Then, the three sub-results are fused together. A post processing step using morphological operations is used to connect the edge information.

Hong et al. [36] also models the background, but this time both well-known RGB and normalized rgb color are applied. As mentioned in previous methods, the mean and standard deviation are used again and these are estimated for each color component. Each color representation has its own classification procedure in which the current image, $I^t(\phi)$, is firstly converted for each color representation. Within each color space the pixel can be classified into four categories: a) foreground, b) foreground with shadow, c) background, and d) background with shadow.

Some other methodologies to detect foreground pixels will be described below. These more relevant methodologies are presented in order to introduce the

proposed algorithm to detect foreground pixels.

Mixture of Gaussians

The existing background modeling methods can be classified as either single-layer or multi-layer. Single layer methods obtain a model for the color distribution of each pixel, based on past observations. Usually, a single Gaussian is used to model the statistical distribution of a background pixel, being updated through a blending approach. Stauffer and Grimson [71] modeled the background with a mixture of Gaussian models (MoG). Instead of explicitly modeling every pixel values as one particular type of distribution, the background is obtained by a pixelwise mixture of Gaussian distributions to support multiple backgrounds. In this process, each pixel's recent history, $\{X_1, ..., X_t\}$ is modeled by a mixture of K Gaussian distributions. The probability density function of observing a certain pixel X value is given by Eq. 2.2 where K is the number of distributions, $\omega_{i,t}$ is an estimate of the weight of the i^{th} Gaussian in the mixture at the time t, $\mu_{i,t}$ is the mean value of the Gaussian i^{th} in the mixture at the time t, $\Sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian in the mixture at the time t, and where η is a Gaussian probability density function and is given by Eq. 2.3.

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{2.2}$$

$$\eta(X, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)} \tag{2.3}$$

K is chosen taking into account the available memory, the computational power, and the complexity of the data to be modeled. Also, for computational reasons, the covariance matrix is given by Eq. 2.4. This assumes that the three color channels are independent and have the same variance. This assumption allows us to avoid a costly matrix inversion at the expense of some accuracy.

$$\Sigma_{k,t} = \sigma_k^2 \mathbf{I} \tag{2.4}$$

In order to update the model with a new pixel value, a standard method for maximizing the likelihood like an expectation maximization algorithm could

be used. However, it requires a high computational power. Therefore, a K-means approximation is used for the updating procedure. First, the distance in standard deviations with respect to each Gaussian is computed. When the distance to the closest Gaussian is less than 2.5 standard deviations, the weight (Eq. 2.5), mean (Eq. 2.6) and variance (Eq. 2.7) of that Gaussian is updated, where α is the learning rate and $M_{k,t}$ is 1 for the match model and 0 for the remaining models. For the remaining Gaussians, only the weight is updated, decreasing the prior $\omega_{k,t}$ of those Gaussians. When there is no match with any of the Gaussians, the Gaussian with the smallest weight ω_t is replaced by a new Gaussian, with the sample value as its mean, preset initial value as variance and a very low prior weight ω_t. After this approximation, the weights are renormalized.

$$\omega_{k,t} = (1 - \alpha)\omega_{t-1} + \alpha M_{k,t} \tag{2.5}$$

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \tag{2.6}$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \tag{2.7}$$

where

$$\rho = \alpha\eta(X_t|\mu_k, \sigma_k) \tag{2.8}$$

To obtain the background model it is necessary to select the correct portion of the mixture model that best represents the scene background. First, the Gaussians are ordered by the value of ω/σ. Then, the first B distributions are chosen as the background model (Eq. 2.9), where T is a measure of the minimum portion of the data that should be taken into account by the background. A pixel is classified as foreground if it has a log-likelihood lower than a given threshold with respect to the background model B.

$$B = argmin_b \left(\sum_{k=1}^{b} \omega_k > T \right) \tag{2.9}$$

Codebook Model

A methodology to segment objects in a non-static background based on a pixel color history cache was proposed by Kim et. al. [46]. Just like the methodology presented in Sec.2.1.1, this segmentation algorithm uses a multi-layer background model. Each pixel's sample background values are quantified into *codebooks*. These are a compressed way of the background model for a long image sequence. Each background pixel model (*codebook*) is represented by one or more *codewords*. Each *codeword* represents a distinct mode of a background pixel.

Let X be a training sequence for a single pixel consisting of N RGB-vectors: $X = \{x_1, x_2, ..., x_N\}$. Let $C = \{c_1, c_2, ..., c_L\}$ represent the *codebook* for the pixel consisting of L *codewords*. Each pixel has a different *codebook* size based on its sample variation.

Each *codeword* $\mathbf{c}_i,\ i = \{1, ..., L\}$ consists of an RGB vector $\mathbf{v}_i = (\bar{R}_i,\ \bar{G}_i,\ \bar{B}_i)$ and a 6-tuple $\mathbf{aux}_i = \langle \check{I},\ \hat{I},\ f_i,\ \lambda_i,\ p_i,\ q_i \rangle$. The tuple $\mathbf{aux}_i$ contains the brightness values and the temporal variables described below:

$\check{I},\ \hat{I}$	the minimum and the maximum brightness, respectively, of all pixels assigned to this *codeword*
f	the frequency of the *codeword*
λ	the maximum negative run-length defined as the longest interval during the training period that the *codeword* has not recurred
$p,\ q$	the first and the last access times, respectively, that the *codeword* has occurred

During the *codebook* construction, sample pixels are considered in chronological order . The sample is first compared to the existing *codewords* to look for a match. A match occurs when the sample color and brightness distortion are within the *codeword* parameters. The *codeword* matching area is represented by a cylinder in RGB color space (Fig. 2.2). If a match occurs, the RGB values of the matching codeword are updated, as well as the values of the 6-tuple. If no match is obtained, a new *codeword* is initialized with the sample RGB values. For an input pixel $\mathbf{x}_t = (R, G, B)$ and a *codeword* $\mathbf{c}_i$ where $\mathbf{v}_i = (\bar{R}_i,\ \bar{G}_i,\ \bar{B}_i)$

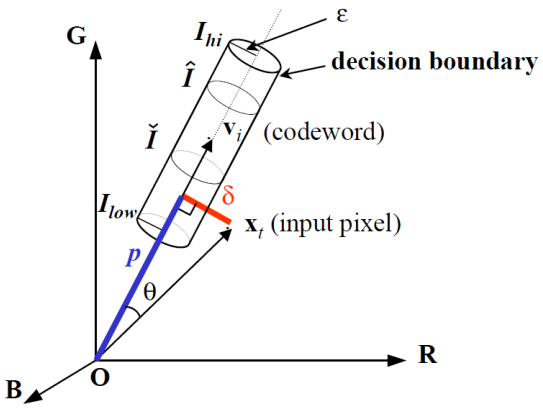


Figure 2.2: Color model used – a separate evaluation of color and brightness distortion.

the color distortion is given by Eq. 2.10 where p^2 is given by Eq. 2.11. This color distortion measure can be interpreted as a brightness-weighted version in the normalized color space.

$$colordist(\mathbf{x}_t, \mathbf{v}_i) = \delta = \sqrt{\|\mathbf{x}_t\|^2 - p^2} \tag{2.10}$$

$$p^2 = \|\mathbf{x}_t\|^2 \cos^2\theta = \frac{\langle \mathbf{x}_t, \mathbf{v}_i \rangle^2}{\|\mathbf{v}_i\|^2} \tag{2.11}$$

In order to allow brightness changes in detection , statistics of $\check{I}$ and $\hat{I}$ are stored, which represent the minimum and the maximum brightness of all pixels assigned to a *codeword*. In order to intrinsically deal with shadows and highlights, brightness can vary within a certain range. The range $[I_{low,I_{hi}}]$ for each *codeword* is defined by Eq. 2.12 where $\alpha < 1$ and $\beta > 1$. The logical brightness function is defined by Eq. 2.13.

$$[I_{low,\ I_{hi}}] = \left[\alpha\hat{I},\ min\left\{\beta\hat{I},\ \frac{\check{I}}{\alpha}\right\}\right] \tag{2.12}$$

$$brightness(I,\ \langle\check{I},\ \hat{I}\rangle) = \begin{cases} true & if\ I_{low} \leq\ \|\mathbf{x}_t\|\ \leq I_{hi} \\ false & otherwise \end{cases} \tag{2.13}$$

Since the training set X can contain both background and foreground pixels it is necessary to discard the foreground pixels from the model. A temporal

filtering procedure is performed to obtain the background model. It is assumed that true background pixels (static pixels and moving background pixels) are quasi-periodic, i.e., their pixel values should repeat over time. The temporal filtering process consists of removing the pixel *codewords* with a large maximum negative run-length λ. Let M (Eq. 2.14) and T_M denote the background model (which is a refined *codebook* after temporal filtering) and the threshold value, respectively.

$$M = \{\mathbf{c}_m | \mathbf{c}_m \in \mathbf{C} \wedge \lambda_m \leq T_M\} \tag{2.14}$$

Unlike MoG segmentation process (Sec. 2.1.1) which computes probabilities using costly floating points operations, this method does not involve probability calculation. The distance from the sample to the nearest cluster mean is computed. The pixels classification, FGS(x), of an incoming pixel value x is defined by Algorithm 1, where ϵ is the detection threshold. The pixel is detected as foreground if no acceptable matching *codeword* exists. Otherwise it is classified as background.

Algorithm 1 Segmentation Process

1: $x = (R,\ G,\ B)$, $I \leftarrow \sqrt{R^2 + G^2 + B^2}$

2: For all *codewords* in M (see Eq. 2.14), finding the *codeword* $\mathbf{c}_m$ that matches x based on the following conditions:

- $colordist(x, \mathbf{c}_m) \leq \epsilon$
- $brightness(I,\ \langle \check{I},\ \hat{I} \rangle) = true$

3: Update matched *codeword*

4: Pixel classification:

$$FGS(x) = \begin{cases} foreground & if\ there\ is\ no\ match \\ background & otherwise \end{cases}$$

Foreground Segmentation Based on Background Suppression

A segmentation process based on background suppression was presented by Calderara et al. in [15]. This segmentation process is based on a system called

SAKBOT (Statistical And Knowledge-Based Object Tracker) presented by Cucchiara et al. in [22] and it includes new techniques for improving reliability in complex scenes.

The initial background model is obtained using a background bootstrapping technique. In this approach, the image is divided into blocks of 16×16 and selectively updates the background model with a block whenever a sufficiently high number of pixels within the block are not in motion. The motion is evaluated with a thresholded single difference between two consecutive frames. This learning process stops when every block is considered "stable" for two consecutive frames. During the processing period, a selective background model update is performed using a temporal median. Only pixels classified as foreground are updated.

In order to identify the foreground pixels a background differencing is used. The difference between the current image I_t and the background model B_t is computed using Eq. 2.15, where i^T is the 1×3 identity vector.

$$M_t(x,y) = \frac{(I_t(x,y) - B_t(x,y)).i^T}{3} \tag{2.15}$$

$M_t(x,y)$ is the foreground image and contains the gray-level information of the difference. Two different local thresholds are used to binarize $M_t(x,y)$: a low threshold T_{low} and a high threshold T_{high}. Let $b_p(x,y)$ be the value at position p inside the ordered circular buffer b of pixel (x,y) and $b_{\frac{k+1}{2}+1}$ the median value. This vector is the same that is used to update the background model. Thresholds are computed using Eq. 2.16 and Eq. 2.17, where λ is a fixed multiplier, while l and h are fixed scalar values and $h > l$.

$$T_{low}(x,y) \;=\; \lambda \left(b_{\frac{k+1}{2}+l}(x,y) \;-\; b_{\frac{k+1}{2}-l}(x,y) \right) \tag{2.16}$$

$$T_{high}(x,y) \;=\; \lambda \left(b_{\frac{k+1}{2}+h}(x,y) \;-\; b_{\frac{k+1}{2}-h}(x,y) \right) \tag{2.17}$$

A pixel is marked as foreground in F_t if it is segmented with T_{low} and it is spatially connected to at least on pixel segmented with the T_{high} threshold. A shadow detection algorithm described in [22] is used to identify and discard shadowed pixels from the foreground mask. This algorithm assumes that the

shadow darkens the underlying background, without significantly changing its color. This algorithm is described in detail in Sec. 2.3.1. Finally, the list of moving objects at the time t is extracted from F_t using a two-pass labeling algorithm.

After the blob detection and the shadow removal, a region-level analysis is performed in order to remove all moving objects generated by small motion of the background. This validation is performed accounting for joint contribution coming from color and gradient information. The gradient is computed with respect to both spatial and temporal coordinates. An overall validation score is computed for each blob. This value is then thresholded and, if it is below the threshold, the blob is discarded and its pixels are marked as belonging to background.

Adaptive Multi-background Modeling

In [14], Boult et al. presented a system to detect and track non cooperative targets under non stationary environments. This system is denoted as Lehigh Omnidirectional Tracking System (LOTS). This algorithm uses two gray level background images: the primary background B_P and the secondary background B_S. This allows the algorithm to cope with intensity variations due to noise or fluttering objects, moving in the scene. Both B_P and B_S are updated via temporal blending to allow them to track slow lighting changes. In order to deal with "ghosts" generated by this blending process, a third background is used. This third background is called *old-image* and it is not updated via the blending model. This background is a copy of an image from somewhere between 9000-18000 frames ago.

Just like in [15], foreground objects are identified using two thresholds, (T_l and T_h). During the thresholding process (threshold-with-hysteresis), a lower resolution image of those pixels above the threshold is created. Each pixel in the parent image (low resolution image) maintains a count of how many of its associated children (high resolution pixels) were above the threshold. The resolution is reduced by a factor of four in each direction. A connected component analysis is performed to identify the foreground areas and to group the segmented pixels. To speed this precess up, the connected component analysis

is applied to the parent image. This resolution reduction fills small gaps as well. Even though it is not as "uniform" as the morphological processing, it is considerably faster. This thresholding with foreground pixel grouping is called quasi-connected components (QCC) analysis.

Finally, a region-level analysis is performed to remove noise regions. Three different processes are used to validate the detected blobs: area analysis, lighting normalized analysis and "ghost" regions detection. The area region analysis is used to filter blobs with small areas, these blobs are considered noise. The lighting normalized analysis is used to discard blobs generated by shadows or highlights. The threshold-with-hysteresis is aplied again to each blob, but only the primary background is analyzed. The current frame and the background model are normalized in the blob position before this process. The third cleaning phase is used to verify if the foreground blob is a "ghost". This is done by performing the threshold-with-hysteresis comparison against an intensity normalized version of the old image.

2.2 Segmentation Process

Taking into account the problems listed previously, in this section, a robust segmentation process for outdoor scenarios is presented. The proposed system aims at robustly dealing with different weather conditions, illumination changes, non-static background objects, shadows and image noise. This system was inspired in LOTS segmentation process (see Sec. 2.1.1), but with some improvements to increase robustness to outdoor scenarios and to work 24 hours a day, 7 days a week. The system requirements and its explanation were presented in detail in Sec. 2.1.

This segmentation process uses a background subtraction based methodology to identify the objects in the image. Two different background models are used to model the background scenario; one to model the static background objects and the other to model the non-static background objects. In order to avoid background model corruption due to integration of pixels mis-classified as background, a third background model is also used. This third model is obtained by a median process of n frames with a Δ frames interval. Two distinct

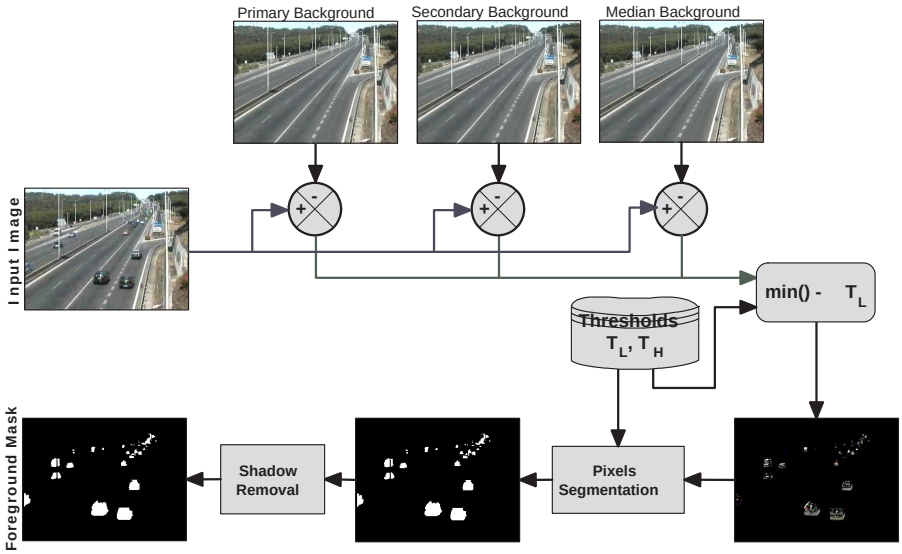


Figure 2.3: Flowchart of the proposed segmentation process – pixel-level analysis.

thresholds are used: a per-pixel threshold for a robust adaptation to the scene variations during the day, and a global threshold, that is the minimum value, to eliminate camera noise, and scenario features, like swaying trees and camera pole vibrations. The global threshold is estimated in the system initialization by the analysis of the scenario variations. A shadow/highlight detection algorithm based on cross-correlation is also used to discard blobs or parts of blobs generated by lighting variation (moving clouds, artificial light changes, etc.). Blobs are validated at a region-level analysis, object shape and size, optical flow, and consistency over time are used to validate the segmented blobs. In order to deal with abrupt variations (e.g.: artificial lighting switching) a frame-level analysis is performed. Thus, the system can quickly adapt itself to scenario changes. The main parts of the methodology will be described in detail below. A flowchart of the segmentation process is illustrated in Fig. 2.3.

2.2.1 Background Modeling

In order to achieve a good segmentation in an outdoor environment it is necessary to have a background model that considers all scenes' variations not

classified as foreground. Otherwise, the false positive segmentation rate will increase. Nonetheless, the background model cannot be too comprehensive due to the high risk of mis-detection.

Three different background images are used to model the background of the scene in the proposed background modeling. These models are: a) B_P - the *primary* background; B_S - the *secondary* background; and B_M - the *median* background. The B_P is the main background model. It has to have, at all times, the background model that is closest to the current frame background. This model shall take into consideration fast variations in the scene, like lighting changes or camera AGC adjustment. The B_S background is used to model objects classified as static but that can undergo small variations in position and/or in shape. By static background object variations it is meant, for instance, swaying trees, fluttering flags, camera pole vibration (causing the image to shake), camera signal noise and digital video compression. Lastly, the B_M background is the median pixel value of the last n frames with a frame interval of Δ frames. This background is the cleanest one of the three backgrounds due to its median properties. Lighting changes, for example, turning artificial lighting on or off and shadows cast by moving clouds can deteriorate the B_P model. The update process of B_P (presented in Sec. 2.2.2) can also leave "ghosts" in the model. Therefore, B_M is used to avoid corruption and deterioration of B_P background in those situations.

Initial Background Model Estimation

A crucial task of all background subtraction based approaches is the initial background model estimation, that needs to be both accurate and fast. An incorrect initial background estimation can lead to "ghosts" in the segmentation process, and, consequently, false foreground pixels will arise in the segmentation process. In dense traffic situations it is frequently impossible to have a clear background for many frames in order to estimate the background model.

An usual approach to initialize the background model is the weighted average [9, 14]. This solution does not estimate a robust background model in scenarios with high traffic density. Due to the statistical properties of the average, all samples contribute to the final estimation. As a consequence, the colors of the

vehicle can blend into the background model. Another drawback of this process is the number of necessary frames to estimate an acceptable background model. In [15] a *bootstrapping* approach is used to estimate the background model. Just like in the weighted average approach, several frames can be necessary to estimate the background model.

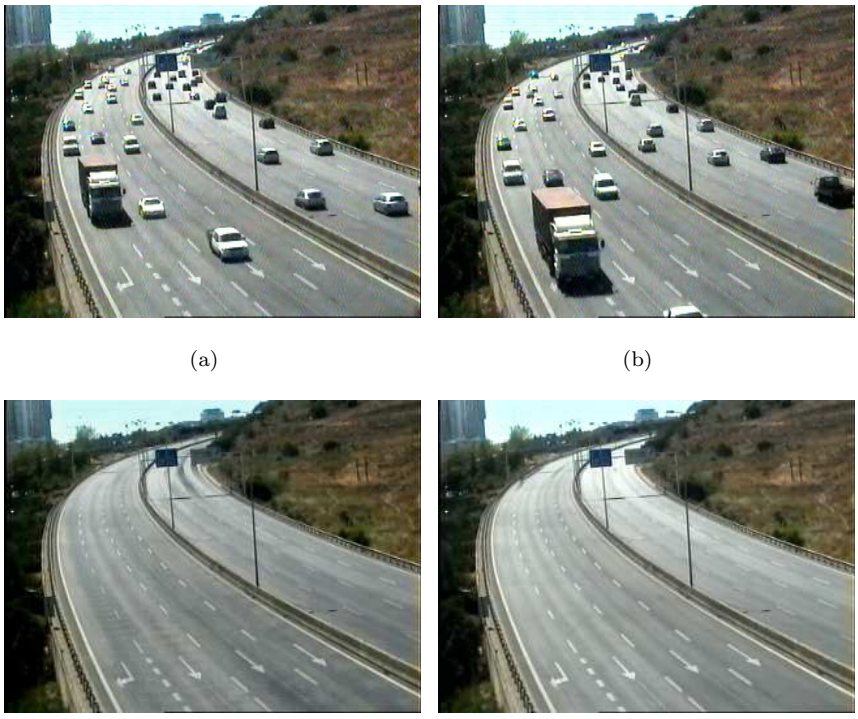
In order to achieve a robust initial background model and minimizing the necessary frames to do so, a median-based process was used. The proposed system calculates the median of a stack of frames based on 2.18, where ϕ is the pixel index, I_t is the current frame, n is the buffer size and Δ is the interval between samples acquisition. From the undertaken experiments, the median background estimation proved to be more accurate than the weighted average estimation. Figs. 2.4 and 2.5 illustrate the comparison between the background estimation with the weighted average and the median. As one can see, the weighted average estimation is darker in the regions where there is a higher number of vehicles passing due to the vehicle blending (top of the road). Besides, the weighted average estimation needs about 500 frames to achieve a good background model estimation in a high traffic density scenario, while the median estimation needs only around 50 frames. The median computational process spends about 3 seconds for a buffer with 50 samples. Initially, B_P and B_S are equaled to this initial background estimation B_M.

In Fig. 2.6 it is shown the effectiveness of this background modeling methodology in an outdoor scenario with moving shadows casted by clouds.

$$B_M(\phi) \;=\; median_{\phi}\left(I^{t}(\phi),\; I^{t-\Delta}(\phi),\; I^{t-2\Delta}(\phi),\; ...,\; I^{t-(n-1)\Delta}(\phi)\right) \quad (2.18)$$

2.2.2 Online Background Model Update

Just like the background model initialization process, the online background model update is a very important task in the performance of the segmentation process. This process is the main responsible for the system stability along the 24 hours of the day. To take into account illumination changes, it is demanding an accurate background update. Traffic jam situations can lead to a misclassification of pixels as background if the pixels in the background model are all



(a) (b)

(c) (d)

Figure 2.4: Initial background model estimation. a) and b) two frames of the scene where background model was computed. c) background model estimation with a weighted average of 500 frames; d) background model estimation with the median of 50 frames

(a) (b)

(c) (d)

Figure 2.5: Initial background model estimation with traffic jam situation. a) and b) two frames of the scene where background model was computed. c) background model estimation with a weighted average of 500 frames; d) background model estimation with the median of 200 frames

updated with the same criteria. In order to avoid this problem, the model update process is carried out after the segmentation process. A mask of the pixels classified as foreground ($M_F(\phi)$) is created during the segmentation process; the mask is then used to define the update criteria. Detailed description of the online process to update B_P, B_S, and B_M will be presented below.

Median Background Update

The online median background estimation is computed almost in the same way as the initial background model estimation but in this situation the buffer always contains the last n samples. The only difference has to do with the buffer size of the samples and the update frequency. Buffer sorting is a computationally heavy task, making it therefore necessary to minimize it. In order to reduce the computational time, both n and the update frequency were reduced (namely $n = 5$ and $\Delta = 30$). In order to achieve good results with a small buffer, the buffer of the samples is only updated if $M_F(\phi) = 0$ for the related pixel ϕ. If $M_F(\phi) = 1$ the buffer is updated in the next frame if $M_F(\phi) = 0$. A new pixel median estimation is computed every time the buffer is updated.

Primary and Secondary Background Update

The current scene conditions should be reflected in B_P and B_S in a normal situation. If it does not happen, then the B_M model is used. First of all, and to ensure that B_P is the closest background model to the currently classified background, B_P is swapped with B_S if B_S is more similar to the current background than B_P. During the foreground pixel segmentation process, the closest background model to the current classified background is also identified for each pixel. In order to avoid B_P and B_S degradation, it is also verified if B_M is the closest model to the current background. Since B_M is a filtered background model (median filtering), having it as the model closest to the current background means that B_P has some disturbance. B_P is equaled to B_M so that disturbances are eliminated. After this swap process, B_P pixels are updated with a weighted average using Eq. 2.19, where η is defined by Eq. 2.20. Different values of η are used in the background update process: η_s is the integration factor of pixels classified as foreground and η_s is the integration factor of the pix-

els classified as background. The integration factor η_f is used for updating the background pixel and is tunned for a fast adaptation to background variations. This background update process is performed at the pixel level. Nonetheless, η_s is much smaller than η_f to avoid the blending of vehicles in the model (namely $\eta_f = 0.05$ and $\eta_s = 0.0005$).

$$B_P^{t+1}(\phi) \;=\; \eta\; I^t(\phi) + (1-\eta)\; B_P^t(\phi) \tag{2.19}$$

$$\eta \;=\; \begin{cases} \eta_S & if\ M_F(\phi) = 1 \\ \eta_F & if\ M_F(\phi) = 0 \end{cases} \tag{2.20}$$

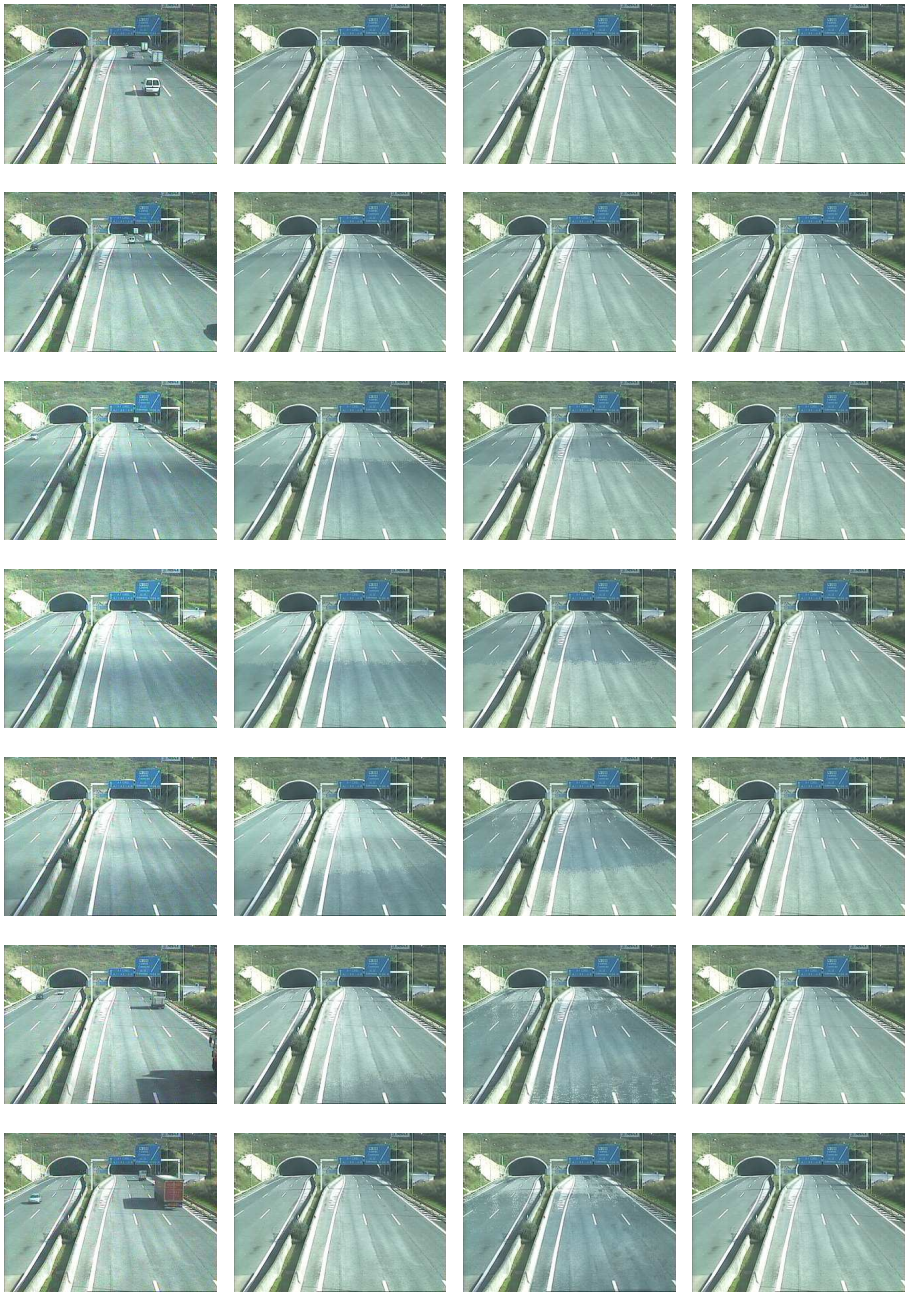
The objective of B_S is to store information about the pixels classified as background, but only those with more than one color on its model. After the learning phase, this background is equal to B_P, but in the detection phase and with the swap operation it is modeled with pixel variations if they exist. However, in some situations, it is possible that B_S is wrongly modeled, and, in this case, if it is not corrected it can lead to mis-detections during the segmentation process. To prevent B_S from being wrongly estimated, it is analyzed at all iterations the last time that the model was the closest model to the current background. If the B_S model is not used during more than T_d iterations, it is equaled to B_P. This way, the model is maintained only if it is frequently used.

2.2.3 Pixel-level Analysis

The main procedure in a segmentation process is the foreground pixel detection. It is verified if every pixel matches the background model. If a pixel does not match the model, it is classified as foreground pixel. A filtering process is also used in order to discard some noisy segmented pixels.

Foreground Pixels Thresholding

Most systems compute the difference between the current frame and the background image and consider as targets the pixels above a certain threshold. Afterwards, neighborhood pixels are clustered to form possible foreground regions. This process usually leaves gaps that might lead to an erroneous foreground



(a) Frame I^t (b) B_P^t Model (c) B_S^t Model (d) B_M^t Model

Figure 2.6: Background model adaptation over time with a moving shadow casted by a cloud. Note that B_P^t always models the scene variations, B_S^t maintains the previous background model, and B_M^t does not change during this period, i.e., it maintains the original background model before the moving shadow.

detection. Morphology can be used to fill in these gaps. However, threshold-with-hysteresis is preferred since it is a more accurate algorithm that only fills in meaningful gaps.

A two level thresholding-with-hysteresis is used to detect the foreground pixels, i.e., all pixels in a foreground region must be above the low threshold, T_l, and must have at least one neighbor pixel above the high threshold, T_h. The thresholds T_l and T_h will be described in detail in Sec. 2.2.6. The thresholded difference image ($D_{th}(\phi)$) is obtained using T_l, according to Eq. 2.21, where $Diff(\phi)$ is the minimum difference between the current frame and the background models (Eq. 2.22). The difference image, $D_{th}(\phi)$, is an RGB image and the differences are obtained for each channel separately. In Sec. 2.2.4 it will be explained how obtain the foreground pixels using $D_{th}(\phi)$.

$$D_{th}(\phi) = \begin{cases} Diff(\phi) - T_l & if\ Diff(\phi) > T_l \\ 0 & otherwise \end{cases} \tag{2.21}$$

$$Diff(\phi) = min_{i \in \{P,S,M\}} \left| I^t(\phi) - B_i^t(\phi) \right| \tag{2.22}$$

Foreground Pixels Filtering

A very challenging problem of a foreground objects segmentation system are the detection of shadowed and the highlighted regions (e.g.: vehicles and clouds cast shadows on the road, artificial lighting switching, etc.). Due to lighting variations between them and the background model, those regions can be classified as foreground. This problem can lead to a change in the shape of the vehicle, an increment of the vehicle's area, and to false vehicle detection. Shadow detection in outdoor scenarios is a challenging problem due to the variability of the shadow model. Description and solutions for this problem will be presented in Sec. 2.3.

2.2.4 Region-level Analysis

After the foreground pixel identification and shadow/highlight removal, two main procedures are carried out during the region-level analysis. Firstly, every pixel classified as foreground and validated in the pixel-level analysis is grouped

into blobs; these sets of blobs are the possible vehicles in the road. Secondly, some validation procedures based on blob area, reliability of segmented pixels, optical flow and a temporal validation are used to discard false foreground objects. These false objects detection are mainly caused by moving background objects like waving trees.

The final output of the overall segmentation process will be the list of validated blobs (vehicles) and $M_F(\phi)$ with the identification of all foreground pixels (pixels belonging to vehicles).

Foreground Pixel Grouping

Foreground pixel grouping is carried out after the thresholding phase. To perform the above-threshold pixel grouping, a *Quasi-Connected Components* (QCC) process is used. This QCC algorithm is inspired in [14] but with some improvements. This process combines thresholding-with-hysteresis (TWH) with gap filling and connected component labeling. In order to keep the QCC process fast, a reduction of resolution is performed. This reduction of resolution also enables gap filling.

During the TWH process it is also created a lower resolution matrix of those pixels above threshold. Each cell in the low resolution matrix maintains a count of how many of the associated pixels are above T_l and T_h in the original image. The resolution is reduced by a factor of 2 in each direction. This way, a cell in the low resolution matrix corresponds to a block of 2×2 pixels in the original image. To classify the block as foreground or background, the original QCC algorithm only analysis the pixels inside it. The proposed improvement analysis the neighbor blocks as well to perform the block classification.

Two criteria were used to define the foreground blocks by analyzing the low resolution matrix (only one needs to be true): a) a block has to have at least one pixel above T_h; b) a block has to have at least two pixels above T_l and has to have a neighbor block with at least one pixel above T_h. This way, the region has an overall high sensitivity, while also trying to ensure that at least some of the pixels are very unlikely to be false alarms. The resulting foreground mask is then resized to fit the original image size and the result is stored in M_F.

Blob Validation

In order to minimize false positive detections, every grouped blob is validated before being labeled as object/vehicle. A set of validation rules are used in order to discard blobs generated by false foreground pixels. Those false foreground objects are mainly generated by illumination changes and noisy pixels that were not identified in the pixel-level analysis. The set of rules used to validate the blobs are:

1 - Blob Area During the labeling process, the area of the objects is obtained. Blobs with an area below A_{min} are discarded. The choice of this threshold value, A_{min}, depends on the scene and on the object position.

2 - Reliability of the Segmented Pixels Internally, the blob should have a strong difference when compared to the background model. The blob is validated if $\chi_{Th}/Blob_{Area} > \delta$, where χ_{Th} is the number of segmented pixels with T_h threshold and δ is a ratio threshold.

3 - Blob Flow The estimation of the optical flow in the blob area occurs to determine if the blob is a moving object. If any motion is detected, then the blob is validated. The optical flow estimation procedure will be described in detail in Sec. 2.4.

4 - Temporal Validation This criterion is used to validate stopped blobs when the above criteria do not succeed. The blobs with no flow should have significant area overlapping between consecutive frames.

All labeled blobs are validated by this set of criteria. If a blob does not succeed in the three criteria (the third and the fourth criteria complement each other), it is removed from the blob list and the corresponding pixels in $M_F(\phi)$ are set to zero.

2.2.5 Frame-level Analysis

A frame-level analysis is used to deal with abrupt changes in the scene. These changes can be generated by abrupt lighting variations and radical camera motion, some examples are depicted in Figs. 2.7, 2.8, 2.9. In a regular situation, it

Figure 2.7: A sudden illumination change in a tunnel scenario.

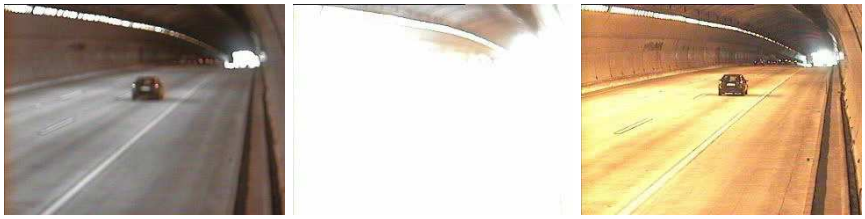


Figure 2.8: Three consecutive frames of an sudden lighting change in a tunnel scenario.

is expected a small variation in the number of segmented pixels between consecutive frames. This assumption is verified in every frame after the thresholding phase. If the segmented pixel growth is above a preset threshold, χ, it is considered that an abrupt change occurred on the scene, Eq. 2.23 represents the used criterion. The threshold value, χ, is obtained empirically and taking into account the average area and velocity of the vehicles in the image.

$$\left| \sum_{\phi} M_F^t(\phi) \; - \; \sum_{\phi} M_F^{t-1}(\phi) \right| > \chi \tag{2.23}$$

For a fast and robust adaptation to the present scene conditions the new background model is learned again. Segmentation thresholds are also estimated according to the new scene conditions. This learning process needs only a few number of frames and less than five seconds to restart the segmentation process. An example of the global segmentation process is depicted in Fig. 2.10.

2.2.6 System Thresholds

Four different thresholds are used in the segmentation process: the per-pixel threshold (T_{pp}), the low threshold (T_l), the high threshold (T_h), and the global

Figure 2.9: The camera's AGC changes the image brightness because of the truck that is coming towards the camera.

threshold (T_g). The T_{pp}, attempts to account the scene noise at a pixel through time, e.g., whenever the camera shakes, the edge pixels will have a significant intensity change; in these cases, T_{pp} will be increased. The T_g is the minimum value to eliminate camera noise and depends on the scenario. The low threshold T_l is the sum of T_g and the per-pixel threshold T_{pp} (Eq. 2.24) that dynamically adapts its value along the process. Finally, T_h is an higher threshold ($T_h > T_l$) and is obtained by Eq. 2.25 where Ψ is a fixed multiplier and is obtained empirically. The adoption of a dynamic per-pixel threshold is trivially explained by the fact that using a fixed, per-frame threshold results in a system less reactive to local lighting changes.

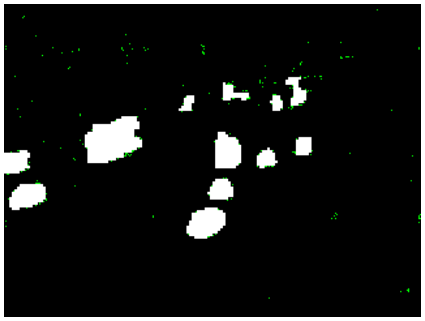
$$T_l(\phi) \;=\; T_g(\phi) \;+\; T_{pp}(\phi) \tag{2.24}$$

$$T_h(\phi) \;=\; \Psi \;\times\; T_l(\phi) \tag{2.25}$$

Thresholds Initialization

Just like the background model, the threshold T_g is also initialized according to the scene and camera properties in the learning phase. Using a stack of learning frames, the T_g threshold is estimated through the analysis of the pixel color variation [15]. In Sec. 2.2.1, a buffer of n frames is used to estimate the initial background model. The same sorted buffer ($B_k(\phi)$) is used to estimate T_g, according to the Eq. 2.26. Where k is position inside the sorted buffer B of pixel ϕ and, consequently, $B_{\frac{n+1}{2}+1}$ is the median value of the data. The threshold T_g is computed by Eq. 2.26. Where λ is a fixed multiplier, while g is a fixed scalar (namely $\lambda = 2$ and $g = 5$).

(a) Vehicles identification with a green bounding box.



(b) Green pixels - above T_l. White pixels - above T_h.

(c) Detected foreground pixels.

(d) Primary background model.

(e) Difference between the current frame and the background model.

(f) Optical flow estimation.

Figure 2.10: Output of the proposed segmentation process in a typical outdoor scenario.

$$T_g(\phi) \;=\; \lambda \left(B_{\frac{n+1}{2}+g}(\phi) \;-\; B_{\frac{n+1}{2}-g}(\phi) \right) \tag{2.26}$$

During the detection phase, the threshold T_{pp} will adapt itself to lighting and scene variations. This threshold is empirically initialized with 10.

Thresholds Update

The threshold levels are updated in the last step of the frame processing, i.e., after the foreground/background segmentation. In order to adjust T_{pp} to the scene variations, $M_F(\phi)$ and $D_{th}(\phi)$ are analyzed. For every pixel, ϕ, if $D_{th}(\phi) > 0$ and $M_F(\phi) = 0$, then ϕ is labeled as a noisy pixel and $T_{pp}(\phi)$ is increased by T_{inc}. By increasing T_{pp} of noisy pixels, the system sensibility is reduced, as well as the probability of having pure noise regions classified as targets. When pixels are not segmented ($D_{th}(\phi) = 0$) more than μ frames their T_{pp} is decreased by T_{dec}, therefore increasing their sensibility. To avoid the instability of the system, T_{inc} should be much higher than T_{dec}. Those values were set to $T_{inc} = 8$, $T_{dec} = 1$ and $\mu = 5$.

2.3 Shadow and Highlight Detection

Brightness variations and lighting changes in parts of the image are the biggest challenge faced by background subtraction based segmentation algorithms. The shadows induced by vehicles themselves may be segmented as part of vehicles, which can not only interfere with information regarding size and shape but also merge vehicles in a single blob (see Fig. 2.11). This can affect many higher level surveillance tasks, such as counting, trajectory analysis, and classifying individual objects in the scene. The shadows induced by the clouds can generate mis-classification of pixels as foreground due to the fast lighting variation of the scene lighting when the clouds are moving fast. The glares induced on the surface of the highway by vehicles night light system and artificial lighting variation also represent a problem for a robust outdoor vehicle segmentation process.

Essentially, a shadow is generated by a change of the illumination conditions (lighting variation, light source occlusion, etc.). Shadow detection comes down to a problem of finding illumination invariant features. In outdoor scenarios, finding illumination invariant features can be very challenging. Geometrically, the shadow can be classified as *umbra* and *penumbra* [70] (see Fig. 2.12). The *umbra* corresponds to the background area where the direct light is almost totally blocked by a foreground object, whereas in the *penumbra* area, the light is partially blocked. From the viewpoint of motion property, a shadow can be divided into static shadow and dynamic shadow. A static shadow is cast by a static object, while a dynamic shadow is cast by a moving object. In most of the video surveillance applications, static shadows are not a problem for a segmentation process (usually, static shadows are integrated into the background model, this way, they are not mis-detected as a foreground object). For this reason, the study here presented on shadow detection focus on the detection of moving cast shadows.

For a given image, $I(\phi)$, the intensity of the pixels can be given as Eq. 2.27, where $i(\phi)$ represents the illumination component and $r(\phi)$ represents the reflectance component of the scene components [70].

$$I(\phi) = i(\phi) + r(\phi) \tag{2.27}$$

(a) (b)

Figure 2.11: Example of a shadow problem in an outdoor scenario, the vehicle's shape and area change significantly. a) Bounding box of the vehicles detected with the proposed segmentation system. b) Foreground pixels detected with the proposed segmentation system (green pixels).

The illumination component, $i(\phi)$, is computed as the amount of light power per surface area of the receiving object and can be further expressed by Eq. 2.28, where c_p is the intensity of the light source, α is the angle enclosed by light source direction and surface normal, c_a is the intensity of the ambient light and t is the transition inside the *penumbra* area which depends on the light source, scene geometry, and $0 \leq t(\phi) \leq 1$.

$$i(\phi) = \begin{cases} c_a + c_p.\cos(\alpha) & \textit{illuminated area} \\ c_a + t(\phi).c_p.\cos(\alpha) & \textit{penumbra area} \\ c_a & \textit{umbra area} \end{cases} \tag{2.28}$$

2.3.1 Shadow Detection Algorithms

Many methodologies have been presented in the literature for moving shadow detection. These methodologies can be classified into three main categories:

Color Analysis The color-based analysis methodologies try to model the color change of shaded pixels and to find color features that are illumination invariant [72, 72, 66, 22, 44, 62]. Several color space transformations have been proposed to better find those discriminant features. The pixel-level analysis, in a general way, is very sensitive to image noise.

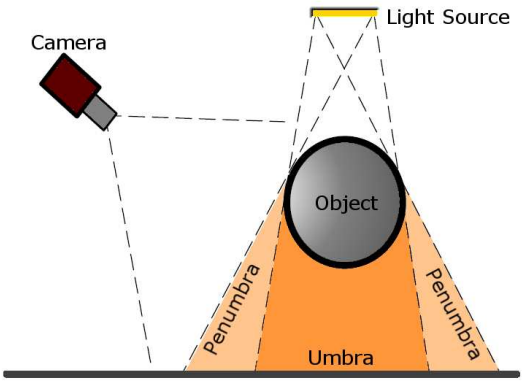


Figure 2.12: The geometric relationship of a moving cast shadow.

Texture Analysis The principle behind texture-based analysis methodologies is that the texture of shaded areas remains the same as that of the background, while the texture of foreground objects is different from that of the background [29, 55, 75, 77, 43, 30]. These methodologies are characterized by a region-level analysis in spite of a pixel-level, therefore decreasing the sensitiveness to image noise.

Geometrical Analysis Geometry-based shadow detection methodologies use camera location, ground surface, object geometry, etc., to identify moving cast shadows [39, 40, 76]. This kind of approaches, in a general way, require some prior information, like camera location, object geometry, etc.

Most of the state-of-the-art algorithms are based on the reference image – background model image of the segmentation process. Let the reference image and current analyzed image be B and I, respectively.

Siala et al. [66] consider the pixel's intensity change equally in RGB color components and a diagonal model is proposed to describe the color distortion of shadow in RGB space. The color distortion is defined as ($d_R = I_R/B_R$, $d_G = I_G/B_G$, $d_B = I_B/B_B$), and the color distortion of shadow pixels is distributed near the line $dR = dG = dB$, which does not occur with foreground objects. Therefore, the shadow pixels are discriminated from foreground objects according to the distance between pixel's color distortion and the line $dR = dG = dB$.

Salvador et al. [64] proposed a normalized RGB color space, rgb, to segment the shadows in still images and video sequences. The r, g and b components are given by Eq. 2.29, Eq. 2.30, and Eq. 2.31, respectively.

$$r = \arctan \frac{R}{\max(R, G, B)} \tag{2.29}$$

$$g = \arctan \frac{G}{\max(R, G, B)} \tag{2.30}$$

$$b = \arctan \frac{B}{\max(R, G, B)} \tag{2.31}$$

After integrating the intensity of neighboring regions, the shadow is detected as the pixels change greatly in rgb color space. Considering that the imtensity in RGB color space decreases in a same scale in shadowed pixels, it can be found that rgb is illumination invariant.

Cavallaro et al. detected shadows by exploiting color information in a selective way. In each image the relevant areas are identified and the color components that carry most of discriminating information are selected for shadow detection. The color model has shown its powerfulness in shadow detection. Nevertheless, the foreground objects may have the same color as the moving shadows, and it is not reliable to detect moving shadows by using only the color information of the isolated points.

Horprasert et al. [38] proposed a computational color model which separates brightness from the chromaticity component using brightness distortions $\alpha(\phi)$ and chromaticity distortions $CD(\phi)$, which are defined as follows. This work considers the color constancy ability of human eyes and exploits the Lambertian hypothesis (objects with perfectly matte surfaces) to consider color as a product of irradiance and reflectance. The algorithm is based on pixel modeling and background subtraction. First, the background model is learned, i.e., the means and the variances of each color channel. Where $E(\phi) = [\mu_R(\phi); \mu_G(\phi); \mu_B(\phi);]$ is the mean vector and $s(\phi) = [\sigma_R(\phi); \sigma_G(\phi); \sigma_B(\phi);]$ is the variance vector. The distortion of the brightness $\alpha(\phi)$ is given by Eq. 2.32. The distortion of the chrominance $CD(\phi)$ of the difference between expected color of a pixel and its value in the current image $I(\phi) = [I_R(\phi); I_G(\phi); I_B(\phi)]$ is given by Eq. 2.33.

$$\alpha(\phi) = \frac{\left(\frac{I_R(\phi)\mu_R(\phi)}{\sigma_R^2(\phi)} + \frac{I_G(\phi)\mu_G(\phi)}{\sigma_G^2(\phi)} + \frac{I_B(\phi)\mu_B(\phi)}{\sigma_B^2(\phi)}\right)}{\left(\left[\frac{\mu_R(\phi)}{\sigma_R(\phi)}\right]^2 + \left[\frac{\mu_G(\phi)}{\sigma_G(\phi)}\right]^2 + \left[\frac{\mu_B(\phi)}{\sigma_B(\phi)}\right]^2\right)} \tag{2.32}$$

$$CD(\phi) = \sqrt{\left(\frac{I_R(\phi) - \alpha(\phi)\mu_R(\phi)}{\sigma_R(\phi)}\right)^2 + \left(\frac{I_G(\phi) - \alpha(\phi)\mu_G(\phi)}{\sigma_G(\phi)}\right)^2 + \left(\frac{I_B(\phi) - \alpha(\phi)\mu_B(\phi)}{\sigma_B(\phi)}\right)^2} \tag{2.33}$$

The obtained $\alpha(\phi)$ and $CD(\phi)$ are normalized w.r.t their root mean square for pixel ϕ to obtain $\widehat{\alpha(\phi)}$ and $\widehat{CD(\phi)}$. The pixel classification, $C(\phi)$, is given by Eq. 2.34, where $\tau_{\alpha 1}$, $\tau_{\alpha 2}$, τ_{CD}, and $\tau_{\alpha lo}$ are system thresholds.

$$C(\phi) = \begin{cases} Foreground & if \widehat{CD(\phi)} > \tau_{CD} \quad or \quad \widehat{\alpha(\phi)} < \tau_{\alpha lo}, \quad else \\ Background & if \widehat{\alpha(\phi)} < \tau_{\alpha 1} \quad and \quad \widehat{\alpha(\phi)} > \tau_{\alpha 2}, \quad else \\ Shadow & if \widehat{\alpha(\phi)} < 0, \quad else \\ Highlight & otherwise \end{cases} \tag{2.34}$$

In [75], several techniques have been developed by Xu et al. to detect moving cast shadows in a normal indoor environment. These techniques include the generation of initial change detection masks and canny edge maps, the detection of shadow region by multi-frame integration, edge matching, conditional dilation, and post-processing.

McKenna et al. [55] assumed that cast shadows result in significant change in intensity without changing chromaticity. Each pixel's chromaticity is modeled using its means and variances, and each background pixel's first-order gradient is modeled by using gradient means and magnitude variances. The moving shadows are then classified as background if the chromaticity or gradient information supports their classification.

Zhang et al. [77] used the normalized coefficients of the orthogonal transformation for moving shadow detection. Five kinds of orthogonal transforms (DCT, DFT, Haar Transform, SVD, and Hadamard Transform) are analyzed, and their normalized coefficients are proved to be illumination invariant in a small image block. The cast shadows are then detected by using a simple threshold on the normalized coefficients.

In [39], a Gaussian shadow model was proposed to detect the shadows of pedestrian by Hsieh et al. The model is parameterized with several features

including the orientation, mean intensity, and center position of a shadow region, with the orientation and centroid position being estimated from the properties of object moments.

Hsieh et al. [40] proposed a histogram-based method to detect different lane dividing lines from traffic video sequence. According to these lines, a line-based shadow modeling process is applied to eliminate vehicle's shadows. Two kinds of lines are used, including the ones parallel and vertical to lane directions, which can be used to eliminate shadows in the different positions of the vehicles.

Yoneyama et al. [76] proposed joint 2D vehicle/shadow models to suppress the moving shadows of vehicles. The proposed 2D vehicle/shadow models are classified into six types and the parameters of these models can be estimated by fitting the segmented vehicles with these models.

Color-based methodologies have shown its powerfulness in shadow detection. Nevertheless, foreground objects may have the same color as the moving shadows, and it is not reliable to detect moving shadows by using only the color information of the isolated points. Texture-based methodologies may be the most promising technique for shadow detection, whereas the state-of-the-art of textural model are intricate in implementation. Moreover, in the homogeneous regions of the images, the textural information might not be reliable. The geometry-based methodologies strongly depend on the geometric relationships of the objects in the scenes, and when these geometric relationships change, these methods become ineffective. Methodologies based on geometrical information will not be taken into consideration for this analysis due to the prior information required.

In order to find a solution that better satisfies the requirements of an outdoor application, two state-of-the-art methodologies to identify shadow pixels were implemented and tested: a color-based algorithm proposed by Cucchiara et al. [22] and a texture-based algorithm proposed by Grest el al. [30]. These methodologies will be described below.

HSV Color Representation Based Shadow Detection

A color-based methodology was presented by Cucchiara et al. [22] to identify shadowed pixels. The Hue-Saturation-Value (HSV) color space (see Fig. 2.13)

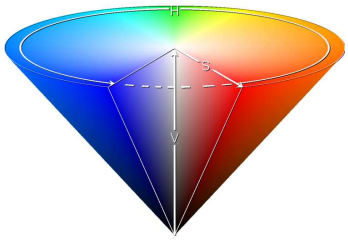


Figure 2.13: The conical representation of the HSV color space.

was used to separate the chromaticity and the luminosity in the pixel color information.

Some considerations are taken into account to verify if segmented pixels are classified as shadow. First, if a shadow is cast on a background, the hue component changes, but within a certain limit. Additionally, the saturation component also changes within a certain limit. The difference in saturation must be an absolute difference, while the difference in hue is an angular difference. A pixel in the position ϕ is classified as shadow if the condition defined in Eq. 2.35 is verified, where $D_H^t(\phi)$ is given by Eq. 2.36, $I^t(\phi)$ is the analyzed frame, and $B^t(\phi)$ is the background model at the instant t. The $.H$, $.S$, and $.V$ denotes, respectively, the H, S, and V components of the HSV color space representation. The thresholds of the system are: α, β, τ_S, and τ_H.

$$\alpha \leq \frac{I^t(\phi).V}{B^t(\phi).V} \leq \beta \ \wedge \ \left|I^t(\phi).S - B^t(\phi).S\right| \leq \tau_S \ \wedge \ D_H^t(\phi) \leq \tau_H \qquad (2.35)$$

$$D_H^t(\phi) = min\left(\left|I^t(\phi).H - B^t(\phi).H\right|,\ 360 - \left|I^t(\phi).H - B^t(\phi).H\right|\right) \qquad (2.36)$$

In the luminance analysis, the threshold α is used to define a maximum value for the darkening effect of shadow on the background, and is approximately proportional to the light source intensity. On the other hand, the threshold β prevents the system from identifying as shadow those points where the background was darkened too little with respect to the expected effect of shadows. A preliminary sensitivity analysis for α, β, τ_S, and τ_H is reported in [21].

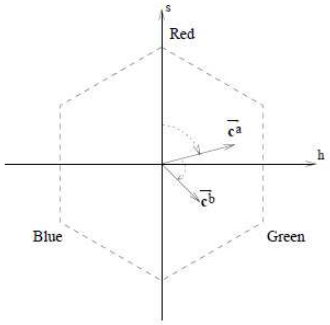


Figure 2.14: The representation of the chromatic plane of the hsL color space.

Shadow Detection Based on Color Normalized Cross-Correlation

This methodology uses the Color Normalized Cross-Correlation (CNCC) to measure the similarity between a shadowed region and the background model of the same region [30]. A shadow is identified if the two image areas that were compared are cross-correlated. This is a texture-based methodology, but it also uses color information to identify the shadowed areas.

The similarity between the background model, $B^t(\phi)$, and the current frame, $I^t(\phi)$, is measured by computing the color normalized cross-correlation. The color of the pixel is converted into the biconic HSL space (see Fig. 2.14) in order to split the color information from the brightness values. To measure similarity, the HSL color space is calculated, not in polar coordinates, but through the projection of the (R, G, B) vector onto the chromatic (H, S)-plane to compute the Euclidean values of hue and saturation. It is denoted the representation in Euclidean coordinates with (h, s). The projected h, s part is scaled, so that its length equals the saturation of the HSL color space. It is also denoted that $c^I = (h^I, s^I, L^I)$ is the pixel components for a foreground pixel and c^B the same for the background image. It is defined on Eq. 2.37 the CNCC over a window sized $M \times M$ for the two color pixels $c^I_{x,y}$ $c^B_{x,y}$ at an image position (x,y). VAR^k is defined by Eq. 2.38 where i ranges from $x - \frac{M-1}{2}$ to $x + \frac{M-1}{2}$ and j from $y - \frac{M-1}{2}$ to $y + \frac{M-1}{2}$, $\overline{L^I}$ is the average intensity in the image I over the $M \times M$ window, $k \in \{I, B\}$ and $c^I_{x,y}$ defined by Eq. 2.39.

$$CNCC_{x,y} = \frac{\sum_{i,j}(c^I_{x,y} \bullet c^B_{x,y}) - M^2 \overline{L^I}\ \overline{L^B}}{\sqrt{VAR^I VAR^B}} \tag{2.37}$$

$$VAR^k = \left(\sum_{i,j} (c^k_{x,y} \bullet c^k_{x,y}) - M^2 \overline{L^k}^2 \right) \quad (2.38)$$

$$c^I_{x,y} \bullet c^B_{x,y} = (h^I_{i,j}, s^I_{i,j}) \circ (h^B_{i,j}, s^B_{i,j}) + L^I_{i,j} L^B_{i,j} \quad (2.39)$$

The operator $\circ$ denotes the scalar product, with negative values set to zero. This operation regards the fact that two colors with an hue angle of more than 90 degrees between them are interpreted as different by humans. A pixel (i, j) is classified as shadow if $CNCC_{x,y} > \varsigma$, where ς is a fixed threshold.

2.3.2 Experimental Results

In order to identify an algorithm that robustly detects shadows in outdoor scenarios, some state-of-the-art shadow detection algorithms were studied (see Sec. 2.3.1). A color-based shadow and a texture-based shadow detection algorithm were selected for the evaluation tests. The Shadow Detection Based on HSV Color Representation and the Shadow Detection Based on Color Normalized Cross-Correlation were subject to several tests in order to verify the behavior of these two algorithms in outdoor scenarios. Some comparative results of these two algorithms are shown in Fig. 2.16. An example of sudden illumination changes detection in a tunnel scenario is also shown in Fig. 2.17.

Some experimental tests were performed to evaluate the performance of the two methodologies to detect shadowed pixels. Fifty frames of four different sites with shadows casted by the vehicles (depicted in Fig. 2.15) were used to perform these tests.

A shadow detection algorithm, in a general way, is used to improve the detection of segmented pixels. This way, it is not relevant to analyze the shadow detection algorithm performance in background regions. To take this into account, and to evaluate the performance of each tested methodology, it was obtained the *shadow detection accuracy* η given by Eq. 2.40, and the *shadow discrimination accuracy* ξ given by Eq. 2.41, where TP_S is the number of pixels correctly classified as shadow pixels, FP_S is the number of wrongly classified shadow pixels (foreground object pixels), $Total_S$ is the total number of shadowed pixels, and $Total_F$ is the total number of pixels belonging to foreground

(a) Site 1 – Outdoor scenario (b) Site 2 – Tunnel scenario

(c) Site 3 – Tunnel scenario (d) Site 4 – Outdoor scenario

Figure 2.15: Scenarios used to compare the texture-based algorithm with the color-based algorithm to detect shadowed pixels.

objects.

$$\eta = \frac{TP_S}{Total_S} \tag{2.40}$$

$$\xi = \frac{FP_S}{Total_F} \tag{2.41}$$

The results of the tests performed on the two shadow detection algorithms with the two measures presented above are presented in Table 2.1.

The tested Color-Based shadow detection algorithm is a good shadow detector in colorized areas, but it does not accurately detect lighting variations in image areas with low color information (e.g.: in Fig. 2.16(g) the black vehicle area is completely classified as shadow due to the similarity between the color information in the image and in the background). In a general way, the Texture-Based shadow detection algorithm demonstrates to be more robust than the Color-Based one. The algorithm proved to be more accurate and robust for the purpose. Therefore, it was adopted to detect shadows and highlights in the segmented blobs.

(a) Original Image (b) Color-Based (c) Texture-Based

(d) Original Image (e) Color-Based (f) Texture-Based

(g) Original Image (h) Color-Based (i) Texture-Based

(j) Original Image (k) Color-Based (l) Texture-Based

Figure 2.16: Shadow detection in outdoor scenarios. Left column – Captured frame; Middle column – Color-Based shadow detection; Right column – Texture-Based shadow detection. The shadowed pixels are identified with the red color.

	Texture-based		Color-based	
	η	ξ	η	ξ
Site 1	0.85	0.11	0.62	0.24
Site 2	0.84	0.09	0.80	0.07
Site 3	0.94	0.14	0.95	0.25
Site 4	0.89	0.02	0.66	0.19

Table 2.1: Comparative results of the tests performed with texture-based algorithm and the color-based algorithm to detect shadowed pixels.

(a) Original Image (I^{t-1})

(b) Original Image (I^{t})

(c) Color-Based

(d) Texture-Based

Figure 2.17: Sudden illumination change detection in a tunnel scenario. a) two consecutive frames of the sudden illumination change; c) Color-Based shadow detection algorithm; d) Texture-Based shadow detection algorithm.

2.4 Image Flow Estimation

In this section it will be discussed the problem of the image motion estimation, also called optical flow. The optical flow estimation can be described as the displacement estimation of a given set of features between two consecutive frames. This methodology can provide useful information about vehicles (velocity, motion direction, etc.) moving on the road.

Unlike the foreground segmentation process presented in Sec. 2.2, the optical flow estimation does not require any scenario model, just two consecutive frames are required. This way, the motion estimation process is very robust to different weather conditions. The major drawback of this process is the dependence on reliable features in the image to estimate the motion. Only points with significant edge information are analyzed. Therefore, it is not possible to obtain the complete shape of the moving vehicles in the scene.

Optical flow techniques are widely used in computer vision applications. The goal is to compute an approximation to the 2-d motion field – a projection of the 3-d displacements of surface points onto the imaging surface – from spatiotemporal patterns of image intensity [1]. Many methods have been proposed to robustly compute optical flow. Those methods can be divided into four main categories:

Block-Based Matching Such approaches define displacement as the shift that yields the best fit between image regions at different times. Finding the best match amount that maximizes the similarity measure (normalized cross-correlation, sum-of-squared difference, etc.) [25, 5, 49, 68].

Energy-Based These optical flow techniques are based on the output energy of velocity-tuned filters. These are also called frequency-based methods owing to the design of velocity-tuned filters in the Fourier domain [34, 2, 7, 12].

Phase-Based In those methodologies, the displacement is defined in terms of the phase behavior of band-pass filter outputs [4, 26, 35, 24].

Differential Techniques Differential techniques compute the displacement from spatiotemporal derivatives of image intensity of filtered versions of the im-

age [37, 32, 74, 50].

A valuable comparison of different optical flow techniques is presented in [8]. The differential technique proposed by Lucas and Kanade [50] achieved an overall better performance in the comparative tests performed in [8]. Therefore, it was adopted to estimate the vehicle's motion.

The information provided by this algorithm will be used in some different parts of this Automatic Traffic Surveillance System, such as: blob validation in the foreground segmentation system (Sec. 2.2), and vehicle motion estimation in the wrong way driver detection system (Sec. 3.3).

2.4.1 Optical Flow Estimation

The differential methodology adopted to estimate the vehicle's motion is based on the differential technique proposed by Lucas and Kanade [50]. This methodology makes use of the spatial-intensity gradient of the image to find a good match using a type of Newton-Raphson iteration. Several improvements have been proposed to this methodology in order to increase robustness, accuracy, and reliability of the algorithm in real scenarios. A probabilistic formulation, presented by Simoncelli et al. [67], provides an objective measurement of the local level of reliability of the motion information in order to reject outlier measurements. This probabilistic formulation compensates the noise in the image derivative computation. Shi and Tomasi [65] proposed a numerically sound and efficient way of determining affine changes by a Newton-Raphson minimization procedure.

Let I^{t-1} and I^t be two consecutive grayscale frames of an image sequence, where, $I(X) = I(x, y)$ is the grayscale value of the image at the pixel position (x, y). Given an image point $\mathbf{u} = [u_x, u_y]^T$ on the first image I^{t-1}, the goal of the feature tracking algorithm is to find the location $\mathbf{v} = \mathbf{u} + \mathbf{d}$ on the second image I^t such as $I^{t-1}(\mathbf{u})$ and $I^t(\mathbf{v})$ are similar. The vector $\mathbf{d} = [d_x, d_y]^T$ is the image velocity at $\mathbf{u}$, also known as the optical flow at $\mathbf{u}$. The image velocity $\mathbf{d}$ is defined as being the vector that minimizes the residual function ϵ defined by Eq.2.42. The similarity function is measured on a image neighborhood of size $(2\omega + 1) \times (2\omega + 1)$. This neighborhood is also called integration window.

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega}^{u_x+\omega} \sum_{x=u_y-\omega}^{u_y+\omega} \left(I^{t-1}(x,y) - I^t(x+d_x, y+d_y)\right)^2 \quad (2.42)$$

At the optimum, the first derivative of ϵ with respect to $\mathbf{d}$ is zero (Eq. 2.43). After the expansion of the derivative it is obtained Eq. 2.44, where ∇I is an image gradient vector and is given by Eq. 2.45.

$$\left.\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}}\right|_{\mathbf{d}=\mathbf{d}_{opt}} = [0\ 0] \quad (2.43)$$

$$\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}} = -2 \sum_{x=u_x-\omega}^{u_x+\omega} \sum_{x=u_y-\omega}^{u_y+\omega} \left(I^{t-1}(x,y) - I^t(x+d_x, y+d_y)\right) . \nabla I^T \quad (2.44)$$

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I^t}{\partial x} & \frac{\partial I^t}{\partial y} \end{bmatrix}^T \quad (2.45)$$

Assuming a small displacement vector $\mathbf{d}$, $I^t(x+d_x, y+d_y)$ is substituted by the first order Taylor expression in Eq. 2.46, where δI is given by Eq. 2.47

$$\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}} \approx -2 \sum_{x=u_x-\omega}^{u_x+\omega} \sum_{x=u_y-\omega}^{u_y+\omega} \left(\delta I - \nabla I^T . \mathbf{d}\right) . \nabla I^T \quad (2.46)$$

$$\delta I = I^{t-1}(x,y) - I^t(x,y) \quad (2.47)$$

Redefined Eq. 2.46 it is obtained Eq. 2.48.

$$\frac{1}{2}\left[\frac{\partial \epsilon(\mathbf{d})}{\partial \mathbf{d}}\right]^T \approx \sum_{x=u_x-\omega}^{u_x+\omega} \sum_{x=u_y-\omega}^{u_y+\omega} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} . \mathbf{d} - \begin{bmatrix} \delta I\ I_x \\ \delta I\ I_y \end{bmatrix} \right) \quad (2.48)$$

Denote

$$G = \sum_{x=u_x-\omega}^{u_x+\omega} \sum_{x=u_y-\omega}^{u_y+\omega} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (2.49)$$

and

$$\mathbf{b} = \sum_{x=u_x-\omega}^{u_x+\omega} \sum_{x=u_y-\omega}^{u_y+\omega} \left(I^{t-1}(x,y) - I^t(x,y)\right) . \nabla I^T \quad (2.50)$$

Then, Eq. 2.48 may be written by Eq. 2.51.

$$\frac{1}{2}\left[\frac{\partial\epsilon(\mathbf{d})}{\partial\mathbf{d}}\right]^T \approx G.\mathbf{d} - \mathbf{b} \tag{2.51}$$

Therefore, following the Eq. 2.43, the optimum optical flow vector is given by Eq. 2.52.

$$\mathbf{d}_{opt} = G^{-1}.\mathbf{b} \tag{2.52}$$

The two key components of any feature tracker are accuracy and robustness. A small integration window would be preferable in order not to smooth out image details for an accurate feature tracking. On the other hand, a large integration window would be preferable to robustly track features with lighting changes, and in particular, to handle with large motions. In order to provide a solution for this trade-off, a pyramidal implementation of the classical Lucas-Kanade algorithm was proposed by Bouguet [13]. This pyramidal implementation of the classical Lucas-Kanade algorithm was used to estimate the vehicle's motion.

2.4.2 Features to Track Selection

The features to track selection procedure is fundamental to achieve an accurate tracker. Shi and Tomasi [65] proposed a reliability criterion to evaluate the texture properties of images areas in order to select good features to track.

A window centered in (x, y) is considered a good feature to track if the symmetric 2×2 matrix Z is both above the image noise level and well-conditioned. The matrix Z is defined by Eq. 2.53, where the summations are intended over a small spatial neighborhood Ω of the pixel, $W(x)$ is a window function that gives more influence to pixels in the center of the neighborhood, and I_x and I_y are the spatial gradients of the gray levels in directions x and y respectively. A neighborhood, Ω, of 3×3 pixels is sufficient for the selection of features.

$$Z = \begin{bmatrix} \sum_{x\in\Omega} W^2(x)I_x^2(x) & \sum_{x\in\Omega} W^2(x)I_x(x)I_y(x) \\ \\ \sum_{x\in\Omega} W^2(x)I_x(x)I_y(x) & \sum_{x\in\Omega} W^2(x)I_y^2(x) \end{bmatrix} \tag{2.53}$$

The noise requirement implies that both eigenvalues of Z must be high, while the conditioning requirement means that they cannot differ by several orders of magnitude. Two high eigenvalues can represent corners, salt-and-pepper textures, or any other pattern that can be reliably tracked. In practice, when the smaller eigenvalue is sufficiently high to meet the noise criterion, the matrix Z is usually also well conditioned. This way, a window is accepted for tracking if the condition defined by Eq. 2.54 is verified, where λ_1 and λ_2 are the two eigenvalues of Z and δ is a predefined threshold. If the condition is not verified, no displacement value is assigned to the pixel.

$$min(\lambda_1, \lambda_2) > \delta \tag{2.54}$$

The algorithm 2 is used to obtain the reliable feature points to track in image $I^{t-1}(\phi)$. After this process, the remaining pixels are typically good points to track. They are the selected feature points that are fed to the tracker.

Algorithm 2 Estimate good features to track in $I^{t-1}(\phi)$

1: Compute the Z matrix and λ_m at every pixel in the image $I^{t-1}(\phi)$, where $\lambda_m = min(\lambda_1, \lambda_2)$.
2: Obtain the maximum value of λ_m over the whole image, λ_{max}.
3: Retain the image pixels that have a λ_m value that verifies the condition: $\lambda_m > max(\delta,\ \tau \times \lambda_{max})$, where τ is a fixed threshold and $\tau \in [0, 1]$.
4: From those pixels, retain the local maximum pixels, i.e., a pixel is kept if its λ_m value is larger than that of any other pixel in its 3×3 neighborhood.
5: Keep the subset of those pixels so that the minimum distance between any pair of pixels is larger than ξ, where ξ is a fixed threshold (e.g. 5–10 pixels).

2.4.3 Experimental Results and Conclusions

In this section a methodology to estimate motion between two consecutive frames was described. For this purpose, the differential technique presented by Lucas and Kanade was used. In order to find good features to track in the image a criterion of reliability to evaluate the texture properties of images areas proposed by Shi and Tomasi was used.

The optical flow algorithm described here was tested using several real datasets from highways traffic surveillance cameras under different weather conditions, illumination, image quality and fields of view. This system – integrated in the wrong way driver detection system presented in Chapter 3 – has also been running in ten different sites of Portuguese highways for fifteen months. During this evaluation period, the presented system demonstrated it's robustness and accuracy under the different outdoor situations like different weather conditions, lighting changes, low image quality (e.g.: video compression and analog video transmission) and vehicle's motion estimation in crowded situations.

Some examples of optical flow estimation in outdoor scenarios are shown in Fig. 2.18. Note that displacement vectors with less than one are discarded and that the presented images are a merge between the two consecutive frames used to estimate the optical flow.

In a general way, the system just fails under low contrast scenario situations and in situations where the image gets blurred due to the camera's auto-focus system or due to adverse weather conditions, like fog. In these situations, two different things could occur: a) no good feature to track is found in the vehicle area, this way no motion information is estimated for the vehicle; or b) for a given feature, no correspondence is found in the I^{t+1} and a wrong flow vector is estimated.

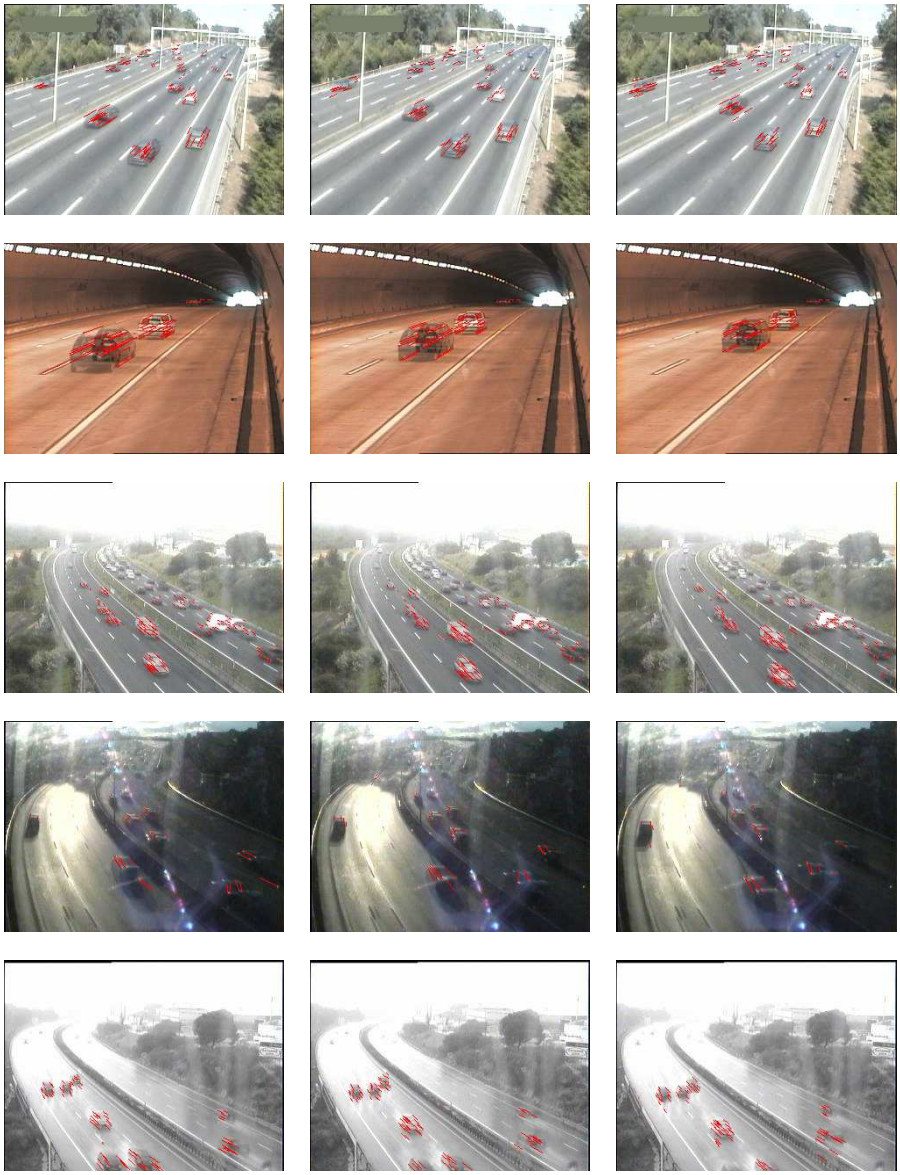


Figure 2.18: Optical flow estimation with the presented methodology in some different outdoor scenarios. These examples are: a) Regular scenarios; b) Tunnel scenario. c) Fog situation; d) Low contrast scenario and sun light in the direction of the camera; and e) Rainy day.

2.5 Experimental Results

The proposed system was tested with a real set of image sequences from highway traffic surveillance cameras under different weather conditions, lighting, image quality and fields of view. It is also being tested in three different sites of Portuguese highways for six months now. The segmentation process is able to detect vehicles and to deal with outdoor adverse weather condition and illumination changes in a robust way. The system can segment vehicles in outdoor scenarios over a 320×240 pixel image at 18 fps on a 3.2 GHz *P4 Intel Processor* under Linux OS. Some results of the segmentation process are depicted in Fig. 2.19, 2.20, 2.21, 2.22, 2.23, and 2.24. On the left column, vehicles are identified with a green bounding box. The minimum difference between the current frame and the three background models is represented on the center column, and, on the right column, the foreground pixels are identified with the green color.

(a) Detected vehicles. (b) Differences image. (c) Foreground pixels.

Figure 2.19: Segmentation process in a typical outdoor scenario.

2.5.1 Segmentation Process Robustness and Accuracy

In order to estimate the robustness and the accuracy of the presented foreground/background segmentation methodology some tests were performed using the dataset and the rules available at VSSN 2006 website[1].

To obtain robust assessment of performance, the proposed methodology was evaluated against different categories of test sequences. The test datasets includes the following problems: a) sudden illumination changes; b) moving background objects; c) gradual illumination changes; and d) training period with foreground objects. The composition of this dataset is described in Table 2.2. The algorithm was not tested in Video 1 and in Video 7 because we were unable to use the provided ground truth.

For each test video the average of false alarms pixels and missed foreground pixels per video frame were calculated using the ground truth video. In order to allow for small boundary errors, errors within two pixels of the boundary between foreground and background will not be counted. The performance evaluation, for each video, started after the initial training period. This training period was set to 10 seconds.

The proposed segmentation process was compared with two different methods with the same datasets and in the same conditions. These methods are: a) Mixture of Gaussians proposed by Stauffer and Grimson [71] and described in Sec. 2.1.1; and b) Improvement of SAKBOT system presented by Calderara et al. in [15] and described in Sec. 2.1.1. The comparative results[2] are shown in Fig. 2.25.

These comparative results prove that the proposed segmentation process is able to deal with outdoor adverse condition achieving good performance results. It robustly deals with moving background objects and background model estimation with foreground objects in the scene. The proposed method achieves an overall better performance when compared with MoG. When compared with SAKBOT system the proposed system achieves an overall similar results.

[1] VSSN 2006 – Algorithm competition in foreground/background segmentation web page: http://mmc36.informatik.uni-augsburg.de/VSSN06_OSAC/.

[2] The performance evaluation results of two compared methods was obtained in [15].

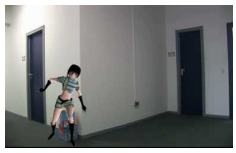
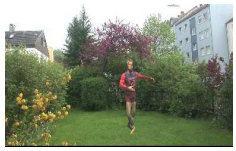
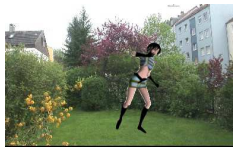
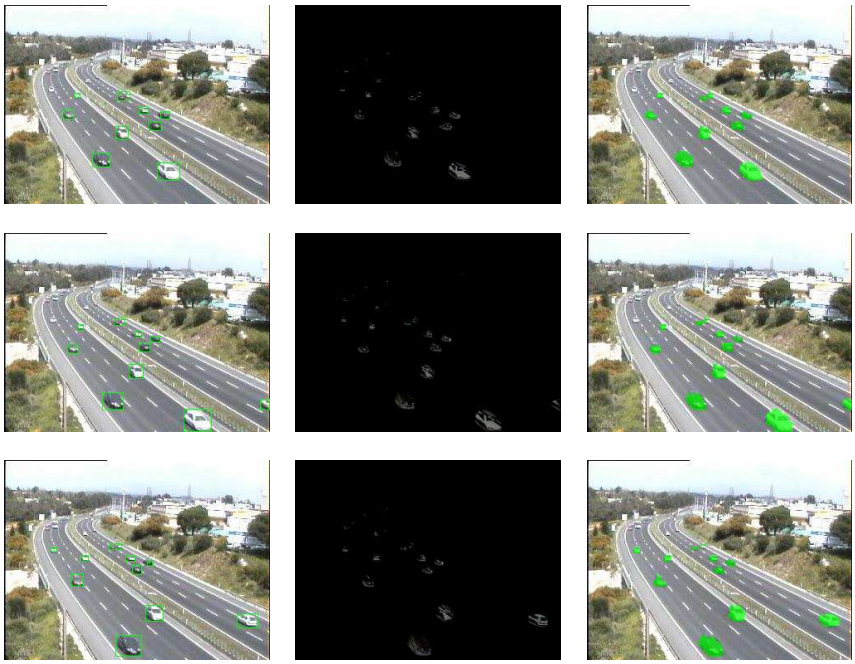
Video	Preview	secs.	fps	Description
Video 1		10	30	- indoor scenario
Video 2		16	30	- indoor scenario
Video 3		36	25	- outdoor scenario - moving background objects
Video 4		32	25	- outdoor scenario - moving background objects
Video 5		30	25	- indoor scenario
Video 6		30	25	- indoor scenario - training period with foreground objects
Video 7		30	25	- outdoor scenario - moving background objects
Video 8		47	25	-indoor scenario -illumination changes -training period with foreground objects

Table 2.2: Composition of the challenging datasets available at VSSN 2006 website and used to validate the proposed segmentation algorithm.

2.6 Conclusions

In this chapter, a methodology was proposed to segment vehicles in outdoor scenarios based on background subtraction. Three different background models were used to model all background variations and to avoid the degradation of the model. The system is able to efficiently estimate the initial background in high dense traffic situations and to update the model to work twenty four hours a day under different weather condition and lighting changes. Two distinct thresholds are used for a robust adaptation to the scene variations along the day. These thresholds are also initialized in the training period and updated in every frame. A shadow/highlight detection algorithm based on cross-correlation was also presented to discard the blobs or parts of blobs generated by lighting variation. In order to deal with abrupt changes variations, a frame-level analysis is performed. Thus,the system quickly adapts to scenario changes.

The experiments conducted on a large number of scenes of Portuguese highways and on the VSSN 2006 test dataset demonstrate that this system is able to robustly segment vehicles under different weather conditions, lighting, image quality and image compression variation. This segmentation process proved to be a good basis for an automatic traffic surveillance system.

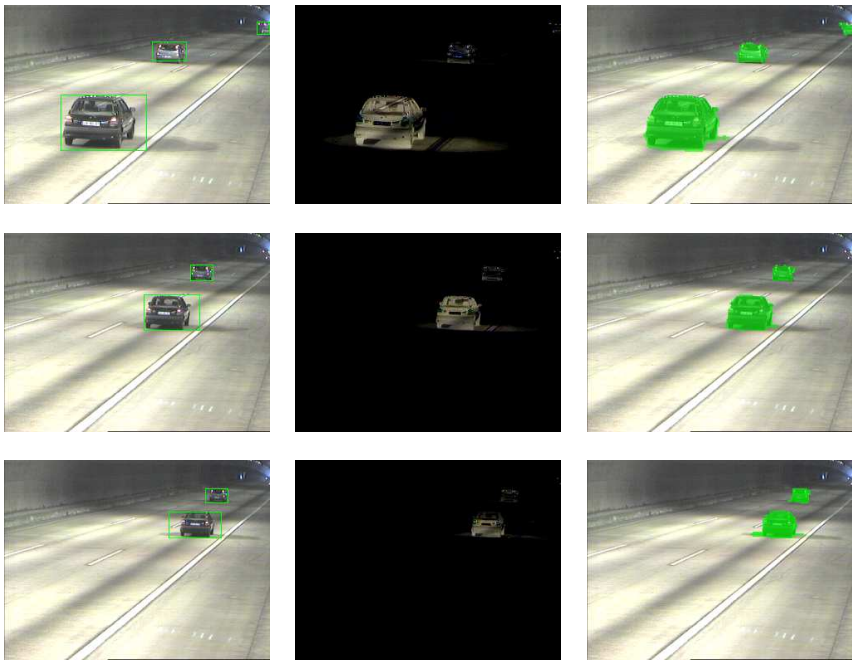


(a) Detected vehicles. (b) Differences image. (c) Foreground pixels.

Figure 2.20: Segmentation process in a typical outdoor scenario.

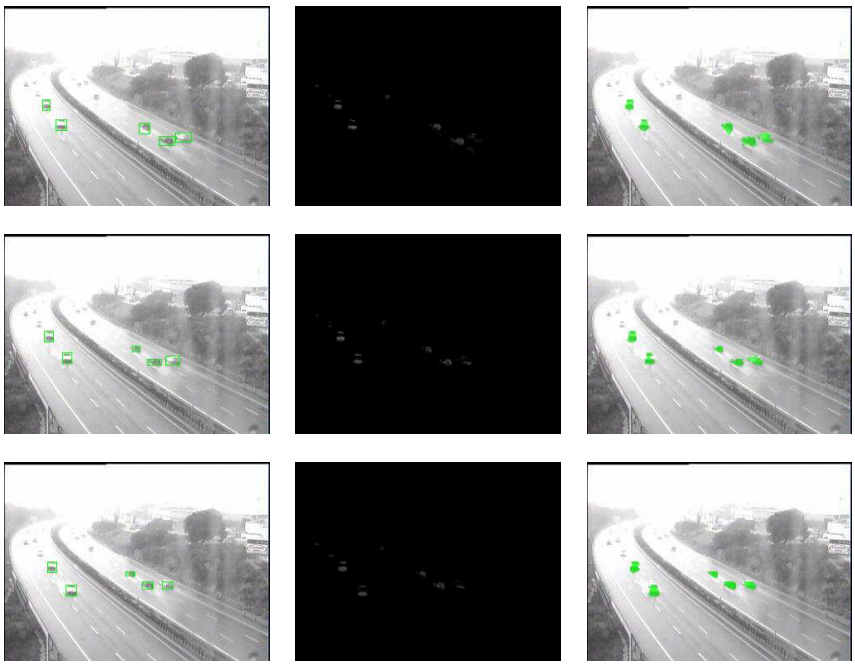
(a) Detected vehicles. (b) Differences image. (c) Foreground pixels.

Figure 2.21: Segmentation process in a typical outdoor scenario.



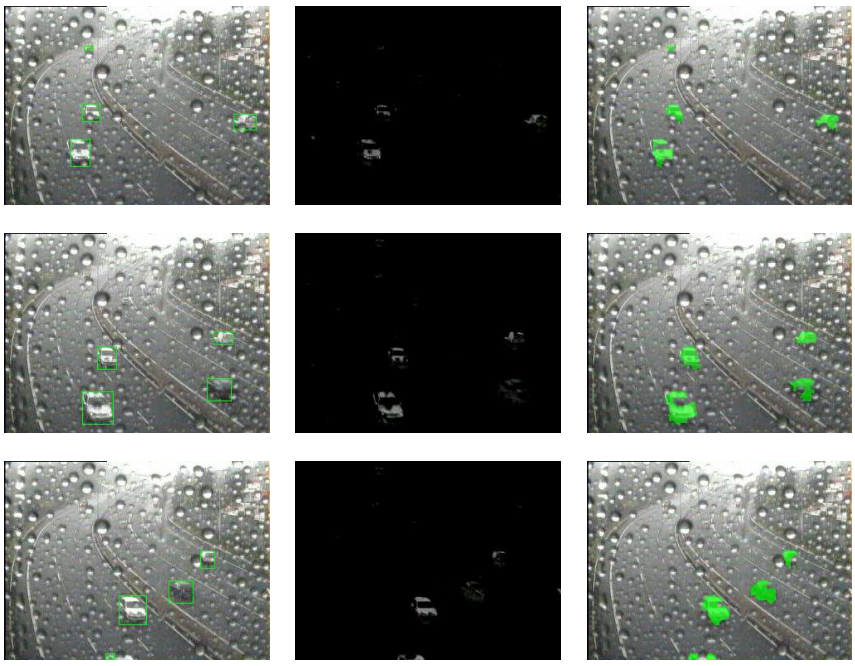
(a) Detected vehicles. (b) Differences image. (c) Foreground pixels.

Figure 2.22: Segmentation process in a tunnel scenario.



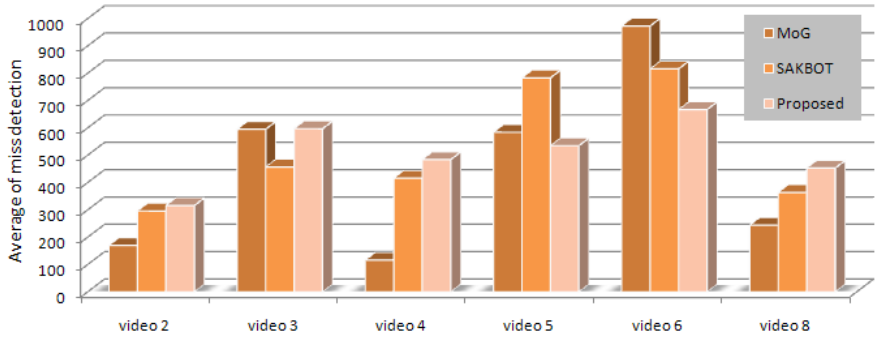
(a) Detected vehicles. (b) Differences image. (c) Foreground pixels.

Figure 2.23: Segmentation process in a rainy situation.

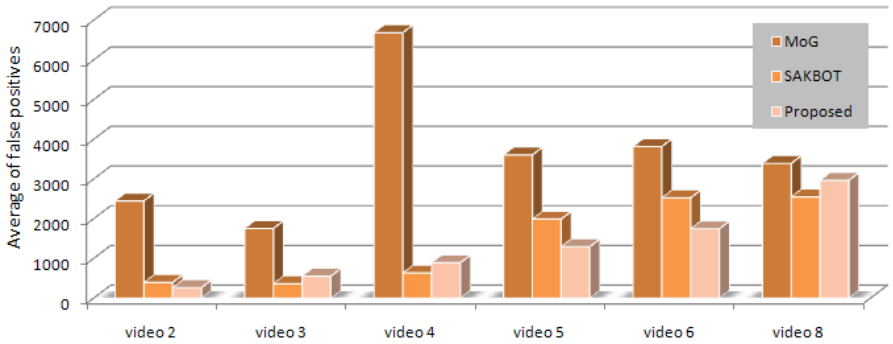


(a) Detected vehicles. (b) Differences image. (c) Foreground pixels.

Figure 2.24: Segmentation process with rain drops in the camera's lens



(a) Average of false negative pixels per frame for each test video.



(b) Average of false positive pixels per frame for each test video.

Figure 2.25: Comparative results between the proposed methodology, the mixture of Gaussians and the SAKBOT system improvement with the VSSN 2006 dataset.

Part II

Automatic Incident Detection

Chapter 3

Wrong Way Vehicle Detection

3.1 Introduction

Vehicles driving on the wrong way represent a serious threat that victimizes a considerable number of people every year across the world. This kind of situations is increasing every year proportionally to the volume of traffic. An immediate detection of a vehicle driving on the wrong direction could help prevent serious accidents. When detected, several actions can be taken, such as warning the oncoming vehicles (e.g.: via traffic telematic systems or radio announcements) and the responsible authorities.

The proposed system aims at automatically detecting drivers circulating on the wrong way and triggering an alarm on the highway traffic telematic system. The system must be robust to illumination changes and small camera movements, being able to robustly identify and track wrong way vehicles against occlusions and crowded events.

Several tests with real outdoor highway datasets proved the robustness of the proposed system. The system was also tested in several real scenarios of Portuguese highways during several months, and the results were also very promising.

3.1.1 Related Work

During the research on this topic, few relevant publications were found. In [42], Santner et al. present a wrong way driver detection system based on stereo vision. The disparity map is used to identify the vehicles circulating on the road. In a next step, it is performed the tracking of the identified vehicles in order to obtain the motion vectors which are classified depending on their direction. Finally, the motion vectors are analyzed to identify wrong way vehicles. The authors state that the false positives due to shadows and reflections are suppressed, this way, the false positive rate decreases significantly. Unfortunately, during the research period, no publications or results on this work were found. Using two cameras can be a considerable drawback if the system is going to be installed in an already existing video surveillance system. Another way to detect wrong way drivers is using a segmentation process (Sec. 2.2), tracking all segmented vehicles, and verifying if the direction of their trajectory is the correct one for the lane or if it is a vehicle circulating on the wrong side. This process has some disadvantages, namely tracking vehicles in crowded situations without grouping those circulating near each other.

3.2 Proposed Methodology

The proposed methodology is based on the information provided by an optical flow algorithm (Sec. 2.4). This optical flow information will be used to estimate the direction of the vehicle. This process is very robust to light and weather conditions variation. This system is based on three main stages. Firstly, the orientation pattern of the vehicle's motion flow is learned and modeled by a mixture of Gaussians. Then, there is a Detection Phase: it is verified, in each frame, if the direction of the areas where movement was detected matches the learned direction model. On both phases, a Block Median Filtering is applied to the motion flow in order to remove noisy data. Finally, a temporal validation is used to analyze the motion direction consistency and, this way, validate the object as a real wrong way vehicle. If the validation succeeds, an alarm is triggered. This validation process consists of tracking the wrong way regions over time and verifying if a coherent trajectory is made.

3.3 Vehicle's Motion Estimation

The estimation process of the vehicle's motion is the main part of the proposed wrong way driver detection system. In order to estimate the direction of the vehicle, θ, it is used the optical flow algorithm described in Sec. 2.4.

During the tests performed on this optical flow algorithm, satisfactory results were obtained when validating the foreground blobs. Some wrong flow vectors are estimated, but the effects on the final result of the segmentation process are not significant. However, for the purpose presented here, those noisy estimations can significantly decrease the performance of the system. Some examples of the referred optical flow estimation disturbances are shown in Figs. 3.1(a), 3.1(c), 3.1(e), and 3.1(g). These wrong estimations are mainly caused by image noise and motion flow discontinuity regions. A *Block Median Filtering* process is used to minimize this problem (Sec. 3.3.1).

3.3.1 Block Median Filtering

In order to filter motion direction (θ) a median filtering is used. This block analysis has also the advantage of reducing the volume of the analyzed information. When applying this filter, the image is divided into blocks of $n \times n$ pixels (see Fig. 3.1). For each block, the median of the directions obtained by the optical flow algorithm is calculated. Note that this is not a regular median operation, the direction θ is not linear ($\theta \in [-\pi, \pi]$).

To maximize the effectiveness of the filter, a high number of optical flow samples should be available. Therefore, a high value of n is desirable. On the other hand, a low value of n should be used so that no significant motion information resolution is lost. The motion information resolution is particularly important when the vehicle's area, in the image, is too small. A block size of 8 pixels ($n = 8$) proved to be a good value to achieve a suitable balance. This value was obtained empirically during several experiments conducted in outdoor scenarios.

From this moment on, every reference made to the motion direction estimation is related to the median motion flow of the block. Likewise, the flow detected in the image is analyzed for each block instead of a pixel-level analysis.

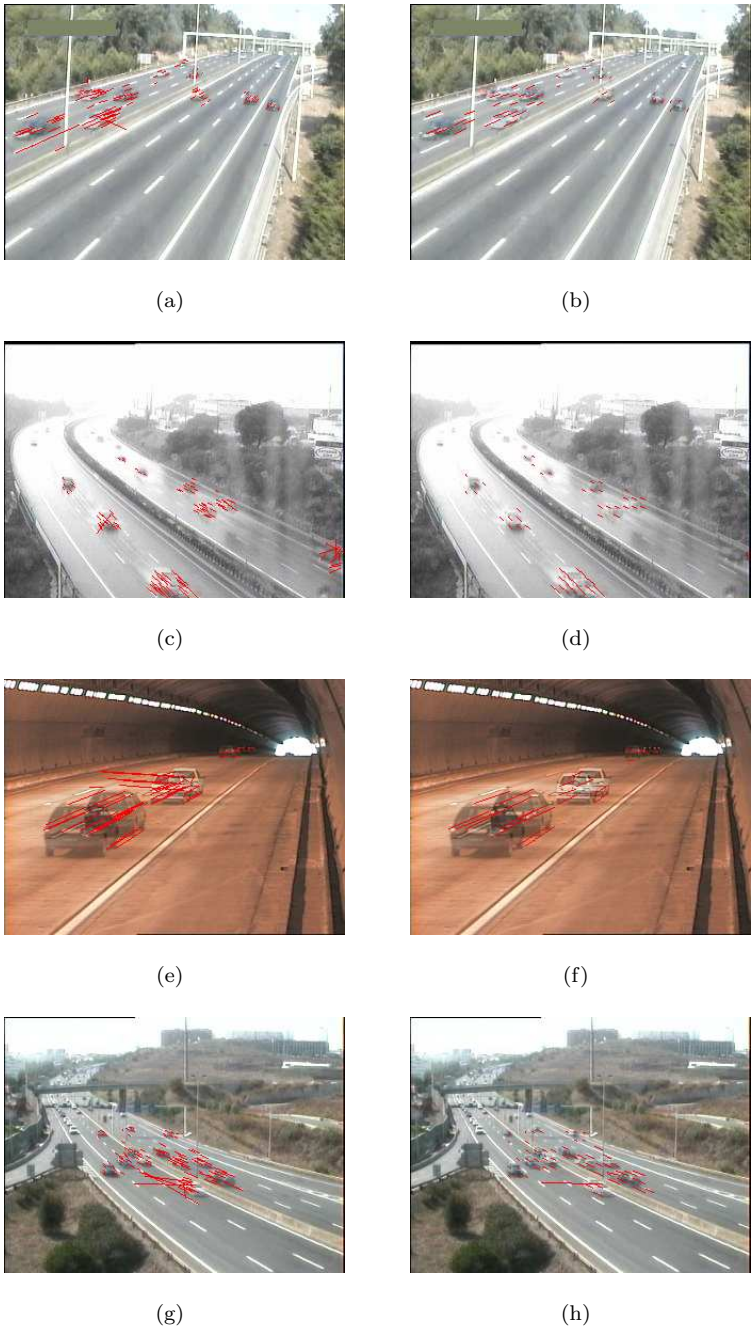


Figure 3.1: Results of the Block Median Filtering to reduce the disturbance in the optical flow estimation. a), c) e), and g) output of the estimated optical flow. b), d), f), and h) result of the block median filtering. Note that in h) not all the wrongly detected vectors are filtered.

3.4 Traffic Flow Direction Learning

During the traffic flow direction learning process it is assumed that vehicles circulating on the road are moving in the correct direction along the lane. At the end of this learning process, a model of the flow direction is obtained for each block of $n \times n$ pixels in the image.

The estimation of each lane's flow direction on the image is learned through the analysis of a large amount of frames. It is assumed that the motion directions of the vehicles (θ) follows a Gaussian distribution. This way, the direction can be modeled by a *Mixture of Gaussians* (MoG). The idea behind this modeling process was described in detail in Sec. 2.1.1. Each block of the image will be modeled by a MoG through the analysis of the vehicle's movement (see Fig. 3.2).

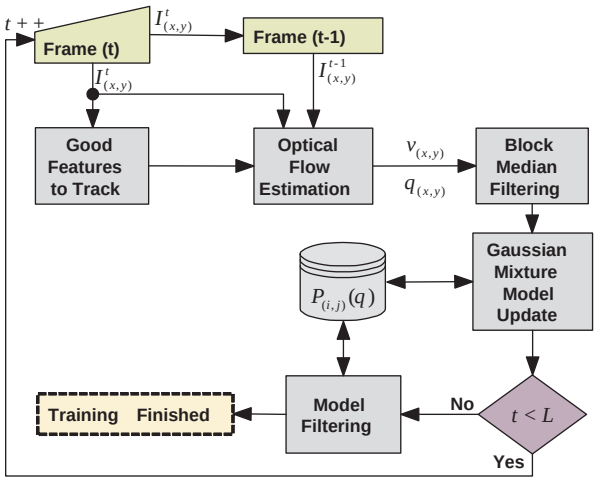
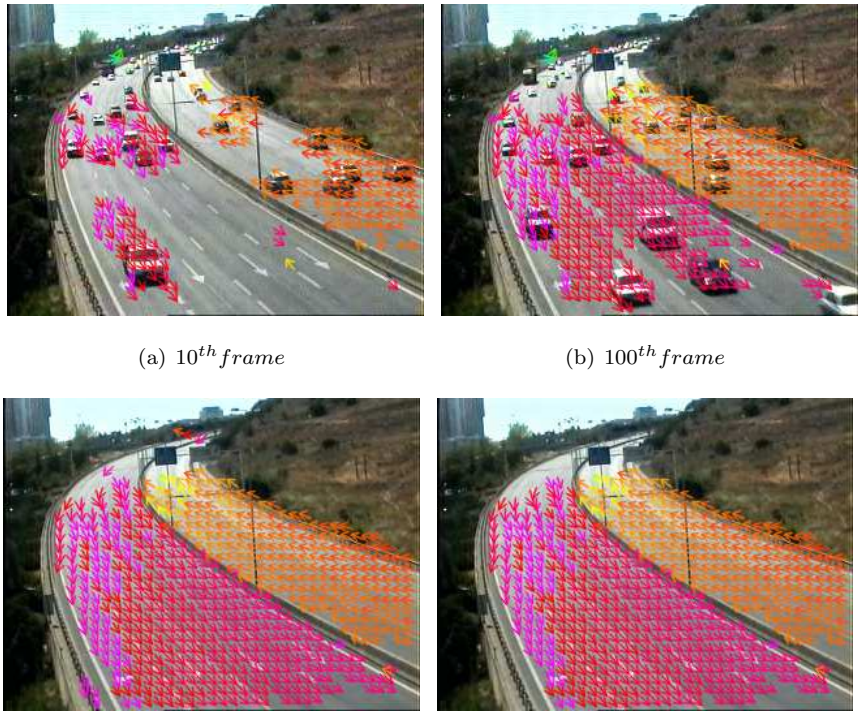


Figure 3.2: Flowchart of the traffic flow direction learning process.

For this purpose, the maximum number of kernels in the MoG was limited to $K = 3$. At the end of the learning process, the models are filtered in order to eliminate some noisy flow vectors. The Gaussian models with low relevance are discarded, i.e., the models with a weight, ω_i, below the predefined threshold are discarded. This way, this learning process is also used to robustly discard noisy estimated flow vectors. The result of this filtering process is shown in Fig. 3.3(d) and 3.4(d).

Another advantage of the Gaussian mixture modeling in this situation is that



(a) $10^{th}\,frame$ (b) $100^{th}\,frame$

(c) $300^{th}\,frame$ (d) Filtered result

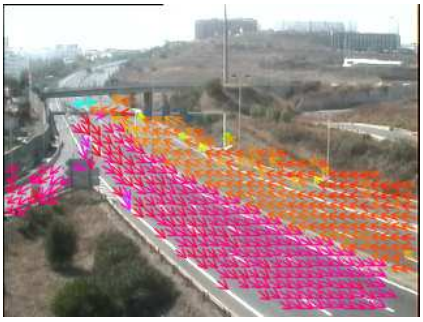
Figure 3.3: Traffic flow direction learning process: a), b) and c) show the evolution of the orientation pattern modeled by the first Gaussian of the MoG on a highway scenario. d) the result of the learning process after the model weight filtering.

it can embrace a variety of directions for the same block, which is very useful in lanes with exits or bifurcations, modeling also the movements of vehicles that are changing between lanes.

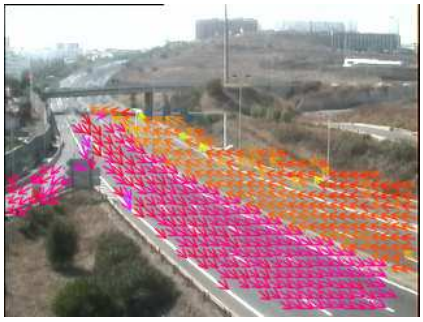
The number of necessary frames to obtain a correct estimation of the MoG depends on the number of vehicles circulating on the road. In our experiments a stack of 1000 learning frames proved to be the necessary to obtain a good traffic directions model. Obviously, if there are no vehicles circulating on one block, the direction of that block will not be learned. Two examples of the traffic flow directions learning process are shown in Figs. 3.3 and 3.4.

(a) $10^{th} frame$

(b) $100^{th} frame$



(c) $300^{th} frame$



(d) Filtered result

Figure 3.4: Traffic flow direction learning process: a), b) and c) show the evolution of the orientation pattern modeled by the first Gaussian of the MoG on a highway scenario. d) the result of the learning process after the model weight filtering.

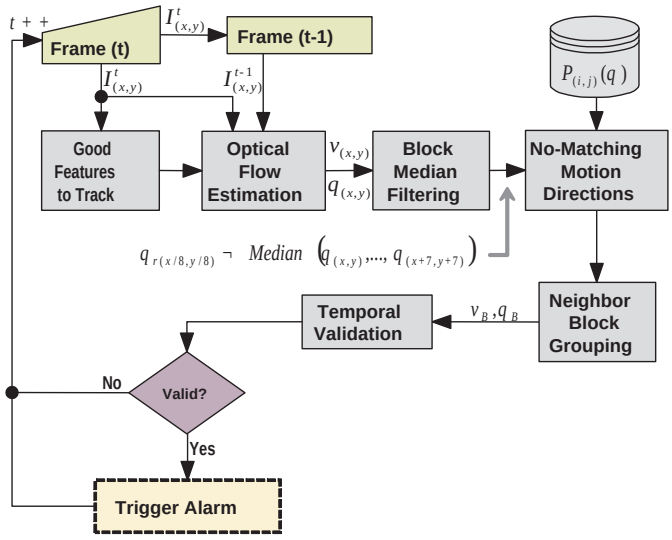


Figure 3.5: Flowchart of the proposed wrong way drivers detection system.

3.5 Wrong Way Vehicle Detection

In this section it is described the methodology used to detect and validate the vehicles circulating on the wrong way of the road (see Fig. 3.5). For each new frame the optical flow and the median of the flow direction are computed for each block. A block is classified as circulating on the wrong direction when the difference, Δ_d, between both the direction of the flow, in the present frame, and the estimated means of the corresponding block learned are larger than 2.57σ for the 99% confidence interval. The difference between two given angles, α_1 and α_2, is calculated by Eq. 3.1, where ρ is given by Eq. 3.2. After this detection, the neighbor blocks with flow estimated in a wrong direction are grouped into blobs just like in the segmentation process (see Sec. 2.2.4). Each blob that results from this grouping process is possible wrong way vehicle.

$$\Delta_d = \begin{cases} \rho & \rho < \pi \\ 2\pi - \rho & \rho \geq \pi \end{cases} \tag{3.1}$$

$$\rho = |\alpha_1 - \alpha_2| \tag{3.2}$$

Some situations and sources of noise were identified as contributing to a significant decrease in the system's performance (see Fig. 3.6). Some of these

situations are:

- moving cast shadows
- illumination changes
- low and variable frame rate
- vibration of the surveillance camera pole
- wrong flow estimation of the LK algorithm due to image noise

This way, it is necessary to validate every object detected as circulating in the wrong way, to eliminate false positives, before triggering an alarm. In order to discard some false positives, a temporal validation is used to verify if the detected objects make a coherent trajectory.

3.5.1 Temporal Validation

The first stage of the temporal validation consists of tracking all the objects detected as circulating on the wrong side of the road and verifying if they appear in consecutive frames. If an object is detected more than n times in m consecutive frames, it will be considered as an object circulating on the wrong side of the road, namely $n = 4$ and $m = 6$. A second order Kalman filter is used to track and predict the position of the vehicles in consecutive frames.

When a detected flow does not match the learned motion direction model, a new *tracker* is initiated. The object image position, P, is given by the center of mass of all neighbor blocks detected as being part of an object moving in a wrong direction. The velocity, ϑ, of the object is obtained in 2 parts: the direction is obtained as the median of the direction of all grouped blocks, and the module is computed by the average motion of all grouped blocks. When tracking the object, it is only necessary to save P, ϑ, and the area of the grouped blocks.

After this validation, a verification is performed in order to see if the detected object makes a coherent trajectory. This is obtained by the analysis of the positions and the estimated displacement of the object in the last m frames. In some scenarios, it is possible that objects, like poles and trees, have a noisy flow

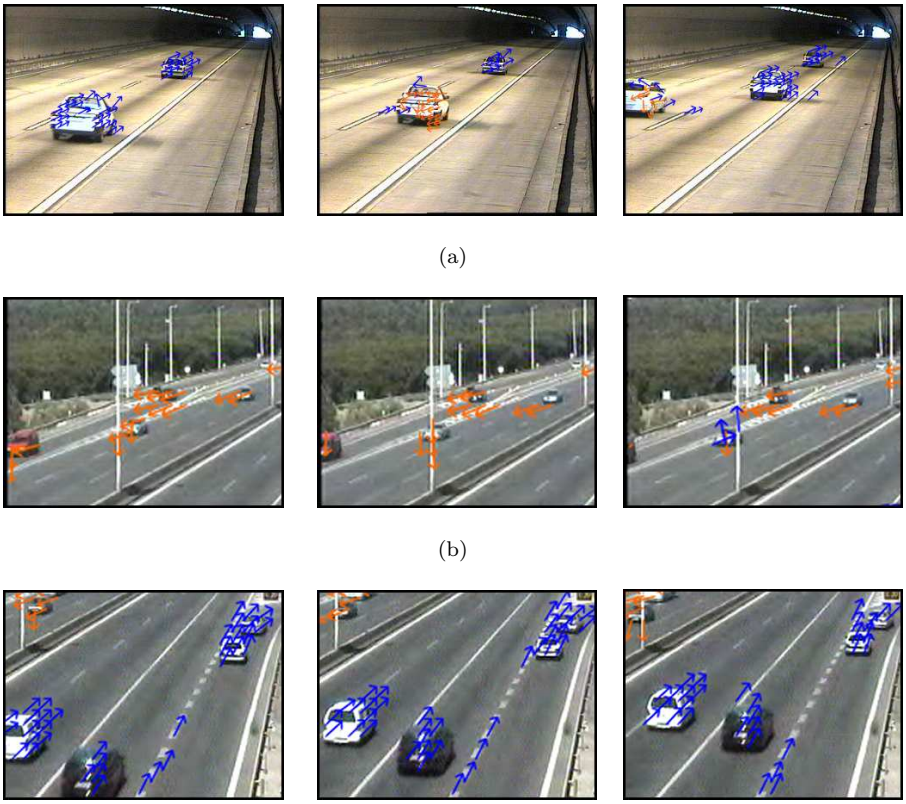


Figure 3.6: Some examples of false optical flow estimation. a) Wrong optical flow estimation in a tunnel scenario; b) Pole interference in the optical flow estimation; c) False optical flow estimation due to cast shadows.

due to cars driving behind these. Therefore, this kind of flow could generate false positives. The variance (ν) of the direction of the flow vectors is also analyzed so that these false positives can be eliminated. If ν is above a preset threshold, this possible object will be discarded.

3.6 Experimental Results

3.6.1 Tests Performed On-Site

The proposed wrong way vehicle detection system was tested in several sites in Portuguese highways for several days. The tests were performed under different weather conditions, illumination, image quality and fields of view. The

description of the test sites is presented in Fig. 3.7. Some examples of successful detections are presented in Figs. 3.13, 3.11, 3.12, 3.10, 3.9. The results of the performed tests are presented in Table 3.1. In most of the detection events, vehicles are reversing, but, for the system, this situation can also be considered as a wrong way vehicle event.

The three main reasons for false positives detection are: a) Vehicle's illumination projected onto the opposite direction road at night (see Fig. 3.14(c)); b) The shadow casted by a vehicle is projected onto the opposite direction road, this situation might occur at sunrise or at sunset and with taller vehicles (see Fig. 3.14(a) and 3.14(a)); c) The higher part of taller vehicles is projected onto the opposite direction road (see Fig. 3.14(d)).

In the tests performed in real highway sites it is not possible to evaluate the detection rate of the system because no ground truth is available. Some other tests will be described below in order to evaluate the detection rate of the system.

(a) Site 1 (b) Site 2 (c) Site 3

(d) Site 4 (e) Site 5 (f) Site 6

Figure 3.7: Sites in Portuguese highways used to validate the proposed wrong way vehicle detection system.

Site	Duration (days)	Detected	False Positives/Day
site 1	68	7	0.0
site 2	23	3	0.0
site 3	77	9	0.0
site 4	193	5	0.13
site 5	220	1	0.33
site 6	36	3	0.11

Table 3.1: Performance of the proposed wrong way vehicle detection system in some surveillance cameras in Portuguese highways.

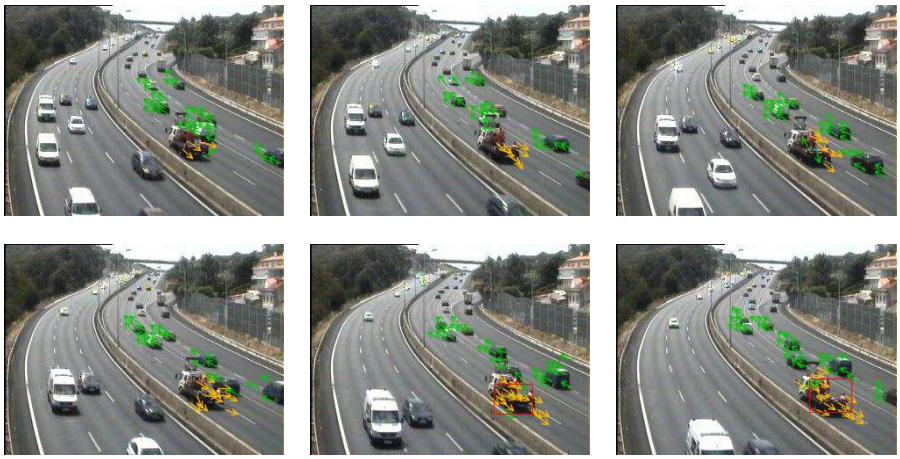


Figure 3.8: Wrong way vehicle detection in a typical outdoor scenario. The vehicle is reversing in a lane disabled for maintenance.

3.6.2 Tests Performed in Datasets of Highway Scenarios

Six video sequences from Portuguese highways scenarios were used to evaluate the detection rate of the system. These challenging datasets were obtained under some different weather conditions. The test videos are described in Table 3.2. In Video 1, 2 and 3, one of the roads is disabled and the traffic of the two directions occurs in the same road. The road direction learning process was performed before these traffic changes. Therefore, one of the traffic directions is considered as circulating on the wrong direction. For Video 4, 5 and 6, all the directions learned during the training phase were increased by π and,

Figure 3.9: Wrong way vehicle detection in a typical outdoor scenario. The vehicle is reversing in the hard shoulder.

therefore, all vehicles on the road should be considered as circulating on the wrong direction.

The results of the tests performed are presented in Table 3.3. An example of wrong way vehicle detection under occlusion is presented in Fig. 3.15. In most of the situations, the miss detection occurs because no good features to track are selected for the vehicle. The absence of good features to track is due to low contrast situations and to image blur. The image blur can also lead to wrong flow estimations. An example of miss detection due to wrong flow estimation is presented in Fig. 3.16. In most of the situations, wrong flow estimation is due to low image quality.

The system is able to detect vehicles moving on the wrong direction of a lane over a 320×240 pixel image at 33 fps on a 3.2 GHz *P4 Intel Processor* under Linux OS.

3.7 Conclusions

In this chapter, a methodology to detect vehicles circulating on the wrong side of the highway using optical flow information is proposed. In the learning phase, the motion direction of each lane is modeled by a Mixture of Gaussians. The optical flow is computed every frame in order to estimate the vehicle's motion

Figure 3.10: Wrong way vehicle detection in a tunnel scenario. The vehicle is reversing in the hard shoulder.

direction. If the estimation direction does not match the direction model, then a temporal validation is used. If the validation succeeds, an alarm is triggered.

The experiments conducted on a large number of scenes demonstrate that the proposed system is able to detect vehicles circulating on the wrong side of the road with good accuracy, and that it is robust to weather conditions, illumination and image quality variation.

Figure 3.11: Wrong way vehicle detection at night. The vehicle is reversing in a lane disabled for maintenance.

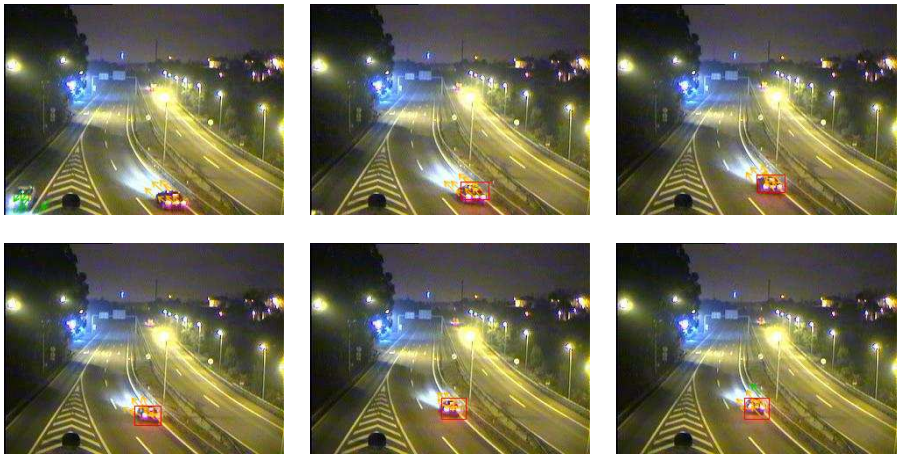


Figure 3.12: Real wrong way vehicle detection at night.

Figure 3.13: Real wrong way vehicle detection.

(a) Shadow casted by a vehicle is projected onto the opposite direction road.

(b) Shadow casted by a vehicle is projected onto the opposite direction road.

(c) Vehicle's illumination projected onto the opposite direction road.

(d) Higher part of the bus is projected onto the opposite direction road.

Figure 3.14: Some situations of false wrong way vehicles detected in the performed tests.

Video	Preview	Duration	Description
Video 1		00:10:12	The road on the right is disabled. The traffic of the two directions occurs in the same road.
Video 2		00:07:48	The road on the left is disabled. The traffic of the two directions occurs in the same road. Low contrast.
Video 3		00:16:40	The road on the left is disabled. The traffic of the two directions occurs in the same road. Low image quality.
Video 4		00:09:24	Wet road in a tunnel scenario.
Video 5		00:07:32	Fog Situation.
Video 6		00:08:17	Rain drops in the camera's lens.

Table 3.2: Datasets used to validate the proposed wrong way vehicle detection system.

Video	Ground Truth	Detection Rate	False Positives
video 1	108	1.00	0
video 2	72	0.97	0
video 3	336	0.94	0
video 4	133	0.98	0
video 5	331	0.98	0
video 6	123	0.99	0

Table 3.3: Performance of the proposed wrong way vehicle detection system in test datasets in Portuguese highways.

Figure 3.15: Wrong way vehicle detection under occlusion in a simulated event.

Figure 3.16: Miss detection of wrong way vehicle due to image noise.

Chapter 4

Stopped Vehicle Detection

4.1 Introduction

A vehicle stopped on the road or on the hard shoulder poses a serious threat to the other road user and to the road safety. This threat can become a really dangerous situation, mainly in a tunnel scenario. A fast detection of a stopped vehicle can help prevent serious accidents by warning a human operator at the traffic management center. Several measures can be taken to minimize the threat, such as warning the oncoming vehicles (e.g.: via traffic telematic systems or radio announcements) and warning the responsible authorities. In several countries, and in order to maintain road safety, it is forbidden to pull over in a highway. Usually, a driver pulls over in case of emergency/malfunctioning/etc. Some actions can be taken to help the driver, such as sending a roadside assistance vehicle.

The stopped vehicles detection system presented in this chapter has three main phases. Firstly, the vehicles in the scene are segmented using the segmentation process presented in Sec. 2.2. Secondly, it is verified if there are any static pixels segmented during a certain period of time. Those static pixels are then grouped into blobs. Finally, a temporal validation is applied to the stopped blobs in order to discard false positives. If the validation succeeds, an alarm is triggered in the traffic telematic system.

Several tests were performed on this system with some public datasets. This

system was also tested on real highway scenarios during several months. The results of the tests proved that the presented system robustly detects stopped vehicles against occlusions and in high traffic density scenarios.

4.1.1 Related Work

A significant amount of different methodologies have been proposed to robustly detect static objects in a scene. This problem can be divided into two main types: a) abandoned item detection; and b) stopped vehicle detection.

Over the last decade, a lot of attention has been given to the detection of abandoned objects based on video surveillance, because of the constant terrorist threat. Some methodologies have been proposed to detect abandoned items [6, 23, 48, 69, 31]. Most of these methods assumes that the scene is not crowded and occlusions are minimal. The system detection performance strongly depends on the performance of the tracking process.

In [63] a general purpose methodology to detect static objects in the scene was presented. A long-term (B_{lt}) and a short-term (B_{st}) background model are used to identify temporarily static regions in the image. Two binary foreground (F_{lt} and F_{st}) maps are estimated by comparing the current frame with the background models. A pixel is classified as stopped if it is classified as background by F_{lt} and classified as foreground by F_{st}. This methodology is based on a pixel-based analysis and does not require a tracking process.

A mixed tracking approach is used in [56] to identify vehicles that stop in a forbidden area of the scene. A predictive Kalman filter-based process is used to track vehicles currently in the scene. A background suppression process is used to identify vehicles in the scene. In order to robustly perform the data association of the vehicles between frames, appearance models and probability masks are used to extract relevant information from each vehicle. The authors claim that this tracking system robustly deals with long-lasting occlusions. An object is classified as *stopped* if it is stopped but it was moving in the last frames, otherwise it is classified as false positive and is discarded.

A tracking approach is also used in [10] to obtain sets of objects trajectory and subsequently detect the stopped vehicles. Firstly, a robust motion detector is used to identify the vehicles in the scene. Then, the tracking process is used

to analyze the blob's activity. An optical flow algorithm is employed to follow feature points inside the detected blobs frame by frame, using local and global information. This information is used to manage the blobs tracking and to identify occlusions. An effective SOM neural network is used to detect and remove outliers during the object occlusion. By analyzing the centroids of the tracked objects, a set of trajectories is obtained. After a moving average filtering process those trajectories are used to detect vehicles that stop. Lastly, a vehicle is classified as stopped if a subset of its trajectory is restricted inside a boundary area.

4.2 Proposed Methodology

In this section, it is presented a novel approach to detect stopped vehicles in highways in cluttered conditions. The main assumption for this method is that if a given pixel is classified as foreground and it maintains the same color during a certain period of time, then it probably belongs to a stopped vehicle.

The proposed methodology does not require object tracking. Therefore, it is not restricted to predefined event heuristics that require detection, tracking and identification of every single object in the scene. The performance of this method is not upper-bounded to error prone detection/tracking and correspondence tasks that usually fail in crowded scenes. The trajectory analysis based methodologies also fail in situations where the displacement of the vehicle in the image is too small due to the perspective effect.

The proposed stopped vehicles detection system has three main phases. Firstly, every vehicle currently in the scene is identified and segmented using a robust segmentation process (presented in Sec. 2.2). Secondly, the color of every segmented pixels is analyzed to identify the segmented pixels that are static in the image. In order to identify the static pixels, a *pixel history cache* is used. This process is inspired in a methodology introduced by Kim et. al. [46] to perform the segmentation of moving objects in the scene (see Sec. 2.1.1). The identified static pixels are then grouped into blobs and labeled. Finally, and after identifying the blobs, a spatiotemporal validation is applied to those blobs to discard some false positives. This validation process is also used to establish

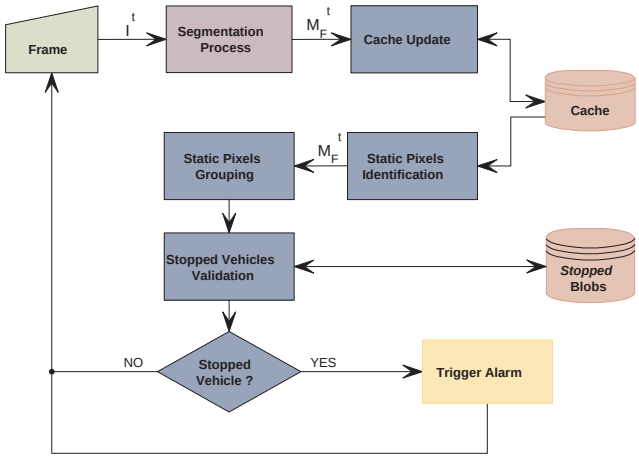


Figure 4.1: Flowchart of the proposed stopped vehicles detection process.

the minimum period of time that the vehicle should be stopped to be classified as stopped vehicle. It is not desired to detect vehicles in a traffic jam situation, where vehicles stop for a short period of time. A blob is classified as stopped vehicle if it maintains the same size and position during a certain period of time. If the validation succeeds, an alarm is triggered in the traffic telematic system. A flowchart of the presented stopped vehicle detection system is shown in Fig. 4.1.

4.3 Static Pixels Identification

Static pixels are identified by the analysis of the segmented pixels color history. A pixel is classified as static if it is labeled as foreground with the same color during a certain period of time. A data structure called *pixel history cache* is used to maintain the pixel color history. For each pixel it is maintained the set of colors that were assigned to the pixel in the last T_h frames and its frequency information. The pixel history cache is an array with the image size dimension. Each array cell corresponds to a pixel history, $I_{x,y}$, called Codebook that has a list of Codewords. Each Codeword represents a color that was assigned to the pixel (x, y), it is composed by the RGB color components and a validation buffer. The validation buffer stores the occurrence history of the color in the last α frames. This validation buffer of a given pixel x, y is updated with '1' if

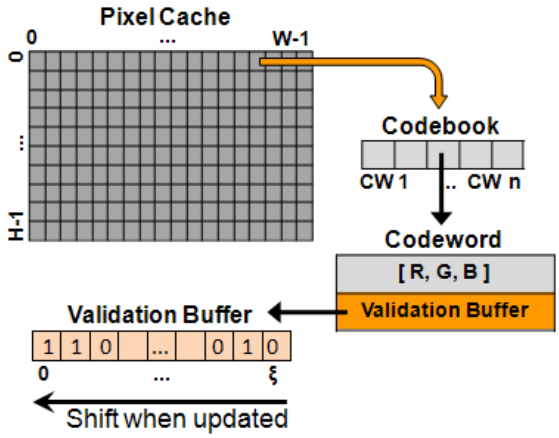


Figure 4.2: Flowchart of the pixel history cache data structure used to store the pixels color history and subsequently used to identify the static pixels.

the color matches the color of $I^t_{x,y}$, otherwise it is updated with '0'. If the pixel (x, y) has not been classified as foreground in the current frame the validation buffer is also updated with '0'. The validation buffer is shifted before the update procedure and the update is performed in the first cell of the buffer. Occlusions, vibrations or lighting changes that can temporarily hide or change a pixel color of the vehicle are taken into account with this validation structure. A flowchart of this pixel history cache data structure is depicted in Fig. 4.2.

4.3.1 Cache Update

For each foreground pixel (x, y) it is verified if the current color of $I^t_{x,y}$ matches the color of any *codeword* present in the *codebook* (x, y). The metric used to identify a match between two colors is the same presented in Sec. 2.1.1. The color matching area is represented by a cylinder in RGB color space (Fig. 2.2). A match occurs when the sample color and brightness distortion are within the parameters. ϵ, α and β were defined empirically using several datasets of outdoor scenarios. If a match between $I^t_{x,y}$ and a *codeword* occurs, the RGB components are updated with a weighted average of $I^t_{x,y}$, and the validation buffer is updated with '1'. For all the other *codewords* present in the *codebook* (x, y) the buffer is updated with '0'. If there is no *codeword* matching, a new *codeword* is created in the pixel (x, y) *codebook*. Every *codeword* of the pixels

classified as background in the current frame are also updated with '0'. Finally, every Cache *codeword* in which the color was not assigned to the pixel $I_{x,y}$ in the last T_h frames is deleted from the *codebook* (see Algorithm 3).

Algorithm 3 Pixel color history cache update.

1: **for all** pixel ϕ **do**
2: Match = 0
3: **for all** Codeword CW_i of pixel ϕ **do**
4: **if** $colordist(CW_i, I(\phi)) < \epsilon$ **then**
5: Update the Validation Buffer of CW_i with '1'
6: Match = 1
7: **else**
8: Update the Validation Buffer of CW_i with '0'
9: **end if**
10: **if** $last\_frame\_was\_assigned(CW_i) > T_h$ **then**
11: Delete the Codeword CW_i from the pixel history
12: **end if**
13: **end for**
14: **if** Match = 0 **then**
15: Create new Codeword with $I(\phi)$ color
16: **end if**
17: **end for**

4.3.2 Static Pixels Detection

A pixel (x, y) is classified as static if there is a *codeword* in the *codebook* (x, y) with β occurrences over the last α frames. This strategy handles the occlusion problem mentioned above. However, when a stopped vehicle turns on the hazard warning lights, two or more colors are intermittently assigned to the pixels in the light's region. To deal with this particular, but frequent situation, a pixel is also validated as static whenever two *codewords* have more then $\beta/2$ occurrences over α frames. An example of stopped vehicle detection in this conditions is depicted in Fig. 4.3(b) (see Algorithm 4).

The system's parameters were obtained empirically during several experi-

Algorithm 4 Static pixel identification.

```
1: for all pixel φ do
2:   number_of_half_β = 0
3:   for all Codeword CW_i of pixel φ do
4:     hits = count_hits_on_validation_buffer(CW_i)
5:     if hits > β then
6:       Assign pixel φ as static
7:     end if
       { Deal with the hazard warning lights }
8:     if hits > β/2 then
9:       number_of_half_β++
10:    end if
11:    if number_of_half_β > 1 then
12:      Assign pixel φ as static
13:    end if
14:  end for
15: end for
```

ments conducted in outdoor scenarios in order to maximize the system's performance ($T_h = 25$, $\beta = 40$, $\alpha = 64$, and $\epsilon = 30$).

4.4 Stopped Vehicle Validation

After the static pixel detection procedure, a labeling procedure is used to identify the *stopped blobs*. Not all the stopped blobs are in fact stopped vehicles. These can be vehicles moving slowly in the image due to perspective effect or to a traffic jam situation. A spatiotemporal validation is used to discard this kind of false positives. A vehicle is classified as stopped if it maintains the same position and size during a certain period of time (t_v). If the validation succeeds, an alarm is triggered.

This validation period, t_v, is also used to establish the minimum period of time during which the vehicle should be stopped to be classified as stopped vehicle, it is not desired to detect vehicles in a traffic jam situation, where vehicles stop for a short period of time. In order to deal with perspective effect

near the image vanishing point, the validation period is inversely proportional to the average velocity on the region where the stopped blob is. This average velocity is obtained by the average optical flow module estimation during a certain period of time in a learning phase just like in Sec. 3.4).

4.5 Experimental Results

The stopped vehicle detection system proposed in this chapter was tested in several outdoor conditions in order to evaluate the system performance. Two different types of tests were performed: a) tests with test sets available on the Web – these tests are very interesting to evaluate comparative results with other developed algorithms to detect stopped vehicles (Sec. 4.5.1); and b) tests performed on-site in some outdoor scenarios during several hours/days – these tests allow us to analyze the performance of the system with different weather conditions, illuminations changes, image quality and fields of view (Sec. 4.5.2).

Some examples of stopped vehicle detection with success are shown in Fig. 4.3. The green bounding box in the left column images represents the vehicle detection, and the red bounding box represents the detection of a stopped vehicle. A chromatic representation of the frequency of the most frequent codeword for each pixel is shown in the middle column images, where the blue color means that the most frequent codeword has matched only once and the red color means that the frequency is above the threshold and that the pixel is considered as static. On the right column images it is represented the color of the most frequent codeword. In the area of a stopped vehicle, the color of the most frequent codeword should be the same as the pixel color of the vehicle. The green pixels represent the absence of codewords for those pixels. A sequence of images showing the detection of a stopped vehicle under occlusion of other vehicles circulating on the road is shown in Fig. 4.4.

4.5.1 Tests Performed in Public Datasets

The proposed stopped vehicle detection system was tested with some public datasets used in the literature (see Table 4.1). Four different datasets were analyzed: a) i-Lids dataset for IEEE International Conference On Advanced Video

and Signal Based Surveillance – AVSS 2007[1]; b) Imagery library for intelligent detection systems (i-LIDS)[2]; c) OpenVISOR – VIdeo Surveillance Online Repository[3]; and d) International Workshop on Video Surveillance & Sensor Networks (VSSN 2006) – Algorithm competition in foreground/background segmentation[4]. The performance of the proposed stopped vehicle detection system tested with these datasets is presented in Table 4.2. The system detected all the stopped vehicles in the datasets without any false positive detected.

4.5.2 Tests Performed On-Site

The proposed stopped vehicle detection system was tested in some outdoor sites during 24 hours. The tests were performed under different weather conditions, illumination and fields of view. The test sites are described in Table 4.3.

Table 4.4 shows the experimental results obtained in the test scenarios. During the performed experiments it was verified that the mis-detection of stopped vehicles was due to the lack of image contrast. Most of the false positives are caused by the segmentation process used as input in this system. The segmentation process does not handle all the sudden illumination changes and some of these situations result in a false positive detection. Sudden illumination changes are mainly caused by artificial lighting variation. Another cause of false positive detection – not so frequent – is caused by segmented objects with the same color and appears frequently (see Fig. 4.5).

The proposed system is able to detect stopped vehicles in highways over a 320×240 pixel image at 16 fps on a 3.2 GHz *P4 Intel Processor* under Linux OS (foreground segmentation process and stopped vehicle detection system).

[1] i-Lids dataset for AVSS 2007 web site: http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_d.html

[2] i-LIDS web site: http://scienceandresearch.homeoffice.gov.uk/hosdb/cctv-imaging-technology/video-based-detection-systems/i-lids/ilids-datasets-pricing/ parked-vehicle-detection?view=Standard

[3] OpenVISOR web site: http://www.openvisor.org/video_videosInCategory.asp?idcategory=12

[4] VSSN 2006 web page: http://mmc36.informatik.uni-augsburg.de/VSSN06_OSAC/

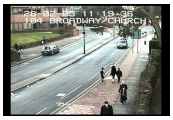
Video	Preview	Duration
V1 – AVSS – PV_Medium		00:02:29
V2 – AVSS – PV_Hard		00:02:54
V3 – ILIDS – PVTR_1		00:07:16
V4 – ILIDS – PVTR_2		00:02:09
V5 – ILIDS – PVTR_3		00:04:16
V6 – OpenVisor – StoppedVehicle_Video_0		00:01:50
V7 – OpenVisor – StoppedVehicle_Video_1		00:02:13
V8 – OpenVisor – StoppedVehicle_Video_2		00:02:00
V9 – OpenVisor – StoppedVehicle_Video_3		00:02:03
V10 – VSSN – TunnelVideo		00:08:00

Table 4.1: Public datasets used to validate the stopped vehicle detection system.

Video	Ground Truth	Detection Rate	False Positives
V1	1	1	0
V2	1	1	0
V3	2	2	0
V4	2	2	0
V5	2	2	0
V6	2	2	0
V8	1	1	0
V9	1	1	0
V10	3	3	0

Table 4.2: Performance of the proposed stopped vehicle detection system in some public datasets.

4.6 Conclusions

In this chapter, a methodology to detect stopped vehicles in highways based on a pixel color history is proposed. This system uses as input the result of a foreground segmentation process presented in Sec. 2.2. A pixel color history cache is constructed with the foreground pixels and, afterwards, the history cache is analyzed in order to identify static pixels. The static pixels are grouped into blobs and validated. If the temporal validation succeeds an alarm is triggered.

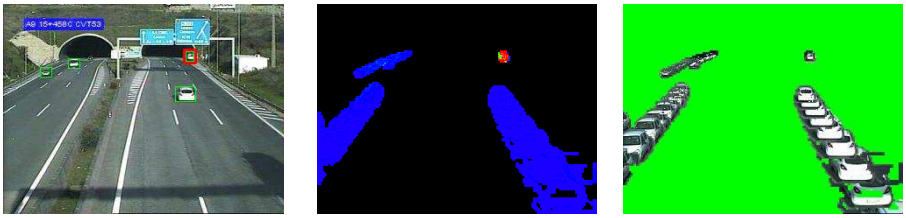
The proposed system was validated on-site in several different scenarios of Portuguese highways during several days. The system was also tested with several datasets available on the Web. The experiments demonstrate that the proposed system is able to detect stopped vehicles in outdoor scenarios with a good accuracy, and that it is robust to different weather conditions, illumination changes and image quality variation. The system achieved an excellent detection rate and a low false positive rate. The false positives are mainly caused by sudden illumination changes not handled by the segmentation process.

Site	Preview	Duration	Description
S1		1 day	Tunnel entrance in a Portuguese highway.
S2		1 day	Part of a parking lot at the Coimbra University.
S3		1 day	Toll booth in a Portuguese highway. Several vehicles stopped to pay the toll.
S4		50 days	Tunnel scenario in a Portuguese highway. No ground truth available for this site.

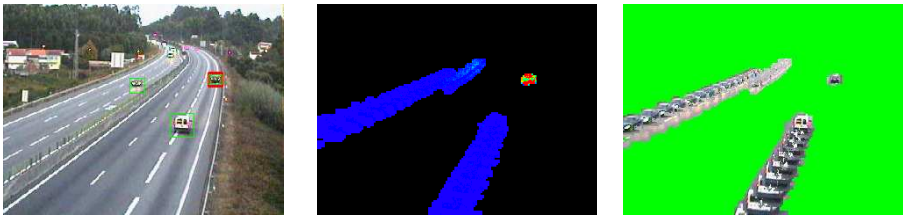
Table 4.3: Scenarios used to perform the on-site tests to the proposed stopped vehicle detection system.

Scenario	Ground Truth	Detected	False Positives/day
S1	3	3	1
S2	79	78	3
S3	1009	1009	2
S4	-	62	0.38

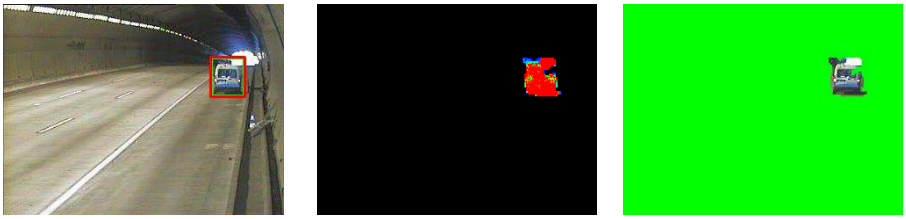
Table 4.4: Performance of the proposed stopped vehicle detection system tested on-site in some outdoor scenarios.



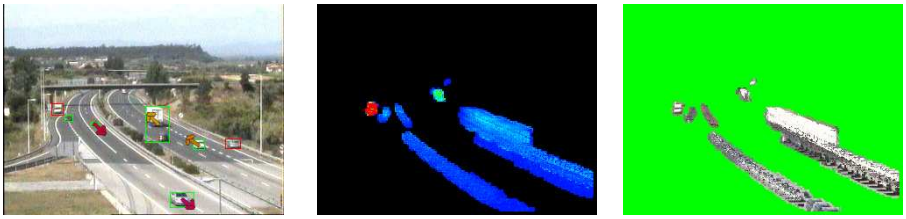
(a)



(b)



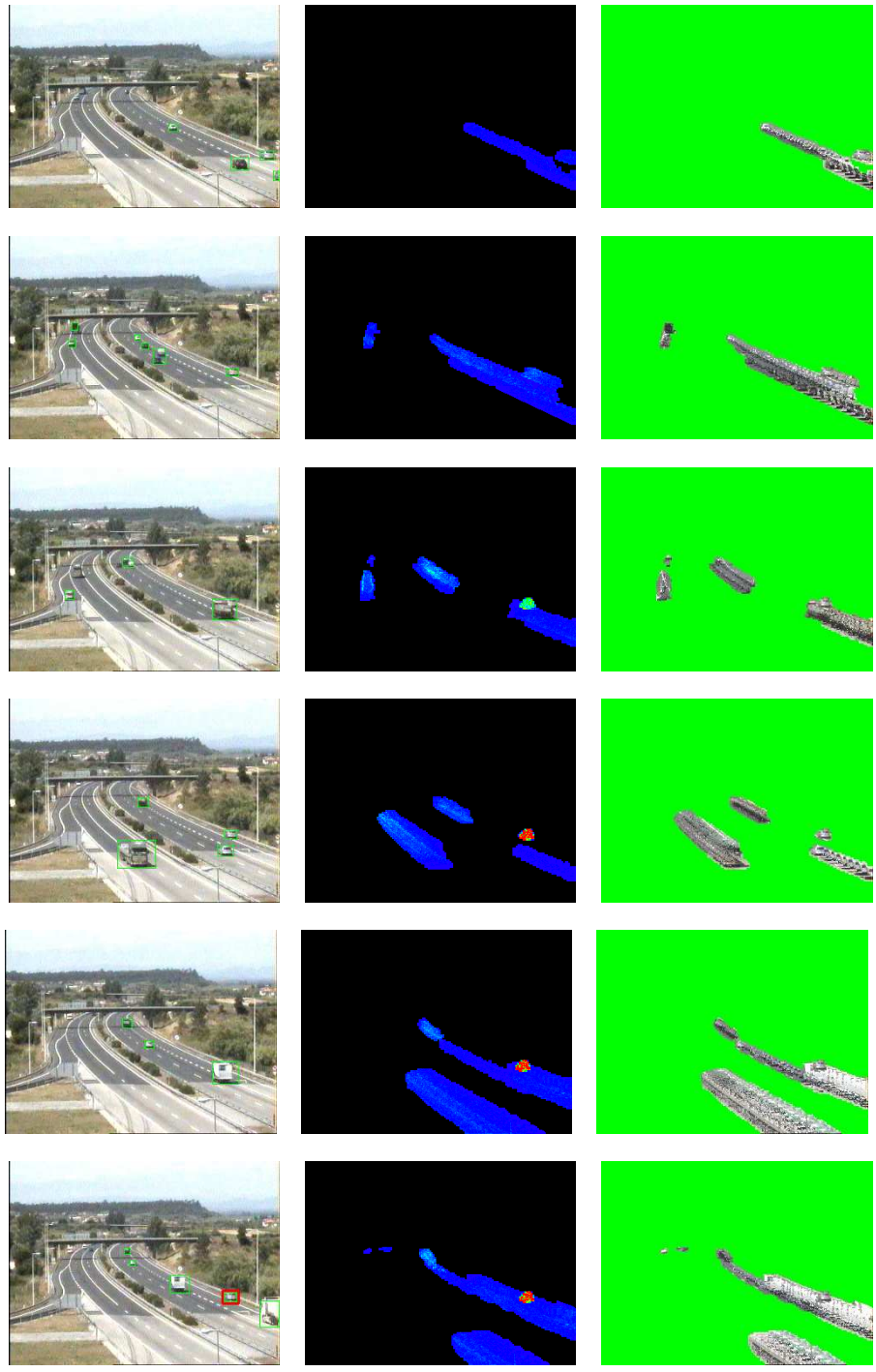
(c)



(d)

(e)

Figure 4.3: Some examples of stopped vehicle detection in some Portuguese highways scenarios. a) tunnel entrance; b) stopped vehicle with the hazard warning lights turned on; c) tunnel scenario; d) stopped vehicle on the left side road (the one on the right was validated before); and e) bridge scenario.



(a) Frame I^t with the detected vehicles.

(b) Number of matches with the most frequent Codeword.

(c) Color of the most frequent Codeword.

Figure 4.4: Example of stopped vehicle detection with occlusions.

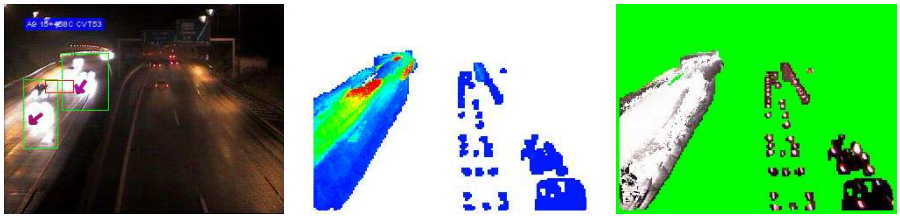


Figure 4.5: False stopped vehicle detection. The areas where the vehicle's lighting is projected are segmented as foreground. For each pixel of those areas, the most frequent color is white and those pixels are classified as static.

Part III

Automatic Traffic Surveillance System

Chapter 5

AVISAR Project

5.1 Introduction

In the context of the road telematics project, Brisa[1] provided their highways with a large road surveillance infrastructure, with the aim of monitoring traffic and, in case of a possible dangerous situation, of analyzing and remotely accompanying the situation and its impact.

For the optimization of the project's efficiency, the network was subject to a thorough analysis. This analysis aimed at finding the right places to install the cameras, based on criteria like traffic intensity, vehicle's accidents or other events, access ways and other strategic places of coverage. The monitoring operation center, CCO [Centro de Coordenação Operacional], was provided with state-of-the-art equipment and systems for the control and operation of this surveillance infrastructure, integrated in the Brisa's Road Telematic System.

It is in this context, of more than 500 cameras and an available team of 8 elements for monitoring and operating, that the AVISAR Project arises, for the automatic detection of incidents and to monitor the traffic on the highways. This ATS system is divided in two main parts: a) an Automatic Incident Detection (AID) system [9, 61, 60, 58, 59, 57] and b) an Automatic Traffic Monitoring (ATM) system [51, 53, 52]. On this chapter it will be described the AID system developed to detect incidents in the highways managed by Brisa. The main

[1] Brisa – Auto-estradas de Portugal S.A. (www.brisa.pt)

modules of this AID system were developed and validated during the research work presented on this thesis and were described previously.

The objective of this project is to develop automatic mechanisms for the analysis and learning of traffic's normal behavior patterns, anywhere on the road network, and from that knowledge obtaining an automatic identification of anomalous situations that may interfere with the road's normal circulation conditions and safety, according to the above established for the manual analysis. In the case of an anomalous situation, the AID system shall dynamically integrate the systems and services that support operations for the possible activation of response plans. The Brisa side infrastructure responsible for the telematic management is the iBrisa management system.

Summing up, with the AVISAR Project, one intends to provide the video surveillance cameras integrated in the Brisa's Road Telematic System with the necessary mechanisms to aid the video surveillance task, aiming at warning about potentially anomalous and dangerous events to road safety.

This project was designed and developed together with Institute of Systems and Robotics[2] and with the collaboration of MakeWise[3] in the system integration.

5.2 System Description

The automatic incident detection system of this automatic traffic surveillance system prototype was developed using the algorithms presented on this thesis. The flow chart of this system is depicted in Fig. 5.1. In order to attain an effective management of the system it was necessary to develop an efficient user interface and communication layer as well.

During the initialization of the system some parameters can be set. Some of these parameters are: the learning period, the video source, the video capture channel and the scene masks. The scene masks allow the system to achieve an overall better performance, increasing the frame rate, decreasing the false positive rate of the AID system and not analyzing non wanted secondary roads. These scene masks will be described in Sec. 5.4.

[2]Institute of Systems and Robotics – University of Coimbra (www.isr.uc.pt)

[3]MakeWise, Engenharia de Sistemas de Informação (www.makewise.pt)

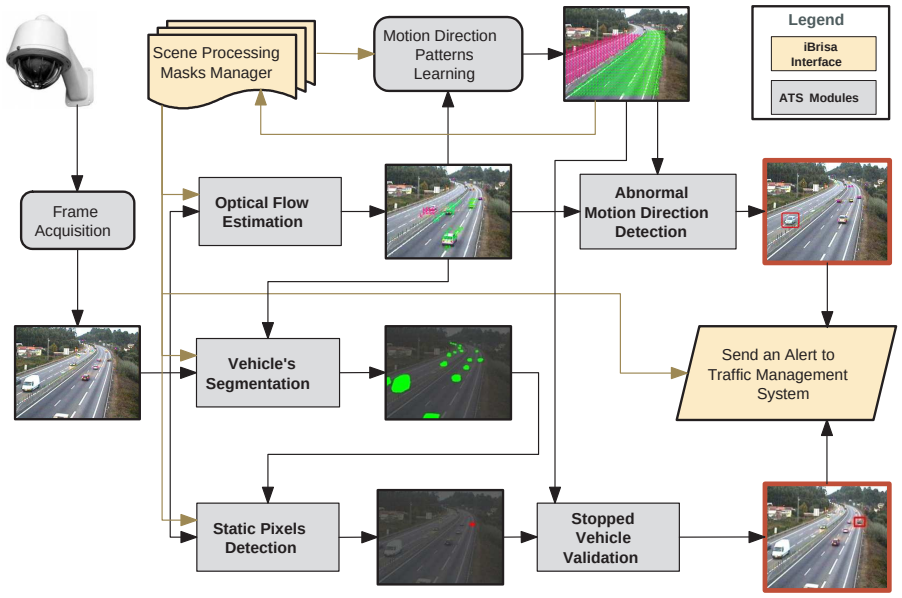


Figure 5.1: Flow chart of the developed Automatic Traffic Surveillance System prototype.

An event is called when an incident is detected by the system (Wrong Way Vehicle or Stopped Vehicle) and useful information about the event will be send. The event information is then send to iBrisa through the telematic management system and the human operator notified. The information provided through the event is defined below.

Incident Event Data
- Incident Type
- Confidence Level
- Direction
- Lane Label
- Time and Date
- Incident Video

With the provided information, the human operator will be able to verify if it is in fact a dangerous situations and activate the possible response plans.

In order to allow the analysis of false positives, some useful information about the system and the scene is stored in the machine where ATS System

is running. Some of the stored information is: sequence of images about the incident, image with the learned flow directions and module, video with the incident, scene masks, and frame rate. For a stopped vehicle detection event, it is also stored the chromatic representation of the pixel color frequency image, most frequent color pixel image, first background image, second background image, median background image, minimum difference between background models image, segmented pixels image, and pixels segmented only with T_l image. A private web site was developed to analyze this information about the incidents in a structured way, this web site will be presented in Sec. 5.5. This information is available via FTP server in every machine running the ATS System. In order to allow the analysis of miss detections, it is also stored video clips with a predefined duration of the last days.

This ATS System is presented as a shared library of Linux operating system and is supplied with some useful interface methods that are listed below:

startProcessing() – Starts the ATS System.

stopProcessing() – Stops the ATS System.

setMask(MaskData) – Set the scene masks.

forceEvent() – Force an event in order to test the communication.

initialize(ConfigurationParameters) – Set the configuration parameters.

getFrame(FrameType) – Returns a frame by type. Several inner processing images can be returned. This method can be called through a web browser and can be very useful for online debugging of the system.

doAction(ActionType) – Preforms an action (e.g.: start storing a video sequence, delete debug data, restart the system, etc.). This method can also be called through a web browser.

Some details about the integration of the library of the presented ATS System with the iBrisa will be given in Sec. 5.3.

5.3 System Integration

The architecture of the developed system integration adopts the SOA paradigm, where every instance of the native ATS library, executed in an individual machine, is made available through a service, known as ATS agent, which corresponds to a Java software component (see Fig. 5.2). This service is consumed by the central TMS application (Telematic Management System, developed using Java, as well), and is responsible for the interaction with all the existing ATS agents in the Brisa network (see Fig. 5.3), including the asynchronous reception of detection events originated by the native library. This TMS application is made available through a WebService interface, which allows it to be used by components of the iBrisa platform, specifically developed for the ATS system. With this architecture, every functionalities of the native library can be used and controlled by a remote central component. The robustness of this architecture was proved with the addition of new sites of ATS agents without disturbing the normal functioning of the ATS system. This system integration was developed by MakeWise.

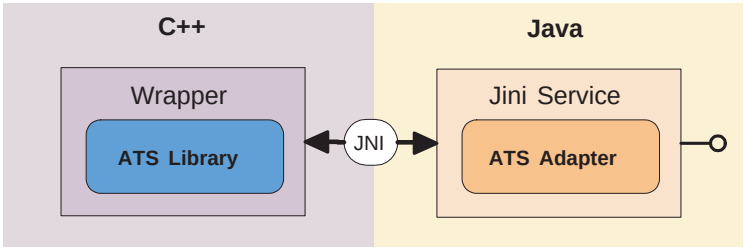


Figure 5.2: Integration of the ATS System library with ATS Adapter.

5.4 Scene Masks

The scene masks are data structures that store information about the scenario that is being analyzed. Three different scene mask are used in this ATS System in order to increase the system's performance: a) the scene processing mask; b) the lanes identification mask; and c) the learned flow patterns mask. These masks are stored in the database of iBrisa and are supplied to the ATS agent

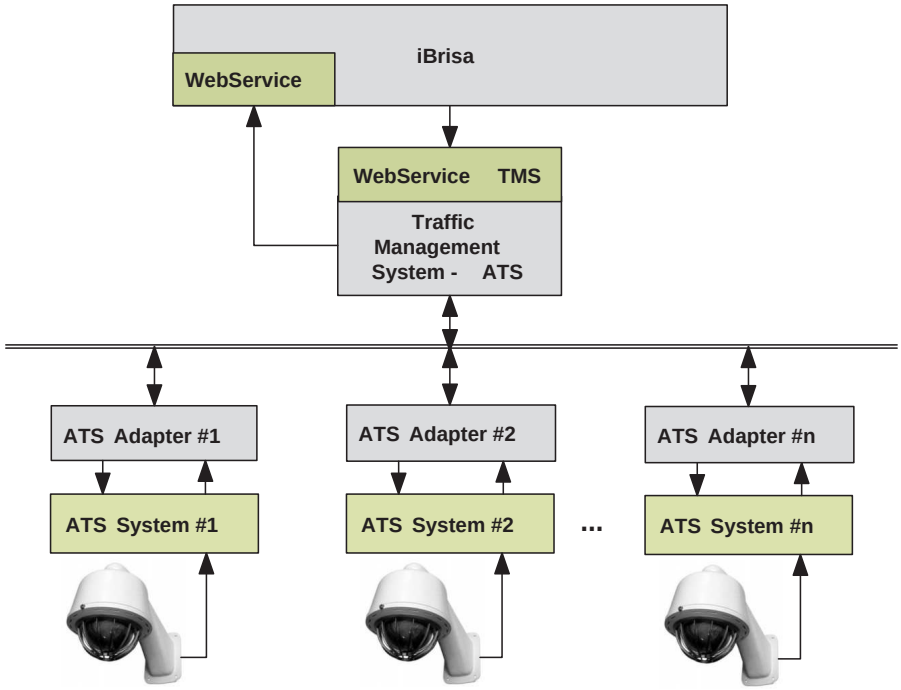


Figure 5.3: Integration of the ATS System in the Brisa surveillance cameras network and communication with iBrisa.

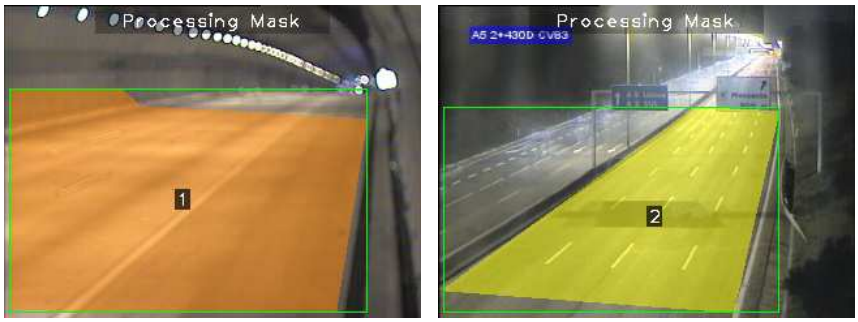
when it is required to start processing. For PTZ cameras, a set of masks is stored for each preset position.

The scene processing mask is used to define the area of the image that corresponds to the roads to perform the surveillance (two examples of this mask are depicted in Fig. 5.5). Only the pixels inside this mask are analyzed by the ATS System. This way, the frame rate of the overall system increases and the false positive rate can decrease. This mask is very important for the wrong way vehicle detection system in order to not process non highway roads; secondary roads near highways are discarded from the analysis (see Fig. 5.4). If, eventually, a two way secondary road is captured by the camera, the system is not able to correctly learn the flow direction patterns and several false positives can be detected. A processing mask is defined by the user and is stored in a database for each camera preset position.

The lane identification mask is used to identify each lane of the analyzed roads. This information is important to report, in the incident events, in which

(a) (b)

Figure 5.4: Two examples of scenarios with non wanted roads to analyze. a) Two way road on the left side of the highway; b) Bridge over the highway.

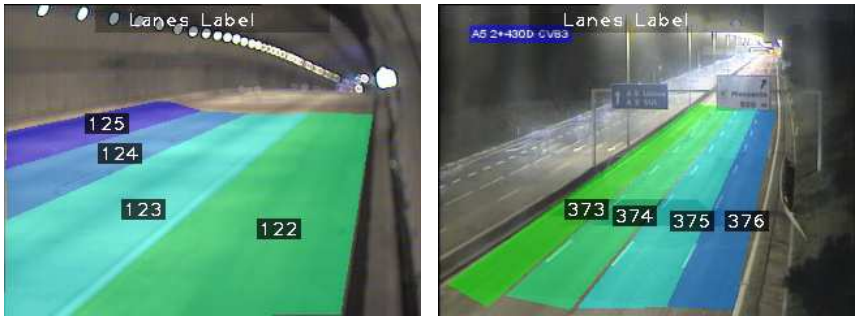


(a) (b)

Figure 5.5: Two examples of scene processing mask.

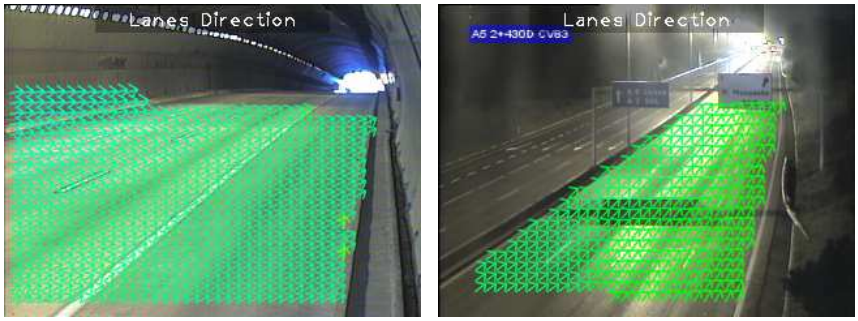
lane has the detected incident occurred. Two examples of the definition of this mask are depicted in Fig. 5.6. A lane identification mask is defined by the user and is stored in a database for each camera preset position.

The learned flow pattern mask is used to store the information of the learning process of the image flow patterns. This mask is loaded onto the system when it starts and it is used as an initial estimation for the learning process. After the learning process, this mask is updated in the database. Therefore, the learning period decreases. Two examples of the learned flow pattern mask are depicted in Fig. 5.7.



(a) (b)

Figure 5.6: Two examples of scene lanes identification mask.



(a) (b)

Figure 5.7: Two examples of scene learned flow patterns mask.

5.5 Online Analysis

The ATS Systems are installed in machines near the cameras, not to degrade the image quality in the video transmission/compression. The communication between the ATS System and the iBrisa is performed through a TCP/IP optical fiber private network along the highways. A web interface was developed to allow the real-time analysis of the ATS System. With this interface web site (authentication required) it is possible to analyze the results of the ATS in real-time through the use of the method *getFrame()* in the library. This web interface also allows analyzing the data of the incident detected events stored in the machine. PHP scripts are used to read the stored data trough an FTP server and then displaying the data in an intuitive way. Some examples of the web site are shown in Figs. 5.8, 5.9, 5.10, and 5.11.

Figure 5.8: Real-time processing images of a set of sites.

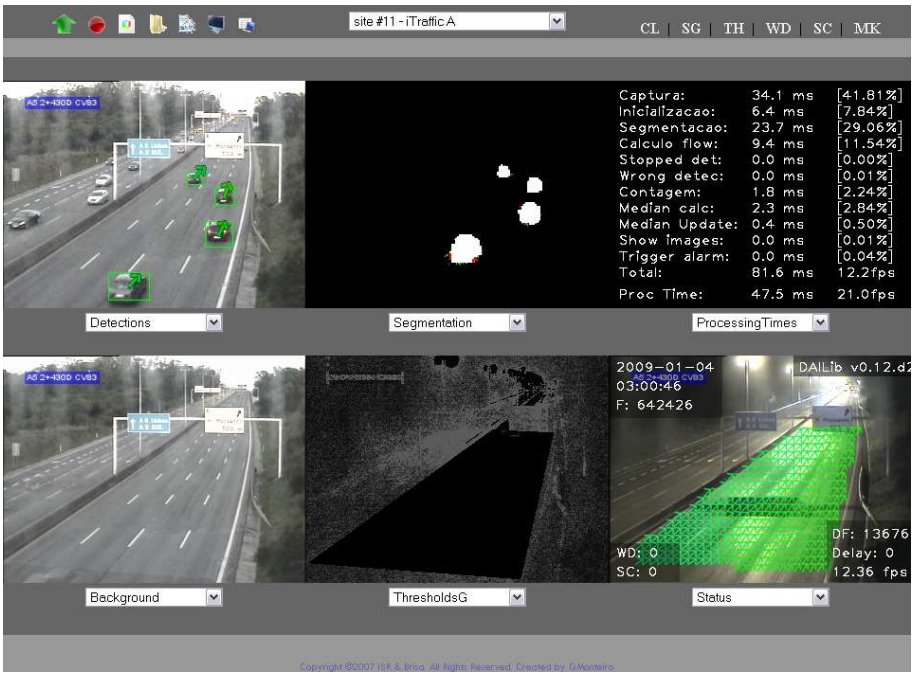


Figure 5.9: Real-time processing images of a site with detailed information.

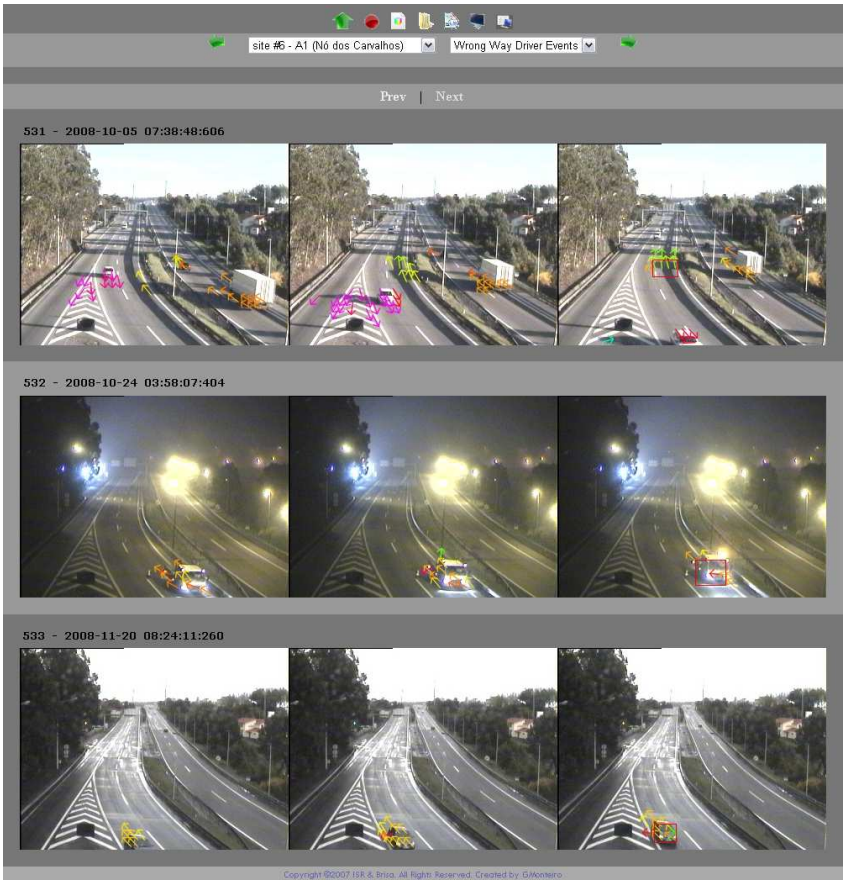


Figure 5.10: List of wrong way vehicle events detected for a site.

Figure 5.11: Stored data of a wrong way vehicle event.

Part IV

Final Notes

Chapter 6

Conclusion

6.1 Achieved Objectives

This thesis is the result of the study carried out in video surveillance solutions to detect dangerous behaviors in highways in order to improve road safety. The main objective was the study, analysis, proposal and development of architectures, models and algorithms to create a robust automatic traffic surveillance system for highways focused on the automatic detection of incidents.

Most of the main modules required for a real automatic traffic surveillance system have been considered during the research work, going from the low level foreground segmentation to the high level dangerous behavior detection. At the low level analysis, a robust segmentation process for outdoor scenarios based on background subtraction was presented. Shadow removal and optical flow algorithms were also analyzed and implemented to improve the reliability of the system. At the high level analysis, two of the most dangerous behaviors in highways were analyzed. A solution to detect stopped vehicles in the road or in the hard shoulder based on the pixel color history analysis was presented. It was also presented a solution to detect wrong way vehicles based on optical flow information analysis.

The solutions proposed on this thesis were tested and compared with other state-of-the-art methodologies. Several datasets of Portuguese highways from different sites, fields of view, weather conditions, hour of the day and with illumi-

nation changes were used to validate the effectiveness of the proposed solutions. Some public datasets were also used in order to evaluate comparative results of the proposed solutions with other methods presented in the literature. The performed experiments proved the robustness and the accuracy of the proposed methodologies to detect anomalous situations in highways. Experimental tests on site were also performed to the global traffic surveillance system in some Portuguese highways scenarios. Very good results were obtained during the several months of tests. These results proved the robustness of the proposed solutions in helping human operators detect dangerous behaviors in highways.

Together with the proposal of new paradigms and techniques, the presented research work was used in several modules of a real implementation of an automatic traffic surveillance system prototype. This automatic traffic surveillance system is currently being used in several sites of Portuguese highways to automatically detect vehicles circulating on the wrong direction and vehicles that stop in the road or in the hard shoulder.

6.2 Publications

During the research carried out on ITS technologies two main topics were focused. The first one addresses collision avoidance systems to be applied on board. The second one, the subject of this thesis, addresses automatic surveillance systems used to monitor drivers' behaviors in highways. This research work gave place to some publications listed below:

- "Stopped Vehicle Detection System for Outdoor Traffic Surveillance", in proceedings of RecPad 2008. Coimbra, Portugal, 2008.
- "Robust Segmentation for Outdoor Traffic Surveillance", in proceedings of IEEE ICIP 2008. San Diego, California, U.S.A., 2008.
- "Robust Segmentation Process to Detect Incidents on Highways", in Lecture Notes in Computer Science, Springer Berlin, volume 5112, 2008.
- "A Wrong Way Drivers Detection System Based In Optical Flow", in proceedings of IEEE ICIP 2007, San Antonio, Texas, U.S.A., 2007.
- "A Framework for Wrong Way Driver Detection Using Optical Flow", in Lecture Notes in Computer Science, Springer Berlin, volume 4633, 2007.
- "A Lidar and Vision-based Approach for Pedestrian and Vehicle Detection and Tracking", in proceedings of IEEE ITSC 2007, Seattle, U.S.A., 2007.
- "Towards a Robust Vision-based Obstacle Perception with Classifier Fusion in Cybercars", in Lecture Notes in Computer Science, Springer Berlin, volume 4739, 2007.
- "Vision-Based Pedestrian Detection Using Haar-Like Features", in journal Robótica, number 67, pages 16–20, 2007.
- "Tracking and Classification of Dynamic Obstacles Using Laser Range Finder and Vision", in proceedings of IEEE IROS 2006, Beijing, China, 2006.
- "Vision-Based Pedestrian Detection Using Haar-Like Features", in proceedings of Scientific Meeting of Robótica 2006, Guimarães, Portugal, 2006.

Bibliography

[1] Verri A. and Poggio T. Against quantitative optical flow. In *Proceedings of IEEE International Conference on Computer Vision 1987*, pages 171–180, 1987.

[2] E. H. Adelson and J. R. Bergen. The extraction of spatiotemporal energy in human and machine vision. *Proceedings of IEEE Workshop on Visual Motion*, 1986.

[3] Tickner A.H. and Poulton E.C. Monitoring up to 16 synthetic television pictures showing a great deal of movement. *Ergonomics*, 16:381–401, 1973.

[4] Waxman A.M., Wu J., and Bergholm F. Convected activation profiles and receptive fields for real time measurement of short range visual motion. *Proceedings of IEEE Computer Vision and Pattern Recognition*, pages 717–723, 1988.

[5] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

[6] E. Auvinet, E. Grossmann, C. Rougier, M. Dahmane, and J. Meunier. Left-luggage detection using homographies and simple heuristics. *Proceedings of IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2006*, pages 51–58, 2006.

[7] H. Barman, L. Haglund, H. Knutsson, and G.H. Granlund. Estimation of velocity, acceleration and disparity in timesequences. *Proceedings of IEEE Workshop on Visual Motion*, pages 44–51, 1991.

[8] J. L. Barron, D. J. Fleet, , and S. S. Beauchemin. Performance of optical flow techniques. In *Proceedings of IEEE Computer Vision and Pattern Recognition 1994*, pages 43–77, 1994.

[9] J. Batista, P. Peixoto, C. Fernandes, and M. Ribeiro. A dual-stage robust vehicle detection and tracking for real-time traffic monitoring. In *Proceedings of IEEE International Conference on Intelligent Transportation Systems 2006*, 2006.

[10] A. Bevilacqua and S. Vaccari. Real time detection of stopped vehicles in traffic scenes. *Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance 2007*, pages 266–270, 2007.

[11] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Proceedings of IEEE Computer Vision and Pattern Recognition 1997*, 1997.

[12] J. Bigun, G.H. Granlund, and J. Wiklund. Multidimensional orientation estimation with applications to texture analysis and optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:775–790, 1991.

[13] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm, 2002.

[14] T.E. Boult, R. Micheals, X. Gao, P. Lewis, C.Power, W. Yin, and A. Erkan. Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets. In *Proceedings of IEEE Workshop on Visual Surveillance 1999*, 1999.

[15] Simone Calderara, Andrea Prati, and Rita Cucchiara. Reliable background suppression for complex scenes. In *Proceedings of International Workshop on Video Surveillance and Sensor Networks 2006*, pages 211–214, 2006.

[16] A. Cavallaro and T. Ebrahimi. Change detection based on color edges. *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 445–448, 2001.

[17] A. Cavallaro and T. Ebrahimi. Accurate video object segmentation through change detection. *Proceedings of IEEE International Conference on on Multimedia and Expo*, pages 445–448, 2002.

[18] A. Cavallaro, O. Steiger, and T. Ebrahimi. Tracking video objects in cluttered background. In *IEEE Transactions on Circuits and Systems for Video Technology*, pages 575–584, 2005.

[19] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa. A system for video surveillance and monitoring. Number CMU-RI-TR-00-12, Pittsburgh, PA, 2000.

[20] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. In *IEEE Transactions on Pattern Analysis and Machine Intelligence 2003*, pages 1337–1342, 2003.

[21] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti. Improving shadow suppression in moving object detection with hsv color information. *Proceedings of IEEE Intelligent Transportation Systems 2001*, pages 334–339, 2001.

[22] Rita Cucchiara, Costantino Grana, Massimo Piccardi, and Andrea Prati. Detecting moving objects, ghosts, and shadows in video streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence 2003*, 25(10):1337–1342, 2003.

[23] J. M. del Rincn, J. E. Herrero-Jaraba, J. R. Gmez, and C. Orrite. Automatic left luggage detection and tracking using multi-camera ukf. *Proceedings of IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2006*, pages 59–66, 2006.

[24] J.H. Duncan and Tsai-Chia Chou. Temporal edges: The detection of motion and the computation of optical flow. *Proceedings of IEEE International Conference on Computer Vision 1988*, pages 374–382, 1988.

[25] Glazer F., Reynolds G., and Anandan P. Scene matching through hierarchical correlation. *Proceedings of IEEE Computer Vision and Pattern Recognition 1983*, pages 432–441, 1983.

[26] David J. Fleet and Allan D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5:77–104, 1990.

[27] G. Foresti. Object detection and tracking in time-varying and badly illuminated outdoor environments. In *SPIE Journal on Optical Engineering*, pages 255–2564, 1998.

[28] A. R. Francois and G. G. Medioni. Adaptive color background modeling for real-time segmentation of video streams. *Proceedings of International Conference on Imaging Science, Systems, and Technology*, pages 227–232, 1999.

[29] G.S.K. Fung, N.H.C. Yung, G.K.H. Pang, and A.H.S. Lai. Effective moving cast shadow detection for monocular color image sequences. *Proceedings of IEEE International Conference on Image Analysis and Processing*, pages 404–409, 2001.

[30] Daniel Grest, Jan-Michael Frahm, and Reinhard Koch. A color similarity measure for robust shadow removal in real-time. In *Proceedings of Vision Modeling and Visualization 2003*, 2003.

[31] S. Guler and M. K. Farrow. Abandoned object detection in crowded places. *Proceedings of IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2006*, pages 99–106, 2006.

[32] Nagel H. H. On the estimation of optical flow: Relations between different approaches and some new results. *AI*, 33:299–324, 1987.

[33] D. Haritaoglu, I. Harwood, and L. Davis. W4: Who? when? where? what? a real time system for detecting and tracking people. *Proceedings of International Conference on Face and Gesture Recognition 1998*, 1998.

[34] David J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision*, 1:279–302, 1988.

[35] Ellen C. Hildreth. The computation of the velocity field. *Proceedings of the Royal Society*, pages 189–220, 1984.

[36] D. Hong and W. Woo. A background subtraction for a vision-based user interface. *Proceedings of IEEE Pacific-Rim Conference On Multimedia*, 2003.

[37] Berthold K. P. Horn and Brian G. Rhunck. Determining optical flow, 1981.

[38] T. Horprasert and D. Harwood. A statistical approach for real-time robust background subtraction and shadow detection. *technical report, Computer Vision Laboratory, University of Maryland*, 1999.

[39] Jun-Wei Hsieh, Wen-Fong Hu, Chia-Jung Chang, and Yung-Sheng Chen. Shadow elimination for effective moving object detection by gaussian shadow modeling. *Image and Vision Computing*, 21:505–516, 2003.

[40] Jun-Wei Hsieh, Shih-Hao Yu, Yung-Sheng Chen, and Wen-Fong Hu. Automatic traffic surveillance system for vehicle tracking and classification. *IEEE Transaction on Intelligent Transportation Systems*, 7:175–187, 2006.

[41] H. W. S. Jabri, Z. Duric, and A. Rosenfeld. Detection and location of people in video images using adaptive fusion of color and edge information. *Proceedings of International Conference on Pattern Recognition 2000*, pages 627–630, 2000.

[42] Horst Bischof Jakob Santner, Matthias Rüther. Stereo vision based detection of wrong-way drivers. http://www.icg.tugraz.ac.at/research/ComputerVision/WrongWayDriverDetection_, November 2008.

[43] Omar Javed and Mubarak Shah. Tracking and object classification for automated surveillance. In *Proceedings of European Conference on Computer Vision 2002*, pages 439–443, 2002.

[44] P. Kaewtrakulpong and R. Bowden. Improved adaptive background mixture model for real-time tracking with shadow detection, 2001.

[45] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi. Occlusion robust vehicle detection utilizing spatio-temporal markov random filter model. In *Proceedings of 7th World Congress on ITS*, 2000.

[46] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. In *Real-time Imaging*, pages 167–256, 2005.

[47] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell. Towards robust automatic traffic scene analysis in real-time. In *Proceedings of International Conference on Pattern Recognition 1994*, pages 126–131, 1994.

[48] P. T. N. Krahnstoever, T. Sebastian, A. Perera, and R. Collins. Multi-view detection and tracking of travelers and luggage in mass transit environments. *Proceedings of IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2006*, pages 67–74, 2006.

[49] J.J. Little, H.H Bulthoff, and T. Poggio. Parallel optical flow using local voting. *Proceedings of IEEE International Conference on Computer Vision 1988*, pages 454–459, 1988.

[50] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of International Joint Conference on Artificial Intelligence 1981*, pages 674–679, 1981.

[51] C. Maduro, K. Batista, P. Peixoto, and J. Batista. Estimating vehicle velocity using rectified images. In *Proceedings of International Conference on Computer Vision Theory and Applications 2007*, 2007.

[52] C. Maduro, K. Batista, P. Peixoto, and J. Batista. Estimating vehicle velocity using rectified images. In *Proceedings of RecPad 2007*, 2007.

[53] C. Maduro, K. Batista, P. Peixoto, and J. Batista. Estimation of vehicle velocity and traffic intensity using rectified images. In *Proceedings of IEEE International Conference on Image Processing 2008*, 2008.

[54] D. Magee. Tracking multiple vehicles using foreground, background and motion models. In *Image and Vision Computing*, volume 22, pages 143–155, 2002.

[55] Stephen J. Mckenna, Sumer Jabri, Zoran Duric, Harry Wechsler, and Azriel Rosenfeld. Tracking groups of people. *Computer Vision and Image Understanding*, 80:42–56, 2000.

[56] Rudy Melli, Andrea Prati, Rita Cucchiara, and Lieven de Cock. Predictive and probabilistic tracking to detect stopped vehicles. In *Proceedings of Seventh IEEE Workshops on Application of Computer Vision*, volume 1, pages 388–393, 2005.

[57] G. Monteiro, J. Marcos, and J. Batista. Stopped vehicle detection system for outdoor traffic surveillance. In *proceedings of RecPad 2008*, 2008.

[58] G. Monteiro, J. Marcos, M. Ribeiro, and J. Batista. Robust segmentation for outdoor traffic surveillance. In *Proceedings of IEEE International Conference on Image Processing 2008*, volume 4633/2007, pages 1117–1127, 2008.

[59] G. Monteiro, J. Marcos, M. Ribeiro, and J. Batista. Robust segmentation process to detect incidents on highways. In *Lecture Notes in Computer Science, Springer Berlin*, volume 5112/2008, pages 110–121, 2008.

[60] G. Monteiro, M. Ribeiro, J. Marcos, and J. Batista. A framework for wrong way driver detection using optical flow. In *Lecture Notes in Computer Science, Springer Berlin*, volume 4633, 2007.

[61] G. Monteiro, M. Ribeiro, J. Marcos, and J. Batista. Wrong way drivers detection based on optical flow. In *Proceedings of IEEE International Conference on Image Processing 2007*, 2007.

[62] Fatih Porikli. Shadow flow: A recursive method to learn moving cast shadows. *Proceedings of IEEE International Conference on Computer Vision*, 2005.

[63] Fatih Porikli. Detection of temporarily static regions by processing video at different frame rates. *Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance 2007*, pages 236–241, 2007.

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

[65] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of IEEE Computer Vision and Pattern Recognition 1994*, pages 593–600, 1994.

[66] Kais Siala, Moez Chakchouk, Faten Chaieb, and Olfa Besbes. Moving shadow detection with support vector domain description in the color ratios space. *Proceedings of IEEE International Conference on Pattern Recognition 2004*, 4:384–387, 2004.

[67] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distribution of optical flow. In *Proceedings of IEEE Computer Vision and Pattern Recognition 1991*, pages 310–315, 1991.

[68] A. Singh. An estimation-theoretic framework for image-flow computation. *Proceedings of IEEE International Conference on Computer Vision 1990*, pages 168–177, 1990.

[69] K. Smith, P. Quelhas, and D. Gatica-Perez. Detecting abandoned luggage items in a public space. *Proceedings of IEEE International Workshop on Performance Evaluation of Tracking and Surveillance 2006*, pages 75–82, 2006.

[70] Jurgen Stauder and Jorn Ostermann. Detection of moving cast shadows for object segmentation. *IEEE Transactions on Multimedia*, 1:65–76, 1999.

[71] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of IEEE Computer Vision and Pattern Recognition 1999*, 1999.

[72] T. Thongkamwitoon, S. Aramvith, and T.H. Chalidabhongse. An adaptive real-time background subtraction and moving shadows detection. *Proceedings of IEEE International Conference on Multimedia and Expo 2004*, 2:1459–1462, 2004.

[73] B. L. Tseng, C. Y. Lin, and J. R. Smith. Real-time video surveillance for traffic monitoring using virtual line analysis. *Proceedings of IEEE International Conference on Multimedia and Expo 2002*, 2:541–544, 2002.

[74] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Journal Biological Cybernetics*, 60:79–87, 1988.

[75] Dong Xu, Xuelong Li, Zhengkai Liu, and Yuan Yuan. Cast shadow detection in video segmentation. *Pattern Recognition Letters*, 26(1):91–99, 2005.

[76] Akio Yoneyama, Chia-Hung Yeh, and C.-C. Jay Kuo. Robust vehicle and traffic information extraction for highway surveillance. *EURASIP Journal on Advances in Signal Processing*, 2005(1):2305–2321, 2005.

[77] Wei Zhang, Xiang Zhong Fang, and Xiaokang Yang. Moving cast shadows detection based on ratio edge. In *Proceedings of International Conference on Pattern Recognition 2006*, pages 73–76, 2006.