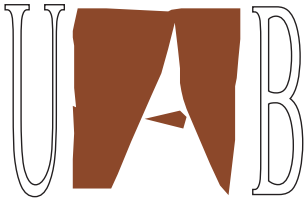


Escola Tècnica Superior d'Enginyeries
Departament de Ciències de la Computació

Model Features and Horizon Line Estimation for Pedestrian Detection in Advanced Driver Assistance Systems

Memòria del Treball Experimental presentat per en **David Gerónimo Gómez** i dirigit per **Antonio M. López Peña** dins del Programa de Doctorat en Informàtica, opció **Visió per Computador**, de la Universitat Autònoma de Barcelona.

El Dr. Antonio M. López Peña, professor titular del Departament de Ciències de la Computació de la Universitat Autònoma de Barcelona,

CERTIFICA

que la present memòria ha estat realitzada sota la seva direcció per en **David Gerónimo Gómez** i constitueix el Treball Experimental del *Programa de Tercer Cicle d'Informàtica, opció Visió per Computador*.

Bellaterra, Setembre, 2006

Antonio M. López Peña

Acknowledgements

This master thesis was created in 2004–2006 at the Computer Vision Center (CVC) at the University Autònoma of Barcelona (UAB). The financial support during that period was given by the FPU program of UAB and by the Spanish Ministry of Education and Science grant BES-2005-8864.

I would like to thank all the CVC people for day–by–day creating the warm atmosphere that makes the center be the very pleasant place to work it is. First to Prof J.Villanueva, director of the center, also to Dr C. Cañero, Dr B. Raducanu, O. Ramos, D. Rotger, D. Aldavert, A. Lapedriza, M. Rusiñol, A. Cervantes, X. Baró, M. Ferrer, A. Hernández, Dr F. Vilariño, J. Mas and all the ones that for sure I forget. To the administration, A. Espina, M.C. Merino, A.M. García, M. Culleré and P. Villa, and to the technical staff, J. Masoliver and R. Gómez, for always attending my needs in an effective manner.

To the members of the Advanced Driver Assistance Systems group, Dr A. López, Dr J. Serrat, Dr F. Lumbreras, Dr A. Sappa, D. Ponsa, C. Julià, J.M. Álvarez, for all their invaluable comments and advices in all the meetings and in informal chats.

Special thanks to Dr Joan Serrat, for introducing me to intelligent vehicles as the advisor of my BSc final project, about horizon line estimation, back two years ago.

Thanks also to Dr Angel Sappa, for the big efforts and the hard work made with the pitch estimation technique. The long and deep discussions we had were more pleasant and less tiring thanks to your always friendly attitude.

Daniel Ponsa also deserves great thanksgiving. For the fruitful discusions about many and many issues, specially for the extense answers you patiently gave to my endless questions. And of course for letting me use the vehicle detector code to construct and test the pedestrian detector. I have learnt a lot from you.

My foremost thanks to Dr Antonio López too. First, for believing I could be a competent coder in your group, two years back when I presented my project. Then, for again believing in me and being my PhD thesis director. My sincere gratitude for your patience, and specially for your invaluable advices, which made me assume the fact that you are always right.

Furthermore, I must thank my parents Ángel and Núria for always encouraging and supporting me, and for teaching me invaluable things like curiosity, spirit of sacrifice and perfectionism, which are always the guideline and inspiration for everything I do. Make this thanksgiving extensive to all my family for caring about me and giving me a hand whenever I needed.

Finally, last but definitely not least I want to thank my dear Verónica, who patiently bore my enthusiastic (and usually eternal) monologues about the last experiments, the current algorithm or the newest article. Thanks for your understanding and support, you are the person that gives me the strength to face new challenges always with illusion. This work is for you.

RESUM

Actualment, els Sistemes Avançats d'Assistencia al Conductor representen un important camp de recerca per millorar la seguretat vial. En aquesta tesina abordem la detecció de vianants fent ús de técniques de Visió per Computador basades en imatges de l'espectre visible durant el dia des d'un vehicle en moviment. Comencem per fer un anàlisis de l'estat de l'art per tal de definir la metodologia general d'un sistema de detecció de vianants complet, consistent en diversos mòduls, cadascun amb una tasca concreta, units per solucionar el problema sencer. A continuació, centrem el nostre treball en dos problemes concrets. Primer, proposem una tècnica d'estimació de l'horitzó que proporciona una sèrie de finestres candidates per a ser classificades com a vianant o no–vianant. Després, per a la classificació de finestres, em provat els millors conjunts de característiques per a construir un model de vianant. Els experiments amb característiques prèviament utilitzades en detecció de cares (Haar wavelets, histogrames d'orientació de contorns (EOH)) i amb d'altres proposades (invariants diferencials, orientació del tensor estructural) ens porten a la conclusió de que combinar conjunts complementaris com Haar i EOH és la millor opció. A més, la nostra tècnica basada en aquesta combinació de característiques i en un classificador AdaBoost aconsegueix el mateix rendiment i és menys costosa en temps de còmput quan el comparem amb els recentment proposats histogrames d'orientació de gradients (actualment les millors característiques per a aquest problema en la literatura). Tots els experiments utilitzen la nostra pròpia base de dades, enregistrada en condicions reals de tràfic a l'aire lliure. Finalment, presentem els resultats quan utilitzem la tècnica d'estimació de l'horizó juntament amb la classificació de finestres, aconseguint bons resultats en escenaris urbans complexes.

Paraules clau: *detecció de vianants, sistemes avancats d'assistencia a la conducció, aprenentatge, AdaBoost, línea d'horitzó, estimació del pitch, característiques invariants, Haar wavelets, histogrames d'orientació de contorns.*

ABSTRACT

Nowadays, Advanced Driver Assistance Systems represent an important field of research in order to improve traffic safety. In this master thesis we face the pedestrian detection task by using Computer Vision techniques based on images of the visible spectrum at daytime from a moving vehicle. We start by making an analysis of the state of the art to define the general methodology for a complete pedestrian detection system, consisting of several modules, each with a specific task, that are attached to solve the whole problem. Next, we focus our work in two concrete problems. First, we propose an horizon line estimation technique that will provide a number of candidate windows to be classified as pedestrians or non–pedestrians. Second, in the sense of the window classification itself, we test the best feature sets to construct a pedestrian model. Experiments with features previously tested in face detection (Haar wavelets, edge orientation histograms (EOH)) and new proposed ones (differential invariants, structure tensor orientation) lead us to concluding that combining complementary sets like Haar and EOH is the best option. Besides, our technique consisting in these combined features and an AdaBoost classifier results in the same performance and less–time consumtion when compared to the recently proposed histograms of oriented gradients (the current best features for this problem in the literature). All experiments are made using our own pedestrian database, recorded in real outdoor traffic conditions. Finally, we present the results when using the horizon line estimation technique together with the window classification, achieving good detection results in complex urban scenarios.

Keywords: *pedestrian detection, survey, advanced driver assistance systems, learning, AdaBoost, horizon line, pitch estimation, invariant features, Haar wavelets, edge orientation histograms.*

Contents

1 Introduction **1**
- 1.1 Advanced Driver Assistance Systems . . . 1
- 1.2 Pedestrian Protection . . . 1
- 1.3 Sensors of the ADAS . . . 3
- 1.4 Challenges of Pedestrian Detection . . . 7
- 1.5 Objectives . . . 8

2 State of the Art **11**
- 2.1 Preprocessing . . . 14
- 2.2 Foreground segmentation . . . 16
- 2.3 Object classification . . . 17
- 2.4 Verification/Refinement . . . 23
- 2.5 Tracking . . . 25
- 2.6 Application . . . 26
- 2.7 Discussion . . . 28
- 2.8 Summary . . . 31

3 Acquisition system and database **35**
- 3.1 Acquisition System . . . 35
- 3.2 CER–01 database . . . 38

4 Horizon Line Estimation **45**
- 4.1 Assumed pedestrian dimensions . . . 46
- 4.2 Previous approaches for defining the searching windows . . . 46
- 4.3 Proposed technique . . . 48
- 4.4 Experiments . . . 51
- 4.5 Summary . . . 52

5 Pedestrian Model **57**
- 5.1 Performance Evaluation . . . 57
- 5.2 Haar Wavelets . . . 59
  - 5.2.1 Feature definition . . . 59
  - 5.2.2 Template sets . . . 59
  - 5.2.3 Integral Image representation . . . 61
  - 5.2.4 Feature scaling . . . 62
  - 5.2.5 Tests . . . 63
  - 5.2.6 The importance of global features . . . 63
- 5.3 Differential Invariants and local jet . . . 67

5.3.1 Test . . . 67
5.4 Edge Orientation Histograms Features . . . 69
5.4.1 Preprocessing . . . 69
5.4.2 Feature definition . . . 69
5.4.3 Bin interpolation . . . 70
5.4.4 Tests . . . 71
5.5 Structure Tensor . . . 73
5.5.1 Tests . . . 74
5.6 Combining feature sets . . . 76
5.7 SVM–based Histograms of Oriented Gradients . . . 79
5.7.1 Feature definition . . . 79
5.7.2 Tuning the parameters . . . 79
5.7.3 Comparing HOGs with our best classifier . . . 81
5.8 Guidelines on Pedestrian Database construction . . . 84
5.9 Summary . . . 85

6 Results **87**

7 Conclusions and Future Work **95**

A AdaBoost **97**
A.1 Algorithm description . . . 97
A.2 Real AdaBoost . . . 99
A.3 Theoretical basis . . . 101
A.4 The Cascade Scheme . . . 101

B Support Vector Machines **103**
B.1 Algorithm description . . . 103

C Receiver Operating Characteristic (ROC) Curves **105**
C.1 Introduction . . . 105
C.2 Classifier performance . . . 105
C.3 ROC Curves . . . 107
C.4 Area Under ROC . . . 108

D Stereo **111**
D.1 Main concepts . . . 111
D.2 V–disparity representation . . . 113
D.3 Distance computation using camera parameters . . . 113

E Glossary **115**

Chapter 1

Introduction

1.1 Advanced Driver Assistance Systems

Since the popularization of automobiles during the 20th century, traffic accidents have become an important cause of fatality in modern countries. For instance, in 2002, motor vehicle accidents represented the half of non–natural deaths in the United States [80]. In order to improve safety, the industry has progressively developed different elements of increasing complexity and performance: from turn signals and seat belts to Anti–lock Braking Systems (ABS) and internal Airbags, as well as the more recent Electronic Stabilization Programmes (ESP), Advanced Front Lighting Systems (AFLS) and external Airbags.

In the last decade, research has also moved towards even more intelligent onboard systems that aim to anticipate and prevent the accidents, or at least, minimize their effects when unavoidable. They are referred as Advanced Driver Assistance Systems (ADAS), in the sense that they assist the driver to take decisions, provide warnings in dangerous driving situations, and even at taking automatic evasive actions in extreme cases. The difficulty here comes from the fact that, to take those actions, the main component of such systems can not only be well–known physical laws (as happens for ABS or ESP) but a not low degree of artificial intelligence is mandatory regarding the understanding of the surrounding scene. Examples of ADAS are the Adaptive Cruise Control (ACC), which automatically adjusts the own vehicle's speed to maintain a safe gap with a preceding vehicle in the same lane; the Lane Departure Warning (LDW), which warns the driver in case the vehicle starts to leave the lane inadvertently (i.e. turn signal not activated); or the Lane Keeping Assistance (LKA), which adds torque to the steering wheel to reduce the minute steering adjustments that must be made by drivers and therefore reduce fatigue on long trips.

It is foreseen that engineering such systems requires a great effort in multidisciplinary research and development: sensors, Artificial Intelligence (including Computer Vision), human machine interfaces, human factors (e.g. ACC plus LKA could be too much degree of driver support), legal issues, etc. The books [113, 14] can be consulted for more details on ADAS aspects.

1.2 Pedestrian Protection

In 2003, the Economic Commission for Europe [38] reported almost $150,000$ injured and $7,000$ killed in pedestrian accidents only in the European Union, representing the second source of traffic injuries

and fatalities, just after accidents involving only vehicle passengers (Table 1.1). The same year, in the United States, almost 5,000 people were killed and 70,000 were injured in the same type of accidents [39]. These terrible **statistics encourage research in pedestrian protection, and the present master thesis fits in this very active field of the ADAS**. Such challenge has already led to the creation of consortiums and organizations to join efforts. Some examples [14] are the research programmes of the European Union: PROTECTOR (2000–2003, 4.4 million euros) and its successor SAVE–U [95] (until 2005, 8 million euros), both in the 5th–FP with Volkswagen AG, Daimler–Chrysler AG and Siemens VDO as partners; and APALACI [2] (until 2008), inside the project PReVENT [89] (70 million euros) from the 6th–FP, with companies like Daimler–Chrysler, FIAT, Volvo, Robert Bosch GmbH, IBEO GmbH and Parma University.

Table 1.1: Road traffic accidents by category in the E.U. during 2003 [38].

Category	Killed	Injured	Total
Cars	20,365	825,186	845,551
Pedestrians	**6,902**	**147,564**	**154,466**
Cycles	2,704	138,012	140,716
Motorcycles	4,029	113,959	117,988
Mopeds	1,376	86,138	87,514
Other	2,693	93,891	96,584
Total	38,069	1,404,750	1,442,819

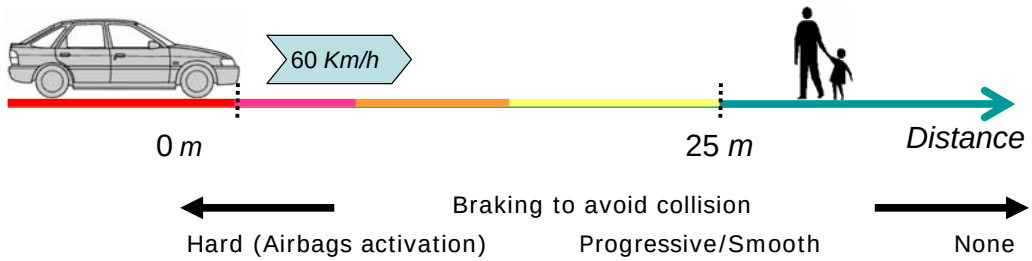


Figure 1.1: An automatic system must take different actions according to the time-to-collision between a host vehicle and a pedestrian. For instance, at 60 km/h and assuming a non-moving pedestrian, if he/she is detected at less than 25m in the vehicle's path, then both internal and external airbags must be deployed and a hard brake must be performed as well, while for further away distances a progressive smooth brake may be sufficient to avoid a collision.

The objective of a pedestrian protection system (PPS) is formally defined as the detection of presence of people (both static and moving) in a defined range of distances, in order to both provide information to the driver and perform evasive or braking actions on the host vehicle (Fig. 1.1). Since we want to detect pedestrians before they enter in the region where collision is unavoidable, the maximum distance for detection increases with both the maximum vehicle's speed where the PPS must be operative and the reaction of the PPS (time to identify the target and finalize the appropriate actions).

The above mentioned European programs and other National ones (France, Germany, etc) have not engineered a perfect PPS, rather they have mainly revealed the inherent difficulty of doing it, specially referring the robust detection of pedestrians in real–time, therefore, they are the basis for further research. Notice that a PPS requires a very challenging trade–off between true positives, to avoid collisions, and false positives, to avoid irreversible and expensive actions. Of course, the trade–off is more demanding when analyzing close regions than when doing so for further distances. Other statistics collected from vehicle–to–pedestrian accidents can also help to design a PPS with better practical trade–offs, for instance, it seems [44] that the more relevant scenario is a walking/running pedestrian crossing the road in front of the host vehicle when this moves at a speed up to 60Km/h.

1.3 Sensors of the ADAS

ADAS applications can be based on line–scan active sensors, this is, sensors that transmit signals and observe their reflection from the objects present in a working space confined in an horizontal plane. For instance, for ACC we can find [14] radar approaches where the signals are radio waves[1] reaching detection distances of about 150m with horizontal field of view (HFOV) of 16 degrees; or laser–based[2] approaches where the signal is infrared light, being the lidar[3] an example reaching detection distances of at least 100m for an HFOV of 18 degrees, and being another example the so–called *laser scanners* that reach detection distances of even 250m for an HFOV of 270 degrees by emitting infrared laser pulses. These sensors can be engineered to have more than a single line (horizontal plane) in order to have vertical information and gain robustness. On the other hand, it seems that for these active sensors is quite unreliable to distinguish pedestrians from other objects in urban environments.

ADAS applications can also be based on matricial–scan passive sensors, such as cameras (CCD or CMOS) operating either in the visible spectrum or in the infrared one. For instance, it is reported [14] an ACC system based on a high dynamic range CMOS camera operating in the visible spectrum that detects vehicles up to 60m for HFOV of 40 degrees and 30 degrees of vertical field of view (VFOV).

Each type of sensor has advantages and drawbacks compared with the others. For instance, active sensors are very good at estimating distances in real–time, however cameras operating in the visible spectrum have the advantage of a high spatial resolution (both horizontally and vertically), high dynamic range and hold interesting information from cues like texture or color. On the other hand, cameras, as well as lidar and laser scanners, suffer in adverse conditions like in presence of fog or heavy rain, while radar offers robust performance in the presence of virtually all weather conditions. Another point about active sensors is whether they would interfere each other in a massive presence of vehicles with such systems.

In a rough comparison we could say that active sensors are better to detect object presence providing superior range estimates and camera sensors are better to classify them. But, of course, researchers working with active sensors try to perform object classification while researchers in the field of Computer Vision use stereo techniques to obtain also distance information (a range of 60m with HFOV of 44 degrees is possible).

In addition, from a business point of view, passive sensors are less expensive and easy to integrate in a car than active ones. For instance, an ACC based on lidar is sold for about \$800, one based on radar for about \$2000, nowadays the price of a laser scanner is about $\$20,000$ (but probably it will go below radar in the near future), while an ACC based on a camera can be probably sold for about \$90.

[1] Radar: RAdio Detection And Ranging.
[2] Laser: Light Amplification by the Stimulated Emission of Radiation.
[3] Lidar: LIght Detection And Ranging.

In the particular case of pedestrian detection, passive–vision–based approaches are widely accepted as the most promising ones [40, 41, 47, 48, 50, 108, 33, 34, 49, 8, 12, 11, 10, 100, 57, 119, 6, 19, 72, 13, 104, 56]. In fact, this is not surprising since vision is by far the most used human sense for driving. Besides, the acquisition rate of modern cameras is sufficient to address the problem of pedestrian detection, in fact, the bottleneck will be the understanding of the images.

Of course, as for other applications, in order to detect pedestrians Computer Vision techniques can be used alone as the mentioned references or can be fused with active sensors [76, 98, 77, 74, 35]. Such fusion can be either low level (e.g. with a proper calibration, the analysis of radar data can point out regions of interest in an image that Computer Vision can classify as pedestrian or not) or high level (e.g. the results of a radar based classifier and a Computer Vision based one could be fused as complementary information). The aim of fusion is to reach real-time, improve robustness or both. Lombardi [69] says that pedestrian protection will probably be more about *sensing* pedestrians than only *seeing* them.

In any case, the conclusion is that camera sensors and Computer Vision play a central role in pedestrian detection (*to sense* includes *to see*), therefore, in engineering PPS and, of course, the better a technique based on a specific sensor type is, the better its contribution to a system that fuses the capabilities of different types of them will be. Accordingly, the first constraint for our work is that **we want to address the problem of detecting pedestrians, both in movement and static (but standing up, i.e. not lying on the road), by using only Computer Vision techniques**.

In addition, attending to the last technologies, another important decision is whether to work with images of the visible spectrum or the thermal infrared (TIR)[4] (up to a few years only available for military purposes) (Fig. 1.2). In fact, for this master thesis the decision is just a matter of possibilities, e.g., we have cameras working in the visible spectrum but not in the TIR one, therefore, **we assume pedestrian detection from images of the visible spectrum**. Nevertheless, we want to point out below the *pros* and *cons* of each sensor type, so that we will see that working in the visible spectrum is a reliable decision beyond the fact that we didn't actually have alternative.

TIR cameras are claimed as the best for nighttime since temperature is an information available both at day and night, while the visible spectrum either appears *too black* and with poor contrast due to the lack of ambient light at night. In addition, objects can be camouflaged due to glaring sources of light.

On the other hand, TIR sensors usually have half the resolution of sensors in the visible spectrum, and generate blurrier images with less background/foreground contrast. Besides, nowadays TIR cameras are still much more expensive than the ones working on the visible spectrum (about $\$10,000$), although efforts are in progress to come up with TIR sensors of low cost.

Moreover, there are on–going steps that could make feasible the use of non–TIR cameras: new headlights systems are improving both in illumination quality and adaptation to the environment (headlights that rotate at curves, that adapt to next vehicle, etc.) and the range of distances where pedestrian detection is supposed to be more useful (up to about 50m far away) can be reached with low beams; CMOS based cameras have the potential of both high dynamic range and auto–gain/shutter/etc. control for each single pixel so that it is possible to increase visibility at low illuminated areas at the same time that at saturated areas (up to now this possibility has not been explored since usual CMOS cameras only perform global auto–gain/shutter/etc. control, not pixel/region–wise); in fact, many CMOS sensors

[4] The sensors to which we refer sometimes are termed as of *night vision*, *thermal infrared*, *infrared* alone, or *far infrared (FIR)*. In fact, the term FIR is quite common. However, here we prefer the abbreviation *TIR* referring to thermal infrared. The reason is that the sensors of this type used for pedestrian detection measure the relative, not absolute, thermal radiation in the range 6-15μm, but according to some sources this is in fact the range corresponding to the so–called *long wavelength IR* (LWIR) while the actual far infrared (FIR) range runs from 15μm to 1,000μm. However, according to other sources the FIR starts at 4μm. Thus, the frontiers seem not to be well established. Accordingly, we prefer here the term TIR, which reminds us that we refer to relative thermal information, this is, images where the brighter a pixel, the warmer the underlaying region of the world.

Figure 1.2: TIR versus visible light [74]. The same scene taken form a TIR camera (left) and a regular color one (right) at daytime. In the TIR image the brighter a pixel value the warmer the underlying region of the world.

also have high sensibility to the near infrared (NIR) spectrum[5], therefore, with a light emitting at the NIR range of wavelengths, visibility at night could be increased (e.g. as can be appreciated in Fig. 1.3, modern xenon headlights illuminate better than classical Halogen ones). Notice that for nighttime, if we don't use TIR sensors, then the acquisition system must be somehow active (emit proper light) to be useful. Anyway, these arguments in favor of non–TIR cameras are still speculations, furthermore it can be foreseen that, for cost reasons, with the same sensor more than one application would be addressed (e.g. pedestrian detection and ACC), therefore, even more considerations must be taken into account to decide whether to use a TIR camera or not, combined with other sensors or not. On the other hand, some industrial and university research groups [5, 105, 1, 20] are investigating the *NIR approach* for nighttime pedestrian detection, therefore, it is a plausible line of work.

In addition, the literature [108, 33, 12, 34, 10, 119, 6, 19, 72, 13, 56, 104] on TIR–based pedestrian detection reveals that, in fact, to work with such images is also difficult:

- Pedestrians are not the only *hot* object, there are others as vehicles, light poles, traffic signs heated by the sun, etc.
- Body–trunk–regions of a pedestrian tend to be darker than head–regions and hand–regions (Fig. 1.2).
- The relative temperature changes during the day and along the year (Fig. 1.4)
- As for the visual spectrum, the TIR one is affected by heavy rain and fog, but also by humidity and wind.
- The intensity values in an infrared image depend also in other factors besides temperature, such as object surface properties (emissivity, reflectivity, transmissivity), surface direction, etc.

Therefore, rough segmentation methods understood as *attention mechanisms* are probably easier to implement for TIR images, specially if acquired at nighttime, than for images of the visual spectrum, however, to classify any segmented area as containing a pedestrian or not will be still a very difficult task. In fact, in order to work with TIR images, many of the current approaches (e.g. [12, 10, 72, 104]) try to

[5]NIR spectrum goes from 0,75μm to 1,4μm, this is, it is the continuation of the visible spectrum which goes from 0,4 to 0,75μm.

Figure 1.3: Illumination based on halogen lights (left) and xenon (right) from the web page of the Spanish division of Hella. Xenon lamps generate a spectrum that is close to natural daylight.

adapt techniques initially devised for images of the visual spectrum, including having object distances by stereo vision [108, 6].

Thus, the work done in this master thesis for images on the visible spectrum could be adapted/tested for TIR images in the future. Accordingly, now on **we assume that our input data are images of the visible spectrum taken at daytime and outdoor from a vehicle that can be in movement**.

Of course, then we may ask ourselves if, beyond the fact that daytime is assumed in many previous works (as we will see later), it is actually a relevant scenario. In other words, what report the statistics about. This point is very varying: if we look to a paper that is based on infrared images, then somewhere in the introduction we find a sentence saying *the majority of accidents involving collisions with pedestrians take place at night*, while if the work is based on images from the visible spectrum then we can find *the opposite sentence*. Indeed, there are sources of all kinds: e.g. in [39] it is stated that the 35% of the pedestrian fatalities in the USA during 2003 happens at daytime; [78] shows that the 61% of the pedestrian accidents in Japan during 1999 occur at daytime but [3] says 32% also for Japan in 1998; in the context of the PROTECTOR project [92] using data from most OECD[6] it was stated that in 1997 about 70% of pedestrian accidents were at daytime; etc. Well, we have not given a very close look to the reason why statistics vary that much (the definition of injury?, the country way of life?, etc.) because we think that even assuming something like the 30% the daytime scenario is relevant. We may say however, that daytime is the period that concentrates more movement of people in urban areas and also that many accidents involving pedestrians are not due to bad visibility conditions but due to distraction, so probably statistics like the mentioned 70% may be realistic. For instance, the report that points out the above mentioned 61% of Japan in 1999 also says that most of these daytime accidents involved people below 12 years or above 65 because daytime is their main activity period [78].

On the other hand, it seems well accepted that the most severe injuries, including pedestrian death, happen at nighttime due to either the higher speed of the vehicles which comes in turn due to the lower traffic density, or the poorer visibility conditions, which is much worse for senior drivers (e.g. at the age of 60 we need twice as much light to see than at the age of 30 [32]). Therefore, although we insist that daytime is a very relevant scenario, nighttime is also crucial and we will consider it in future research.

[6]OECD: Organization for Economic Cooperation and Development.

Figure 1.4: A human body in a cold night appears white *(a)*; but in the same scene different pedestrians can appear as white or not, e.g. in *(b)*, the further away pedestrian goes out of a warm car and appears white but the closer pedestrian, after walking for a while, has cold clothes and does not appear as full–body white; a human body can even appear black in a very hot summer day scenario *(c)*. Even the same pedestrian can appear having different contrast with the background for different body parts *(d)(e)*. These images are from [56].

1.4 Challenges of Pedestrian Detection

Once stated that we are going to work with images of the visible spectrum at daytime, let's point out which are the difficulties. We can start by common challenges for on–board vision systems of the ADAS. They arise from dealing with a mobile platform in an outdoor scenario, this is, variability is present everywhere:

- Images are acquired from a moving camera, contain objects with unknown movement that are placed at different distances in front of a background that continuously changes both in content and in spectral conditions (e.g. illumination changes due to weather, sunrise or sunset; presence of shadows). Hence, Computer Vision algorithms must detect, recognize and track objects of interest with a potentially high intra–class variability, seen from different angles, sizes, and different light conditions in a changing background. In fact, the non-static background makes almost impossible to take advantage of several techniques commonly used in other vision-based applications to detect objects (like background substraction in surveillance systems to help detecting people).
- Computation of distances to objects is quite difficult using a single camera. The reason is that the horizon line jumps up and down when the vehicle brakes, accelerates, has different passenger distributions, when the asphalt has irregularities, etc. Therefore, stereo techniques are needed if we want such information, or we must assume a robust way to compute the horizon line dynamically (e.g. assuming constant road orientation, usually known as flat road assumption, and using lane markings if there are present, etc.).
- Moreover, real–time requirements usually go from 5Hz to 25Hz, which is very challenging given the degree of complexity needed to address such high variability.
- Besides, sometimes more than one image is required to take any decision, but low latency is mandatory since anticipation to hazards is crucial. We must take into account that both awareness of the driver of any warning and automatic mechanical/electronic actions will take also time.
- High robustness is not an option. Not clear warnings could confound the driver while false alarms could lead the driver to distrust the system in future warnings. Besides, a wrong automatic reaction of the vehicle can lead to irreversible situations.

In this context, pedestrian detection from images is one of the most challenging problems of ADAS:

- Pedestrians are non–rigid and aspect–changing objects, this is, they not only are seen from different view angles wearing different clothes but also they appear at different articulated poses.
- Pedestrian detection makes sense mainly in urban areas where, on the contrary to scenarios as highways, a high number of different clutter objects and textured backgrounds are present (e.g. near buildings, vehicles, poles, trees, etc.), sometimes leading to situations where the pedestrian is partially occluded (e.g. the pedestrian starts to cross the road coming behind a parked van; the pedestrian carries bags; etc.).
- Crowd: pedestrians not always appear in isolation but many times they walk in group.
- In such urban scenarios the road goes up and down more than in fast roads and we do not always have lane markings information. Therefore, the reliable computation of the distance to a pedestrian from a single image is near impossible, i.e. stereo is going to be needed to have distance information[7].
- Robustness is specially critical. For instance, imaging a system that automatically must decide whether to activate external airbags or not in case a collision with a pedestrian is unavoidable. If it is true that there is a pedestrian in danger but the airbags are not deployed (miss–detection), then the pedestrian can be seriously injured. On the other hand, to deploy such airbags is an irreversible action with an unnecessary high economical cost if there was not a pedestrian (false alarm).
- The previous observations make quite difficult to reach real–time and low latency, but pedestrian protection is very demanding regarding anticipation of proper actions as deploying airbags or automatic braking.

We can summarize the above mentioned challenges/difficulties as follows:

- **Cluttered and changing background**.
- **Very high intra–class variability**.
- **Targets and camera following different unknown movements**.
- **Fast system reaction is required (real–time plus low latency)**.
- **Very demanding robustness**.

1.5 Objectives

At this point we have highlighted that statistics of modern countries encourages research in pedestrian protection systems to prevent accidents and minimize injuries, fact that drives our interest towards this very active field of the advanced driver assistance systems. Accordingly, we want to address the problem of detecting pedestrians, both in movement and static (standing up) which is by far the most relevant scenario regarding pedestrian accidents.

Among different possibilities, Computer Vision techniques are clearly key participants (perhaps not working alone but in fusion with other systems) therefore, in this master thesis we explore pedestrian detection only from the side of Computer Vision. More specifically we restrict ourselves to pedestrian detection from images of the visible spectrum taken at daytime, of course, outdoor and from a moving

[7] In fact, time–to–contact can be computed from only the change in the perceived size of objects if they are tracked [29]. However, this technique is hardly applicable to pedestrians since their size is not constant in time.

vehicle. Although we start by excluding nighttime images, daytime is also relevant according to the accidents statistics (after all it is when most of the pedestrian activity usually holds).

Given this context more specific objectives are outlined in the following:

- We want to review the literature of the field addressing the same problem (i.e. pedestrian detection at daytime from images). From the review, we want to define the architecture of modules in which all works can fit so that we can compare them and gain understanding about the problem. We believe this is more useful for us (and hopefully for the reader) than just enumerating in sequence all the steps of each relevant work, one work behind another, as usually done in other reviews. Besides, we expose how the challenges exposed in Sect. 1.4 are faced in terms of the proposed architecture. We develop this objective in Chapt. 2.

- We want to establish requirements to our images that correspond to a useful PPS. In particular, we want to work with an acquisition system that allows us to detect pedestrians in the range from 5m to 50m, assuming standard pedestrian sizes. Moreover, the chosen environment for pedestrian detection is the most relevant but challenging, we refer to urban scenarios. Besides, the geometry of the acquisition system should not be too biased to pedestrian detection but also allowing to incorporate other ADAS applications, in particular lane markings detection and vehicle detection. Of course, a vehicle can harm a pedestrian going backwards or laterally, but we assume the vehicle going forward and, thus, our system will be forward–facing. Again this is, in fact, the most relevant scenario (also for the other mentioned applications). Details are given in Chapt. 3, where the importance and difficulty of having good databases is also addressed. In particular, we show that we have collected images presenting the variabilities exposed in Sect. 1.4.

- As we can imagine, and the literature review will confirm, a key issue is to reduce the number of possible image regions to explore but without missing regions with pedestrians. This could have two positive effects: approaching the system to real–time and reducing the number of false positives. Notice that false positives could give rise to false alarms at the system level, with the potential problem of making the PPS useless.

  The simplest information to use are pedestrian size constraints (PSC) based on the projective geometry of the acquisition system and the size of the pedestrians. In this master thesis we restrict ourselves to the use of these constraints. However, they are actually useful only if we know the horizon line in the image, which changes from frame to frame. Unfortunately, horizon line recovery is not an easy problem, and exploring a reliable method is also an objective of this master thesis. We go into this issue in Chapt. 4.

- The last but most important objective in this master thesis is the implementation of some of the most relevant classifiers used in pedestrian detection, as well as new ones. Then, compare them by using ROC curves. For the development of the classifiers we use supervised learning. This means that, during the learning phase, we provide examples (pedestrians) and counter–examples (non–pedestrians). To obtain them we use images of our own databases. This is, the ROIs having a pedestrian or other content come from these images. The way of selecting such ROIs is just those surpassing the PSC, which is an important point since some of the reviewed works present results build upon databases where the corresponding example ROIs are of more quality than in automotive applications and the negative examples are not sufficiently representative of urban environments around a road.

  The particular classifiers that we study are the following. We implement the one using the so–called histograms of oriented gradients (HOG) as features and support vector machines (SVM) as learning method. This was claimed in 2005 as the classifier with the best performance for pedestrian detection. We also implement a series of classifiers used in other contexts as face detection. In particular, based on the different types of Haar wavelet sets and edge orientation

histograms (EOH) as features and Real AdaBoost as learning method (however, only using one cascade to facilitate the ROC based comparison). We also include as feature differential invariants computed from Haar–like directional derivatives. Some of the feature sets used with Real AdaBoost are also combined (e.g. Haar+EOH, Haar+Differential Invariants, etc.) since in other contexts these combinations provide improved ROCs (e.g. Haar+EOH in face detection). Finally, since local gradient orientation appears in most of the best classifiers we study the use of the structure tensor to work with a less local type of orientations.

All the classifiers developed in our study try to detect pedestrians in a global (holistic) way. We postpone for future work component–based classifiers. In fact, such classifiers are based on global–like classifiers.

Another issue is the time consumption of each classifier. We use efficient computation methods as integral images and so on. However, we have not done a deep comparison regarding time. The reason is that for doing that, work out of the scope of this master thesis is required (left as near future work). We refer to issues as code optimizations, developing more cascades for each AdaBoost–based classifier, studying the sharing of features between classifiers (e.g. for both vehicle and pedestrian detection), etc.

Our work about pedestrian classifiers is developed in Chapt. 5.

To illustrate the practical result of the work done to achieve our objectives we include Chapt. 6, where we apply the horizon line estimation plus pedestrian classifier to whole image sequences taken downtown. Chapter 7 summarizes the conclusions of the present work and the future work that such conclusions suggest. Finally, Chapters A–E are included as appendices where important *tools* used in this master thesis are explained up to certain level of detail (that would annoy if included in the main chapters).

Chapter 2

State of the Art

First works in pedestrian/people detection from images were not aimed by PPS but by surveillance applications. In practice this means that they are based on a static camera (or several of them) giving rise to algorithms that can assume a non–changing background (or at least without sudden changes) and no egomotion. In [81] we can find a review of works that use background substraction. Thus new objects in specific regions of the background with specific aspect ratio and movement dynamics can be assumed to be pedestrians. Classification plus learning approaches based in spatio–temporal features and appearance have been also tested [84, 111].

During the late 90s start to appear pedestrian detectors based on images for PPS. Although the objective of the processing can seem the same than for surveillance, i.e. to detect people, the challenges that arise from the requirements of the PPS (Sect. 1.4) drive the research to new approaches. The aim of this section is to review the main literature of this field of the ADAS in order to put in context the work of this master thesis (beyond the already mentioned restriction of working with daytime images of the visible spectrum). Thus, we skip works either from the context of surveillance or based on TIR images. In fact, regarding the classification module, that we will see later, the techniques for TIR images are most of the times just adaptations from techniques used for daytime images of the visible spectrum.

There are several ways of addressing the review of the state of the art. For instance, either summarizing the most relevant papers one by one in isolation or providing a taxonomy based on a given criterium (e.g. the key–technique of each work, the sensor used, how a particular issue is addressed, etc.). Different examples are [69, 48, 23, 81]. In the following we present a more up to date review of pedestrian detection works. First we outline the main modules a pedestrian detector for PPS may/should have in order to address the difficulties pointed out in Sect. 1.4. Then we comment the main contributions done for each module. Tables 2.1 and 2.2 offer a summarized view of the state of the art.

We propose the following modules (Fig. 2.1):

- **Preprocessing.** Takes the input data from the camera and prepares it to the further processing. For example, preparing multi–resolution images, providing image–stabilization between frames, etc.

- **Foreground Segmentation.** Extracts interesting regions of the image (ROIs) to be sent to the next module for classification, i.e. regions likely to contain a pedestrian, trying to avoid as many background ROIs as possible. A key of this module is not to miss pedestrians, otherwise the next module will not be able to correct the error. Here we can find segmentation based on cues as stereo, symmetry, etc.

- **Object Classification.** Receives the ROIs and classifies them as containing some pedestrian or not. Two main components can be found in building a classifier: 1) *features* (Haar wavelets, edges, etc.) 2) *learning method* (SVM, AdaBoost, hierarchical template matching, etc.). Of course, other decisions must be taken as if the classifier will be component–based, multi–class, etc.
- **Verification/Refinement.** Additional check for the ROIs classified as pedestrians, focused on filtering false positives using criteria not overlapped with the classifier. This module is sometimes also used to perform a fine segmentation of the pedestrian, i.e. detecting the pedestrian contour. Also, removing redundant detections must be done somehow since most of the classifiers are robust to small shifts of the models searched, which is mandatory but implies having multiple detections. With the help of tracking, spurious detections can be also removed or even doing some movement reasoning.
- **Tracking.** Tries to follow the detected pedestrians along time with several purposes as avoiding false spurious detections, predict the next pedestrian position and speed, and infer pedestrian behavior.
- **Application.** Takes high level decisions by making use of the information provided by the previous modules. This processing depends on the level of protection that the system wants to provide: warning to the driver, automatic deployment of airbags, braking, etc.

The sequential processing provided by these modules must cope with the challenges established in Sect. 1.4. We mean that, unfortunately, things are not that simple as to say that one given module will solve one given challenge. Each module contributes up to some degree to solve each challenge, and is the set of all them that must solve the pedestrian detection problem:

- **Cluttered and changing background** plus **very high intra–class variability**. The main module addressing this problem is the **Classification** module, this is, its performance to classify a given ROI as containing a pedestrian or background. Of course, here a key partner is the **Segmentation** module probably helped by the **Preprocessing** one, since they should have the ability of removing most of the background ROIs. As we will see, the discriminative power of a classifier depends on the *performance ratio* given by the *number of false positives* (background classified as pedestrian) and *number of false negatives* (pedestrians classified as background). Unfortunately, the normal situation is that by decreasing the number of false negatives we increase the number of false positives. At the system level this means that to avoid missing pedestrians we must admit a number of false alarms that can make the system totally useless (the driver either would distrust it or even could be distracted up to causing the opposite effect, i.e. an accident). Therefore, if the preprocessing plus segmentation are able to remove most of the background ROIs we can admit a higher number of false positives in our classifier in order to miss less pedestrians. Besides, the **Tracking** module could also remove spurious detections, although it is not easy if we are not able to achieve a high processing rate. Finally, the **Verification** module, when included, has the task of reducing the number of false positives without increasing the number of false negatives.
- **Targets and camera following different unknown movements**. Coping with such challenge in order to benefit from movement analysis is very difficult. In fact, the most widespread approaches for **Preprocessing**, **Segmentation** and **Classification** modules do not use movement. Therefore, *a priori* only the **Tracking** module must take, somehow, such circumstances into account. However, as we will see in this master thesis (Chapt. 4) the uncertainty in such movements could increase the number of ROIs arriving to the **Classification** module, in particular, due to the fact that the horizon line in the images changes from frame to frame. Therefore, a relevant task for the **Preprocessing** and **Segmentation** modules is to try to attenuate this problem, probably with the help a the **Tracking** module if this one is able to maintain a *pitch* estimation.

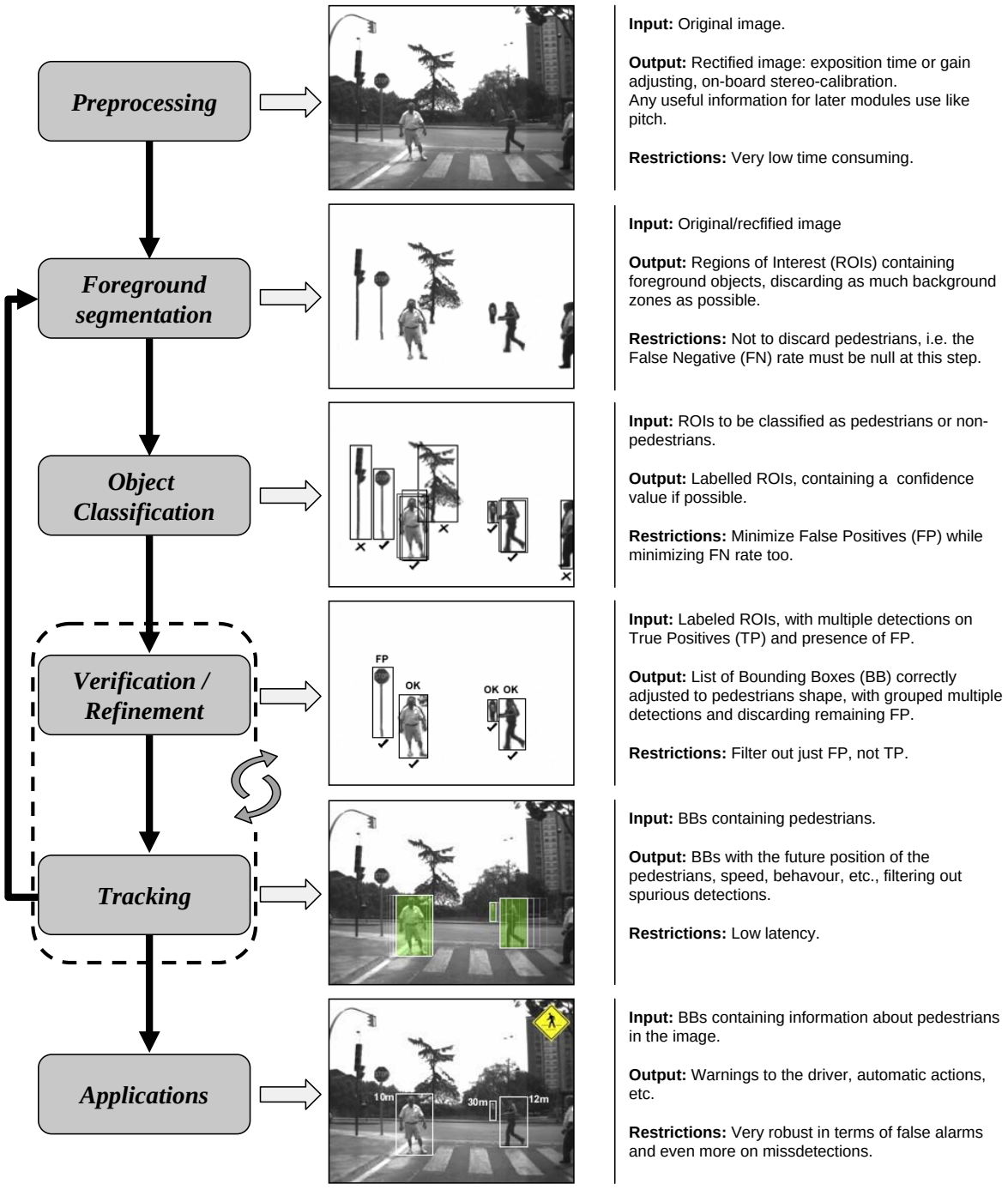


Figure 2.1: Modules proposed for pedestrian detection. Not all these modules are present in the works found in the literature, and some of them can appear grouped by just one algorithm, however, we think this is a correct description. Besides, sometimes verification/refinement either comes after tracking, or they are coupled in a way that makes difficult to divide such processing in two high level modules.

- **Fast system reaction is required (real–time plus low latency)**. Often the **Classification** module is the most time consuming one, due to the high number of features to compute and the high number of ROIs that it usually processes. However, it is difficult to lower the time consumption without affecting the performance ratio. Therefore, again the **Segmentation** and **Preprocessing** modules are crucial: they should provide a number of ROIs that is close to the actual number of pedestrians in the scene, but without consuming themselves too much time. Here, the **Tracking** module can also help since it can provide estimates of where we should expect previously seen pedestrians. On the other hand, if the tracking is used to consolidate detections, it should have low latency. For instance, working at 25 frames per second and tracker latency of 5 frames, if a car runs at 60Km/h, when the system accepts that there is a pedestrian, the car would be about 4m closer to the pedestrian than using a single–fame decision. Of course, whether this is dangerous or not depends on the actual distance to the pedestrian.

- **Very demanding robustness**. This is important at the system level, therefore, it must be the result of the good work of all the modules. From the combination of the **Preprocessing**, **Segmentation**, **Classification** and **Verification** modules provide few false alarms (even one per hour seems too much) and no miss–detections. From the **Tracking** module provide reliable information to the **Application** module as pedestrian position, speed, etc. Achieving the desired robustness also depends on engineering strategies as allowing different performance ratios for different distances, spending more time for near distances, and so on.

Fusion with other sensors can also speed up the system and add robustness. For instance, low–level fusion can be done at the **Segmentation** module since radar/laser scanner sensors (Sect. 1.3) can provide ROIs quite fast, while high–level fusion can be done at the **Tracking** module since those other sensors can also try to perform their own classification (using different information than vision). Information of the own vehicle could also help. For instance, if we know the *pitch* of the vehicle, we can compute the position of the horizon in the image, which can be used to reduce the number of ROIs to explore (assuming a flat road). Of course, the speed and yaw angle of the own vehicle can be used by the tracker in order to reduce the uncertainty of its estimates.

All together gives idea of the difficulty of pedestrian detection since issues as segmentation, classification and tracking are still open problems of the scientific community and full research areas in their own. Let's see now how the different issues have been addressed up to now by reviewing the main contributions to the pedestrian detection problem.

2.1 Preprocessing

Image acquisition

A common preprocessing step in cameras for the automotive industry is done during acquisition. Parameters as exposition time or gain are automatically changed by the camera in order to avoid saturation (e.g. when the camera faces the sun) and dark images (e.g. when entering a tunnel). Therefore, this is a preprocessing done frame by frame.

Binocular Stereo systems

When using binocular stereo (e.g., as we will see later, to perform foreground segmentation) calibration is needed. In theory this is done once before the system starts to operate, i.e. this preprocessing is not frame by frame. However, due to the use of the system in a moving platform, it is likely that

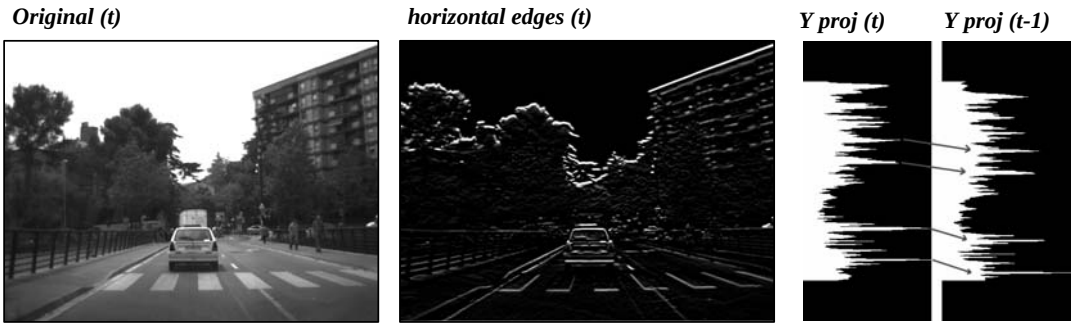


Figure 2.2: Image stabilizer. Horizontal edges are extracted, then the horizontal projections of two frames are correlated in order to estimate the horizon line variation.

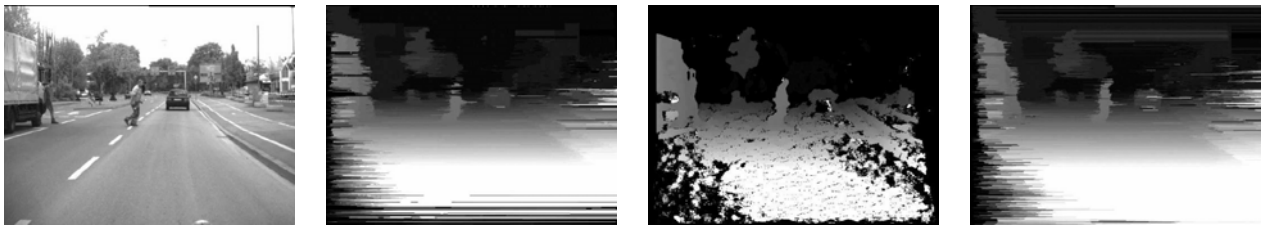
with the time the calibration is not longer correct. For this reason, research is being done in on–board self–calibration, [16, 31] are examples.

Monocular systems

Other preprocessing needed for using a moving platform is *horizon line estimation* (in Chapt. 4 we will address this problem too). When the vehicle accelerates or brakes, the horizon line position changes in the images; the same happens due to uneven road surfaces or different placement of weights inside the vehicle. This is a very important problem for distance estimation using a monocular system. Assuming a flat road, given an horizon line we can compute the distance to the camera that corresponds to each image row. However, typical oscillations in the position of the horizon line can lead to errors of the 100% in the distance computation, making this method useless. This is specially relevant in non–fast roads as urban scenarios, where, in addition, the horizon line can change just because we are facing a sloped road.

Wrong distance estimation is not only a problem for pedestrian detection but also for vehicle detection and others. The main consequence is to increase quite a lot the processing time if we want to obtain reliable results. For instance, if we are not able to estimate the horizon line then a range of possible distances must be considered for each row (this is equivalent to say a number of horizon lines), let's say n_d distances, thus, roughly speaking we can say that the processing time is multiplied by n_d (usually a number from 3 to 20). Notice that using a monocular system we must perform a foreground segmentation consisting in eliminating the 2D ROIs that surpass the PSC of this preprocessing, therefore, the more ROIs the more time consumption of such segmentation.

In [4, 54] image stabilization is used for posterior vehicle detection based on the detection of a global vertical optical flow ([8, 21] present similar ideas for TIR based pedestrian detection). In [67] the stabilization is based on computing straight lane markings. Fig. 2.2 depicts the idea of computing a global vertical optical flow: two successive frames are compared by correlating the projection of the horizontal edges image. If the projections present a clear displacement, an inverse vertical shift is applied to the input image so the current frame can be compared directly with the previous one.



(a) *(b)* *(c)* *(d)*

Figure 2.3: Comparative of some stereo vision algorithms [106]. *(a)* Original image. *(b)* OpenCV, *(c)* *Winner Takes It All Multiple Windows 5*, *(d) Dynamic Programming* results.

2.2 Foreground segmentation

Binocular Stereo

In order to retrieve distance information of the 3D world, binocular stereo algorithms perform a 2D correspondence between two images acquired from seldom cameras looking at the same scene with parallel optical axis but separated by some centimeters (baseline). More details are given in App. D.

The idea of using stereo is to provide 2D ROIs that correspond to 3D vertical objects that fit the PSC (Fig. 2.3).

[42] presents a correspondence analysis based on a local structure classification, assuming that epipolar lines between the two cameras coincide with the image rows, and solving correspondence ambiguity by a disparity histogram. Later on, [40, 41] constructs a multi–resolution method with sub–pixel accuracy based on it. This stereo technique, originally aimed at detecting vehicles, is also exploited in pedestrian detection. In [49], the returned depth map is multiplexed into different discrete depth ranges, which are then scanned with pedestrian–sized windows taking into account the ground plane location with appropriate tolerances. If the depth features in one of the window areas exceed a given percentage, the window is added to the ROI list supplied to the classifier.

[120] uses the algorithm presented in [62] in order to outline possible pedestrian ROIs. Depth thresholding, morphological operations and blob analysis are applied to the disparity map. PSC are also taken into account.

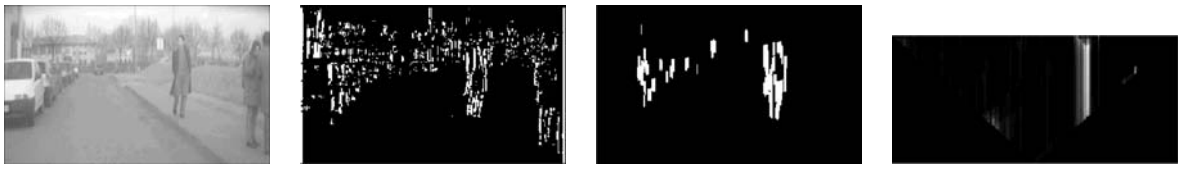
In order to distinguish between ground, background and vertical objects, *Broggi et al.* first used the disparity map [17] and later changed to the v-disparity technique in [18] (App. D). The detected vertical objects are used to construct ROIs for further processing. In [57] it is followed a similar approach.

More recently, in [101] authors also propose disparity segmentation as foreground pedestrian segmentation. Size criteria are used to select the pedestrian candidates.

Rough appearance

Along several works [17, 7, 9, 18, 11], the approach used for segmentation at the *Parma University* (Broggi, Bertozzi, Fascioli and others) consists in first detecting *vertical symmetry* derived from grey level and vertical gradient image values, and then adjusting a ROI coherent with the PSC around each relevant symmetry axis. This second step is needed despite the fact that a scan through a potential vertical symmetry axis is made along a range of distances, evaluating different bounding boxes of a size according to the depth and aspect ratio corresponding to a pedestrian, i.e. coherent with the PSC. Presence of lots of horizontal edges is taken into account as a non–pedestrian quality.

In some cases [17, 18] the symmetry is evaluated after using stereo to select a first set of ROIs, thus



(a) *(b)* *(c)* *(d)*

Figure 2.4: Symmetry detection in [17]. *(a)* Original image. *(b)* Original vertical edges image. *(c)* Resulting image after substracting background vertical edges from original vertical edges by using stereo constraints. *(d)* Symmetry map highlighting zones where pedestrians are likely to appear.

the segmentation combines stereo (3D) and symmetry (2D) information. In figure 2.4, the algorithm extracts the edges by using a Sobel operator, and applies a stereo–preprocessing in order to eliminate objects of the background. Once vertical edges have been extracted, the scanning is made and a vertical–lines map provides information of the vertical symmetries in the scene.

In [100] the authors generate ROIs by filtering out windows with lack of distinctive texture properties and not coherent with the PSC. The authors claim an average of 75 ROIs per image, however, no more details about such distinctive features are given.

2.3 Object classification

Given a list of ROIs, the classification step must label them as (containing) pedestrians or non–pedestrians. All the found approaches are purely 2D.

Silhouette matching

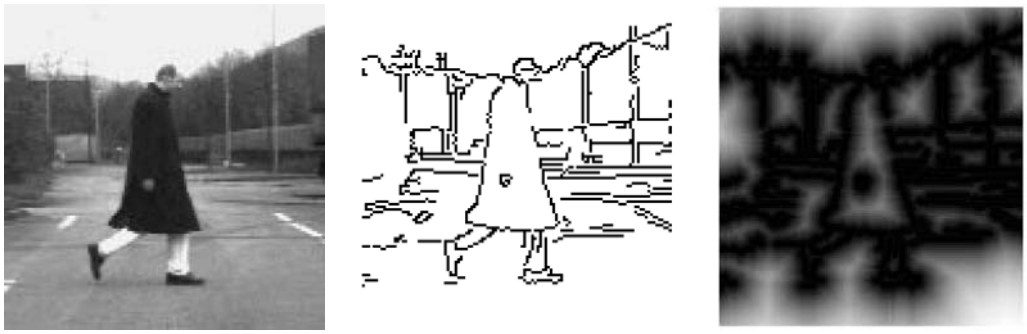
In order to classify as pedestrian a ROI provided by the foreground segmentation (based on stereo and symmetry), [17] proposes to search a model of the upper part of a pedestrian. In particular, a specific binary pattern representing head and shoulders is partially matched by correlation with the edges' modulus map of the ROI. The binary pattern is considered under different sizes. If the matching succeeds, the ROI is classified as pedestrian–like.

Along a series of papers [46, 41, 47, 48, 51, 50, 49], *Gavrila et al.* progressively develop a classification system for the ROIs supplied by their binocular–stereo–based foreground segmentation.

The first step of the classification is the so–called *Chamfer System*, a hierarchical template–based silhouette classifier: a distance transform (DT) of a given ROI is calculated (Fig. 2.5) and a coarse–to–fine template matching by using a hierarchy of pedestrian shapes is performed. The shape hierarchy is generated off–line by a clustering algorithm (Fig. 2.6).

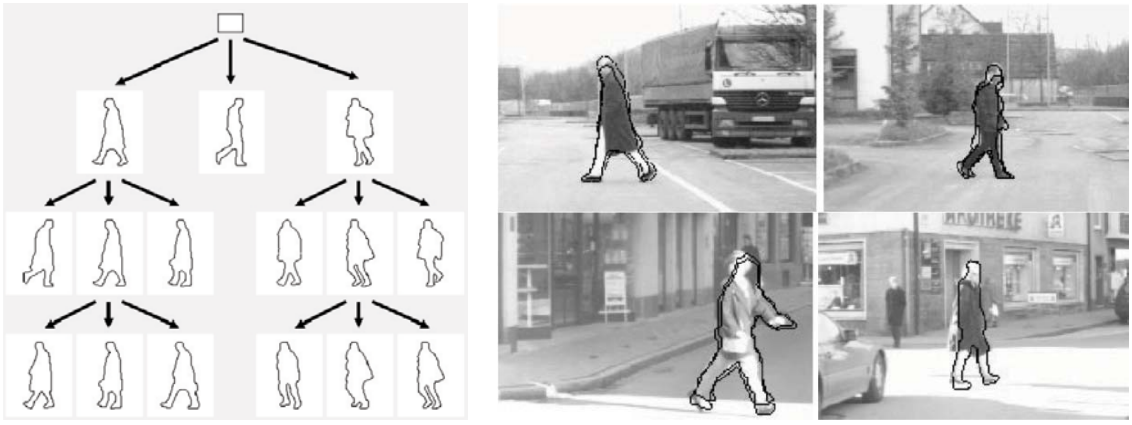
To finally reject or accept the ROIs classified as pedestrians by the Chamfer System, a second step performs a classification based on texture and a neural network with local receptive fields (NNLRF) (radial basis functions, i.e. RBFs), were also tested in preliminary approaches). Up to now no more details are given about such texture–based classification[1].

[1] In the web site of Dr. Gavrila, `www.gavrila.net`, there are announced two new journal papers where more details will be given: 1) *D. M. Gavrila and S. Munder. Multi-Cue Pedestrian Detection and Tracking from a Moving Vehicle. To appear in the International Journal of Computer Vision, 2006*; 2) *S. Munder and D. M. Gavrila. An Experimental Study on Pedestrian Classification. To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006.* An also confusing point here is that the name *NNLRF* does not suggest



(a) *(b)* *(c)*

Figure 2.5: Distance Transform [47]. *(a)* Original image. *(b)* Edge image with non–maximum–suppression. *(c)* Distance transform.



(a) *(b)*

Figure 2.6: The Chamfer System [47]. *(a)* Coarse–to–fine hierarchy of pedestrian templates. *(b)* Detection of human sized shapes that match with the templates in a real environment.

Appearance–based classification

Roughly speaking we can say that the methods included here start by defining a *space of features* capturing appearance and such that, given ROIs containing *meaningful examples* (pedestrians) and *counter–examples* (i.e. non–pedestrians), they use such features to *learn* a classifier. In fact, this approach includes a lot of on–going research in Computer Vision not only for doing pedestrian detection but for object classification in general: *which are the best features?*, *how many do we need?*, *how can we compute them efficiently*, *how do we build a training set (examples/counter–examples) sufficiently representative but not too large?*, *what is the best training method (fast and with good generalization properties)?*, *should we use a component–based approach?*, *should we use a multi–class approach?*, etc.

Since appearance–based classification is such a general problem, here we not only find pedestrian classifiers in the context of PPS, but also methods that try to classify a ROI as people or non–people for other purposes (e.g. for internet searches, or just using the pedestrian class among others to demonstrate the goodness of a classifier under design). Out of the PPS context, however, we find many times both the use of images of better quality and conditions than the typical images managed by a PPS, and testing based only in few examples.

A common detail of this type of approach is to normalize either the size and aspect ratio of the ROIs coming from the foreground segmentation before processing them. In fact, such kind of normalization can either be explicitly done or just *simulated* by the way the ROI features are computed. Another common detail is the fact that the different approaches discard color, which is normal since the variability of clothes probably makes this cue useless. There are cases where the three channels of a color image are treated separately and some rule of the type *the best among the channels* used, we don't consider this as color processing since the color itself does not matter. The different approaches also share the fact of actually learning full–body pedestrians. Finally, a last common detail is that the most widespread learning methodology is single–class[2]. Once this is said, we concentrate the review of appearance–based classification mainly in the kind of features used and the learning method.

In fact, the previously mentioned classification based on texture (feature) and the NNLRF (learning), applied to the ROIs accepted by the Chamfer Matching, follows the appearance–based classification idea. Therefore, the current approach of *Gavrila et al.* uses silhouette and appearance for classification.

In [120] the gradient magnitude of the ROIs segmented by the stereo step is used as feature and a three–layer feed forward neural network (FFNN) as learning method. Thus, somehow the appearance captured corresponds to the silhouette, but without explicitly delineate it. This is not an exception since gradient information (modulus, orientation or both) plays a key role in most of the feature sets.

In [82, 83, 84, 85] *Papageorgiou et al.* from the *CBC and AI Lab of the MIT* base the feature set in the so–called *basic Haar set* of wavelets (Chapt. 5). Such wavelets can be scaled and the authors propose the use of just two scales for efficiency, rejecting very fine and very coarse scales, but following an overcomplete scheme (i.e. the wavelets overlap). As learning method the authors use support vector machines (SVM, App. B) with a quadratic kernel, reducing the initially obtained surface based on about 1000 vectors to another of just 29. The trained pedestrians were mainly on frontal and rear views (which in fact does not agree with statistics on accidents, which state that the worst case consists in a pedestrian crossing perpendicular to the vehicle's trajectory). In [82] the authors proposed an incremental bootstrapping to catch meaningful counter–examples, this is, the non–pedestrian ROIs classified as

---

any temporal *information integration*, however, when this type of neural network is mentioned in [49] and also in the pg. 45 of [74], the authors refer to [117] and [118], respectively. These papers speak about adaptable time delay neural networks (ATDNN) which integrate spatio–temporal information. Thus, at this moment this texture based classifier is quite unclear for us.

[2] In the recent literature we can find some multi–class approaches which is a promising working line, however, they are not yet sufficient mature from the viewpoint of pedestrian detection so we will consider them in the near future but not for this master thesis.

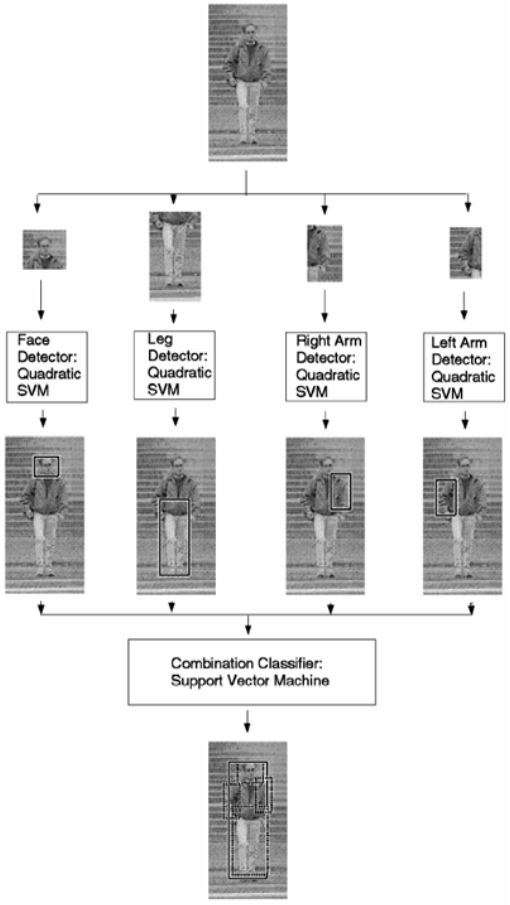


Figure 2.7: Subregions of component–based approach proposed in [79].

pedestrian are used to increment the set of counter–examples and retrain. More recently other authors [57] have followed the same ideas but with some modifications: 1) a classifier is used for frontal/rear views of pedestrians an a different one for side poses; 2) rather than using Haar wavelets a 3×3 Sobel edge detector is used.

Latter [79] the same MIT group proposed a component–based approach. Rather than using a full–pedestrian model (holistic approach) pedestrians are divided in four manually chosen parts (head, legs, right and left arms) and for each part a classifier is trained using the original scheme: Haar wavelets and quadratic SVM. Then a final *combination classifier* is learnt from the four responses of the component–based classifiers (i.e. distance to decision surface), where this last learning is based on a linear SVM (Fig. 2.7).

More recently, this idea of a 2–stage component–based classifier has been also followed by *Shashua et al.* [100], although with modifications to adapt the method to more realistic conditions regarding the images than the assumed in [79]:

- A ROI coming from the foreground segmentation is divided into nine manually chosen subregions that overlap (1 to 9 in Fig. 2.8). For each subregion a biologically inspired descriptor is computed. In particular, it is the same than the descriptor of a *keypoint* in the context of the

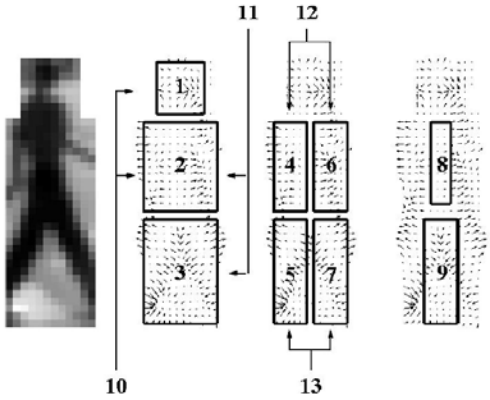


Figure 2.8: Subregions for a candidate window [100]. Gradient image is divided into nine subregions, and four combinations are also taken into account. The distribution of the arrows (gradient strength) is passed to the classifier.

scale invariant feature transform (SIFT) proposed in [71]. The proposed set up in [100] for such descriptor consists in the division of the subregions in 2×2 cells, having each cell an associated gradient orientation histogram of 8 bins. The smoothed gradient magnitude is used to weight the contribution of its corresponding orientation. This means that each subregion is characterized by a descriptor of 2×2×8=32 entries (the descriptor is normalized to gain robustness regarding illumination changes).

- The training set is divided into nine mutually exclusive training clusters where each cluster represents a training collection from a particular pose and illumination conditions. This is a manual process that tries to reduce the variability of the pedestrian class.
- For each subregion and each cluster it is trained a different classifier using the subregion descriptor detailed before. The training is based on the so–called *Ridge regression*. We can say either that each subregion is defined by nine classifiers (one per cluster), or that one cluster is characterized by nine classifiers (one per subregion). At this point the authors consider convenient to add four new subregion defined by overlapping some pairs of the previous ones (10 to 13 in Fig. 2.8), hence, characterized by a descriptor of 64 entries. Therefore, now each cluster is defined by thirteen classifiers (one per subregion). Since we have nine clusters, we have 9×13=117 classifiers, i.e. given a ROI we can reduce it to 117 numbers (each number gives the resemblance of the ROI with a pedestrian, looking only in a subregion with the *eyes* of the corresponding classifier). This new descriptor of 117 entries is the analogous to the descriptor of 4 entries coming from the classification based on Haar plus quadratic SVM used in [79].
- Each component of the descriptor of 117 entries is seen as a *weak classifier* and an AdaBoost learning (Chapt. A) as *combination classifier*. This is the analogous to the linear SVM used in [79] as combination classifier.

Recently, in [30], *Dalal and Triggs* present a pedestrian detection algorithm that, in terms of performance, compares very favorably to some of the most well–known previous existing ones. For a given ROI their method uses as features a dense grid of histograms of oriented gradients (HOG, Chapt. 5). In fact, such features are similar to the descriptor that is associated to each keypoint in the context

of the SIFT, but a new ingredient is added in the HOG: the *block* concept, which is a collection of a fixed number of adjacent cells. Now a ROI is divided in squared cells and, for each cell, the histogram of gradient orientations is computed, as with SIFT. However, now, blocks are formed as a collection of contiguous cells (forming a kind of squared sub–ROI) and the histogram of each cell is normalized according to the whole block. Different blocks are obtained by scanning the ROI in a sliding fashion with a fixed step. Thus, blocks overlap and, therefore, a given cell can belong to more than one block. This implies that each cell contributes with more than one normalized histogram of orientations to the final descriptor of the ROI (one per block to which the cell belongs). It seems that the method is more robust to illumination changes than other methods. The learning to obtain a classifier is based on a linear SVM. The different parameters of the descriptor must be set for each database in order to obtain the best performance. The authors use pictures taken from photo–cameras, this is, of better quality than the usual images we obtain from an automotive camera in movement. With the settings for their database, the HOG descriptor of a ROI has $3,780$ entries, i.e. the vectors managed by the linear SVM are of $3,780$ components.

Latter, *Zhu et al.* [121] introduce variation in size, location and aspect ratio of the blocks. They propose a larger set of blocks than in the original work [30], in particular, $5,031$ blocks versus 105 for a given ROI. The classification is based on a training process similar to the one introduced in [112], i.e. a rejection–based cascade of classifiers constructed with AdaBoost (Chapt. A). In particular, a feature corresponds to the description of a block, i.e. the normalized histograms of its cells. With their settings this implies that a block is represented by a vector of 36 entries, and such vectors are used to train a *weak classifier* based on a linear SVM. Then AdaBoost is used to train the cascade of classifiers from the weak classifiers. In this process, since there are 5,301 blocks and so weak classifiers, a method to sample only 250 blocks from the whole set is used. That sampling method [99] (pg. 180) guarantee nearly as good performance as if we considered all the random variables. The final cascade has 30 levels. The reported results are similar to those in [30] but being the new method about 20 times faster. To this time reduction also contributes the use of the *integral image* idea the authors apply to the orientation bins, as in fact was previously proposed in [65] in the context of face detection.

Besides, the method in [112] by *Viola and Jones* for face detection was adapted for pedestrian detection to include it in the comparison by *Zhu et al.*. The result is a rejection–based cascade of 18 stages and about 800 weak classifiers based on the simple set of Haar wavelets (Chapt. 5) with different sizes and positions (in fact, Haar wavelets and AdaBoost were already used for pedestrian detection in [111] but with images taken from a static camera, being the wavelets spatio–temporal). The resulting classifier from the Haar wavelets and AdaBoost approach included in the comparison is twice faster than the HOG plus AdaBoost, however the accuracy of the former is claimed as not as good as the latter.

We also want to mention in the review a recent work [107] based on *convolutional neural network* (CNN), a special variant of multilayer perceptrons (MLP) in which the first layers are configured to act as a hierarchical feature extractor. According to the authors this type of network requires about 40 times less computation than the SVM while providing a superior accuracy, thus, for us is an option to explore in the near future.

Finally, in another recent work [101], after foreground segmentation based on disparity, authors propose a classifier based on the so–called *four directional features* (FDF) and the SVM with a Gaussian Kernel. The four directional features consist of four images that are divided by edge into horizontal, vertical, diagonal right top and diagonal left top, reduced to low resolution through the Gaussian filter. These features, however, are less rich than a set of Haar wavelets at different scales.

2.4 Verification/Refinement

Given a ROI classified as containing a pedestrian, it is interesting to compute the bounding–box/contour of such pedestrian. This would help tracking (or any analysis of time coherence) to actually use only properties of the pedestrian. Besides, precise location of the pedestrian can help to know his/her distance to the vehicle and ultimately the time–to–collision, Merging redundant detections can also help to perform extra verifications to remove some remaining false positives.

In some systems, part of the work required for this module has been already done in previous ones. For instance, in [49, 74] a stereo–like verification step is proposed. For any ROI classified as pedestrian a silhouette is available thanks to the Chamfer System used during the classification. This silhouette is used to isolate the image region corresponding to the pedestrian in the left image of the stereo pair. This region is then used for doing a cross–correlation in the right image of the stereo pair within a certain disparity search range. At this point, the original ROI is re–classified as non–pedestrian, i.e. rejected, if the resulting cross–correlation function has not the expected shape (height and spread). Notice, for instance, that the shape of such cross–correlation function generated by a mainly homogeneous area (e.g. asphalt) must be different from one generated by a more textured area (e.g. pedestrian). Figure 2.9 illustrates de idea.

In a previous work [40] the authors also suggested the analysis of the gait pattern for pedestrians crossing perpendicular to the camera. Notice that such analysis can not work for pedestrians walking parallel to the camera optical axis, either frontal or rear viewed, however, if the camera is forward–facing, then it is the same than saying *crossing perpendicular to the vehicle trajectory*, and this turns out to be one of the most relevant scenarios regarding pedestrian accidents [44]. Notice also that such type of verification, in fact, comes after tracking. However, the last papers from that group do not mention gait pattern analysis anymore. Besides, in [40] it was not clarified how the tracking was done.

Broggi, Bertozzi et al. [17] use the head–and–shoulders silhouette matched during classification as reference to refine the detection of the pedestrian feet, i.e. from the shoulder down, the feet are searched in the vertical edge map that was previously created for symmetry detection during the segmentation phase. Once this process is done, an initial pedestrian bounding–box is available. Since this process is done using the left image of the stereo pair that supports their first step of segmentation, now the pedestrian bounding–box defines a pattern that is searched in the right image of that stereo pair. This search is limited by a rough estimation coming from the segmentation modulus. Once the pattern is found a more accurate distance to the pedestrian is computed and this is used in turn to refine the localization of the feet (thinking in a pedestrian as a vertical plane). Since the new bounding–box is supposed to be more accurate, criteria as the PSC and grey level homogeneity (based on entropy) are applied as final verification regarding the classification of pedestrian/non–pedestrian.

The same authors used a similar procedure in [8, 11], however, since in these works the silhouette method was not used, the initial pedestrian bounding–box is fully computed by reasoning about symmetry and density of vertical edges. After it comes the same stereo refinement and then yet another difference is that the final verification (extra rejection) is carried out by *a number of filters based on regionalization* (no more details are given since they claim them to be under development). In a different work [64], that final verification is based on the configuration of 3D vertical curves inside the bounding–boxes provided by the symmetry analysis. A new pedestrian bounding–box is computed but this time defined in the $X - Z$ plane of the 3D world. Since this is a kind of *bird–view* of the pedestrian the bounding–box is, in fact, a circle. Clearly, this provides the 3D pedestrian location. Notice that in these works [8, 11, 64] stereo is not used during the segmentation phase, hence, the stereo reasoning described above implies to add stereo to the system only due to this verification/refinement module.

In [7] these authors also tried another approach where the initial pedestrian bounding–box was computed similarly than in [8, 11], however, in this case rather than using stereo the alternative under test was based on autonomous agents acting as edge trackers (Fig. 2.10). Basically, they try to find the

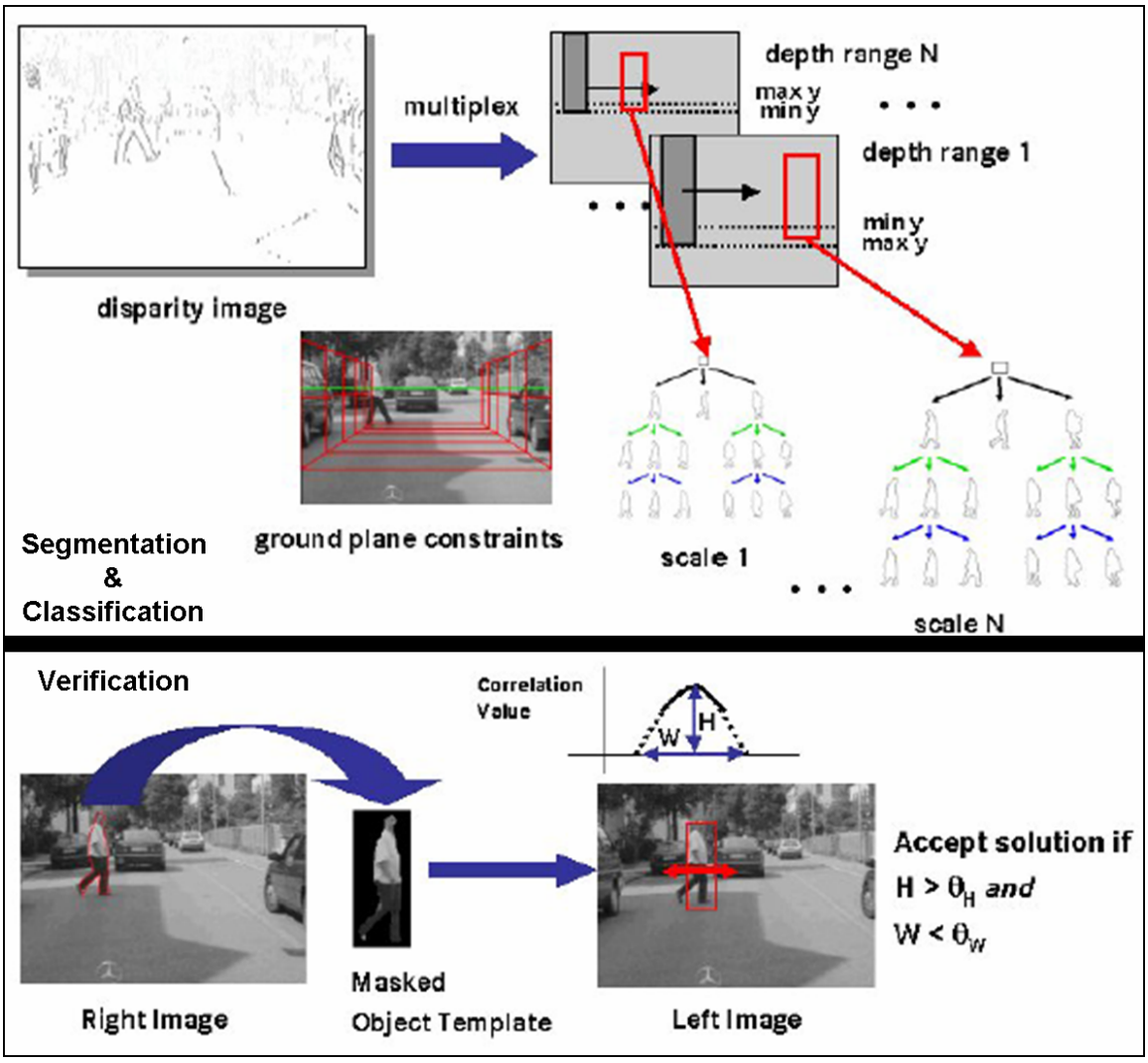


Figure 2.9: Images taken from [74] (pags 37 and 38). The verification step is based on the shape of a cross–correlation function between a supposed pedestrian region in the left image of the stereo pair and the corresponding region of the right image, within given disparity limits.

silhouette of a pedestrian inside the bounding–box to take a classification[3] decision.

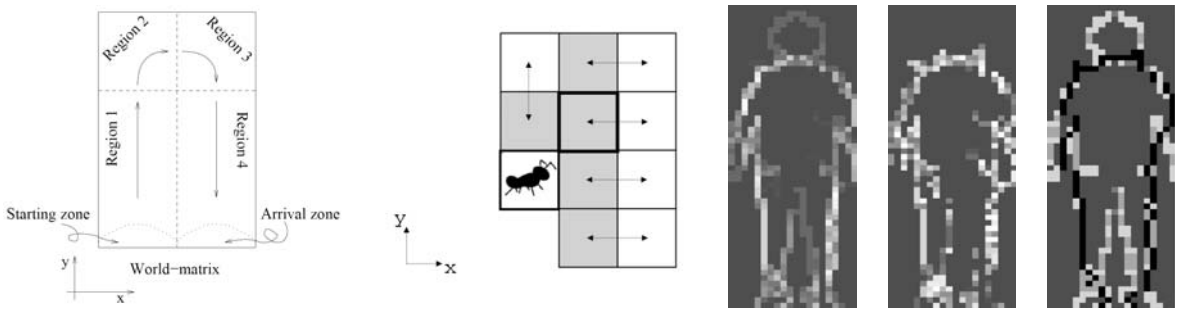


Figure 2.10: Picture from [7]. The algorithm, inspired by the *Ant Colony Optimization* metaheuristic, uses *artificial ants* (the agents) that follow the edges of the image, leaving a *pheromone* (weights) path to guide other *ants*. *(a)* Artificial ants move through the world–matrix regions 1,2,3 and 4. *(b)* Possible movements of the ant (from current position to the gray cells) when moving through region 2. *(c)* Map of the world–matrix. *(d)* Map of the pheromone trails deposed by ants after two cycles. *(e)* Example of the result obtained by the best purely stochastic and after two cycles.

In [100] it is proposed a *multi–frame approval process*. Basically, the ROIs classified as pedestrians still must to surpass a validation process that collects information from several frames: gait pattern, inward motion analysis based on ego–motion from the preprocessing modulus, consistence measure of the single–frame classifier, tracking quality measures, etc. The approval is based on a decision–tree type of classifier that also comes from training. The authors point out also that in this decision tree a key role in played by the classification scores of background subclasses as vehicles, poles, guard–rails, repetitive texture, etc. The distance to the pedestrians is also computed by locating their feet. Here verification comes after tracking too.

In fact, it is clear that systems incorporating tracking are like to include some verification after tracking to remove detections that are incoherent or of low confidence over time.

2.5 Tracking

In [84] it is suggested an heuristic integration through time assuming proximity of correct pedestrian detections in consecutive frames and that false positives rarely persist (thus, they can be removed). But not more details are given.

In [41] the distance, speed and acceleration of the detected objects are controlled by two Kalman filters [115], one for the lateral motion and the other for the longitudinal one. For the lateral motion the yaw rate of the own vehicle is used (it is not clarified if it is somehow estimated or just read as information provided by the car, we understand the latter). As we mentioned before, the ground plane is also estimated with tracking help. Posteriorly, in [49] authors from the same research group use an

[3]Yes, *classification*. But isn't this the verification/refinement module?. Well, regarding the works at Parma University, we must say that we have fitted their steps in our architecture according to the main papers. This means to include bounding box refinement in the verification/refinement module. But from the view point of [7] alone, such refinement plus the silhouette search based on ants are probably more like to belong to the classification module. After all here we offer our interpretation, but it is not unique.

α-β tracker (a simplified Kalman filter with pre–estimated steady–state gains and a constant velocity model) based on the bounding box representation coming from their stereo verification phase. A more sophisticated work regarding tracking (done by some of the same authors) can be found in [55], where the authors use three independent visual cues (silhouette, texture, stereo) which are modelled as mixture of Gaussians in a Bayesian framework that performs tracking of multiple objects directly in 3D space. The optimal Bayesian tracking is approximated by particle filters.

[86] uses a variant of the particle–based filter known as *Condensation* [59] in order to track silhouettes. These silhouettes can be the generated during the previously seen classification based on Chamfer System [46, 52, 47, 48] but approximated by B-Splines for tracking purposes.

In [8, 11] a Kalman–filter–based tracker is also proposed with the aim of rejecting spurious detections as well as computing the trajectory of the pedestrian. It is mentioned that the necessary *merge function* (that address the *association problem*) needed for associating newly detected pedestrians with existing tracks is based on overlapping between detections and on Mahalanobis distance. The test are done, however, in not too complex indoor scenarios.

In [57] Kalman filters are used to maintain pedestrian estimates and Bayesian probability to provide an estimate of pedestrian classification certainty over time and a pedestrians' trajectory and speed. Nothing is said about the *merge function*, only that by doing such merge spurious detections are removed (which is a kind of verification after tracking since in this case the pedestrian detections come directly from the classification module).

In [64] the circular bounding–boxes of pedestrians are tracked. In particular, the bounding–boxes in one frame are searched in the next frame by a nearest neighbor approach that computes the optimum matching between all the present circles. As features for the matching, 2D and 3D characteristics are used, e.g. mean grey–level, mean disparity, etc.

In [101] Extended Kalman Filtering (EKF) is used to track the pedestrians, also assessing their classification certainty over time for rejection (i.e. verification) purposes.

2.6 Application

At the application level it is necessary to know pedestrian trajectory and speed in order to evaluate the risk of collision and so taking proper actions.

This kind of analysis do not usually appear in the reviewed literature, however [74] is an exception. It describes how in the context of the SAVE-U project there were implemented two types of actions at the application level: acoustical driver warning and automatic braking. These protection measures follow a strategy based on three phases:

- *Phase 1: Early Detection.* The system detects and tracks all pedestrians in front of the vehicle (within the sensor coverage area), but none of the protection measures are activated yet.
- *Phase 2: Acoustical Driver Warning.* A pedestrian is detected to enter the vehicle's path, but there is no risk of an immediate collision yet. The driver is alerted by an acoustical signal about this potentially dangerous situation.
- *Phase 3: Automatic Braking.* A high risk of a collision has been identified. The vehicle is decelerated in order to avert the collision or, in case a collision is unavoidable, mitigate the impact.

Given both the difficulty of estimating the speed of the pedestrians and their unpredictable behavior (e.g. suddenly start or stop braking, even change direction) the system takes decisions only based on pedestrian position and direction, but not speed. For that purpose three areas of risk are proposed (Fig. 2.11).

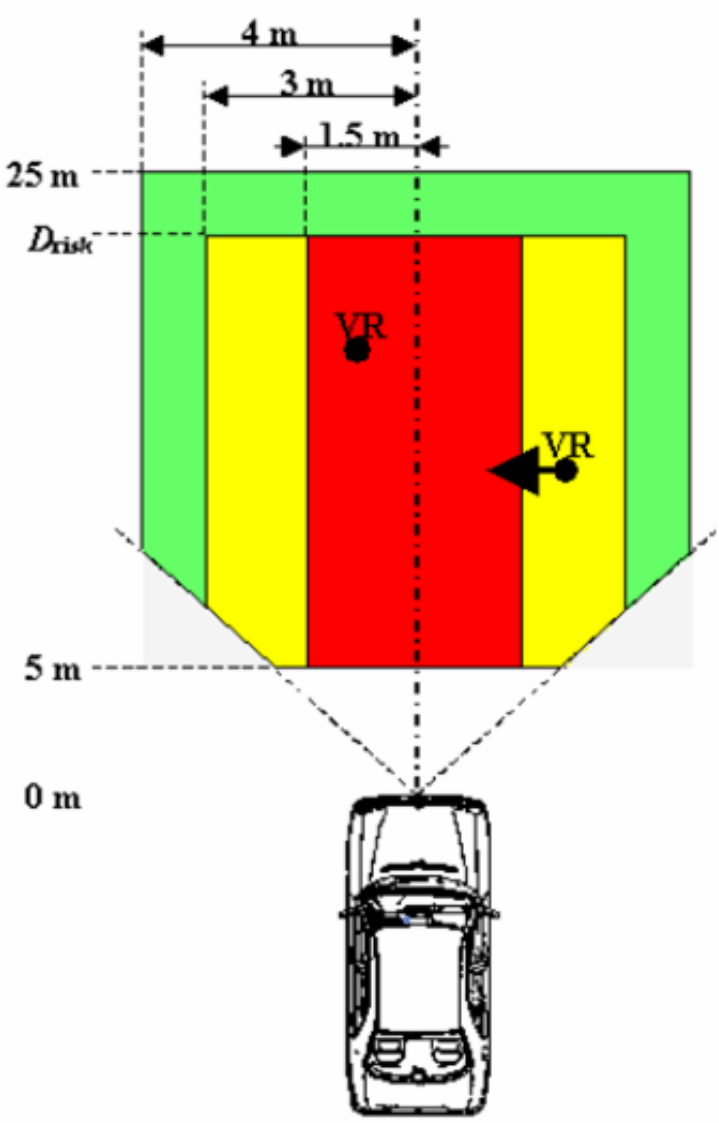


Figure 2.11: Areas of risk. Red: risk of collision (Phase 3). Yellow: the pedestrians are going to enter the vehicles' path (Phase 2). Green: no protection measures are needed (Phase 1). The distant boundary D_{risk} is set to 25m for the acoustical driver warning, and dynamically adapted to the vehicle speed within the range 5m to 20m for automatic braking. Picture taken from [74]. Notice that the red area matches the fact that most pedestrians accidents were when the pedestrian is crossing parallel to the vehicles' path [44].

2.7 Discussion

Preprocessing

Regarding real–time adjustments of the cameras (exposure time, gain, etc.), our impression is that there is already much work to do since these adjustments tend to be global. For instance, inside large tunnels the adjustments are correct, driving with an homogeneous sky brightness too. However, the sensors that we have tested up to now are not able to provide adjustments for situations where there are bright and dark areas at the same time, e.g. a short tunnel (not a bridge) where you see the darkness of the tunnel and the brightness of the end of the tunnel. Besides, transitions of adjustments from a dark situation to a bright one (e.g. going out a long tunnel during a sunny day), or the opposite, tend to take a few frames. We believe, however, that cameras for the automotive industry will finally solve these problems because it seems to be possible (e.g. it is possible to set individual adjustments for each pixel of a CMOS sensor, something not yet exploited).

With respect to stereo calibration, we think that on–board self–calibration is a research line quite important due to the benefits stereo can provide (e.g. for foreground segmentation). Besides, it seems possible to engineer a binocular stereo system for a vehicle, so that it would be sufficiently robust as to need such on–board self–calibration only from time to time, i.e. as the wheel revision, oil revision, and so on.

Referring the real–time computation of the horizon line using a monocular system, we must say that our experience reveals that this is a quite difficult issue. Horizontal–projection–like stabilizers are based on a strong assumption: changes in the scene are smooth, but many times this is not longer true. For instance, passing under a bridge in a highway produces high peaks in horizontal edges projection, so correlation pulls the stabilization results to incoherent values. Urban scenarios are even worse. Shadows change and new objects appear suddenly, resulting in new edges with new artifacts in the new frame projection, making difficult to obtain the proper result from the correlation. Other stabilizers depending in road features, lane markings for instance, will not work if those marks are not present, thus, the stabilization is limited to specific scenarios as highways (where pedestrian detection does not make too much sense).

Open issues on that are both the possibility of making the compensation at the camera level (for instance, correlating images at higher frame rates that the output of the camera) and the use of sensors different from vision to detect camera movements, such as inertial systems. On the other hand, the ego–motion estimation from a single image proposed in [102] is claimed as quite robust by the authors. We haven't tested it since here we will propose a technique based on binocular stereo.

Foreground segmentation

Stereo based segmentation has several advantages: 1) robustness to illumination changes; 2) the provided distances are not only useful to determine relevant ROIs, but distance to a detected pedestrian is a key information in is own, useful for tracking and also at the application level, e.g. to compute time–to–collision; 3) stereo can also be used as information to segment other relevant objects as vehicles, i.e. stereo can be an information shared by different ADAS applications; 4) the typical range of depths were distance computation is reliable while having a useful HFOV, fits the one useful for PPS (i.e. up to about 70m).

However, there are also several drawbacks: 1) since the images can contain uniform areas with no clear texture, the matching is not assured, so "blind" zones can appear. This disadvantage usually appears when dealing with images taken in the evening, where big shadows tend to be black with no gray tones, and very often when pedestrian clothes have similar color than road asphalt or background objects; 2) it is mandatory a fast stereo computation (e.g. below 50ms), which is not easy since we

are facing a time consuming operation. In [106], a study of different fast dense vision algorithms using SIMD[4] technology is presented (Fig. 2.3) and a more recent study can be found in [109]; 3) As [120] states, stereo algorithms manage occlusion well, however, when there are objects partially occluding a pedestrian, the resulting vertical regions can have a range of depths as well as an aspect ratio that may not fit that of a pedestrian. Therefore, there is risk of missing the pedestrian.

Despite of these problems, stereo is still a very promising technique. Its distance estimation is not beaten by any other computer–vision–based method. Moreover, at least, we can perform free–space analysis [41], i.e. avoiding to process regions of low height as the road surface, etc. Such criteria does not remove from the posterior analysis cars, traffic posts, etc., but also conserves occluded pedestrians. In fact, a result that is not usually provided by authors using stereo is how many ROIs per frame are output in average along an urban scenario. [101] is an exception since they claim about 5 to 20 ROIs per frame depending on the type of road (urban roads are the worst case, of course).

Regarding the texture based method commented in [100] we think that it is quite promising if actually only gives 75 ROIs per frame. However, since the details of how it works are not provided we cannot give comments about.

Referring the use of vertical symmetry, the researchers at *Parma University* do not provide statistics about the number of ROIs that such criterium generates. We guess that too much irrelevant ones, since the same authors have combined such information with stereo techniques at the segmentation level to improve their system. Besides, in order to cope with the variability of pedestrians sizes and the uncertainty of their distance to the camera, the computation of such vertical symmetry includes testing for different window sizes around each potential vertical axis. Therefore, regarding processing time, somehow symmetry computation introduces in the foreground segmentation module the part of the problem of a classification module that has not the help of a previous foreground segmentation, i.e. having a large number of ROIs/windows to test. On the other hand, authors reduce the number of symmetry axis to test by doing it only where there is a relevant concentration of vertical edges. Summarizing, we think that symmetry after stereo may be useful, but not alone.

Object classification

It seems clear that silhouette matching methods are not applicable in a stand–alone fashion, even the very elaborated method of the Chamfer System needs an extra step that follows the appearance–based classification idea. We may guess possible problems of the Chamfer System when pedestrian are close to the camera as a result of the higher variability of the pedestrian silhouette, and, therefore problems due to too much false positives in textured background. The extra appearance–based classification helps to reduce these problems.

Taken this into account and the plethora of methods that we have reviewed following appearance–based classification, it seems clear that this is a promising line of research. Of course, the good generalization properties of learning methods as SVM and AdaBoost are contributing to the reliability of appearance–based classification. The use of fast computable and meaningful features (Haar, HOG, etc.) are contributing as well. However, a following simple example can show us that still there is much work to do.

Let's assume that an image arrives to the classifier and that we only use PSC to reduce the number of windows. Given the size of the images we work with in this master thesis, as well as the detection range and pedestrian sizes and aspect ratios we consider, the PSC reduces the number of ROIs to evaluate to *only* about $10,000$ ROIs per image (Chapt. 4) which, in fact, is a hard reduction compared to brute force that would provide about 1 million ROIs after removing the sky area from the analysis. Now, let's assume the best pair false positives/false negative reported in classification using images taken from an

[4]SIMD: Single Instruction Multiple Data

automotive camera in movement. This corresponds to the component–based system in [100], where for a detection ratio of the 95% we have a false positive ratio of about the 0.1%. This means that if we want to detect the 95% of the pedestrians then, in the worst case we can expect about 1,000 false alarms per image. If the system works a 25 frames per second, then, this translates to 25,000 false alarms per second, which is useless for a PPS. Of course, thanks to additional factors as foreground segmentation, detection clustering, time coherence assessment, etc., this number can be drastically reduced at the system level, for instance, imaging we only have 75 windows to inspect after foreground segmentation (number claimed in [100]) the false positives per second goes down to 187.5. Now if they are spurious and do not maintain coherence in time the number can be reduced to only a few false positives per second. Besides, usually it is not accomplished the worst number of 0.1% regarding false positives. Anyway, even 1 false positive per second is too much since this means 60 per minute, and so on, still useless for PPS. As a matter of fact this is one of the conclusions of the ambitious European Project called SAVE-U [74] (pag. 77).

From this example we see the importance of improving all the system modules. Regarding classification we must improve the ratio false positives versus false negatives. However, we must be realistic and expect perhaps a single order of magnitude improvement (e.g. convert the 0.1% of the example into 0.01% keeping the 95% of detection success) in the next years. It does not seem too much but everything does account, after all it is not the same 25,000 false alarms per second at the classification level than *just* 2,500. Here, an important issue is to achieve similar results with faster methods. In particular, if not only pedestrians must be detected but also cars, for instance, then it is important to share features (another open research line of Computer Vision).

A last comment regarding classification is a complain. When comparing new approaches with previous ones, an extra difficulty is that there are not serious databases of pedestrians available as public domain. *Serious* means thousands of meaningful examples and counter–examples taken from a moving vehicle with an automotive camera at different environments, weather conditions, moments of the day, etc. Of course, there are some available free databases where pedestrians are a type of object among others, however these are *toy* databases since they have few examples, not too hard counter–examples, all of good quality, taken in very good conditions, and so on.

Verification/Refinement

Regarding verification and refinement done to help the own verification as well as to provide new data (e.g. pedestrian distance), we must say that as long as the classification has been done based on a 2D image (which is the case of the reviewed works) it makes sense to use range information coming from binocular stereo. However, it us unclear for us why some works only use stereo information at this stage but not during the segmentation (e.g. [9, 11]), we think that if a stereo system is available then it must be of help during the foreground segmentation.

When the work of this module (or part of it) is actually done after tracking then time coherence may be used to discard spurious detections (i.e. false pedestrians) besides, if a pedestrian is well tracked we can see it as static and, therefore, movement–based reasoning can be used (e.g. gait pattern analysis as [58, 117, 116, 118], [40] and [26, 28, 27] or others [75]). Notice that using movement to do foreground segmentation and object classification is difficult due to the ego–motion, background clutter, etc., but the situation here is that we detect a potential pedestrian by other means, then it is tracked and finally the time–coherence reasoning is done.

Another work for this module is the fusion of multiple detections from the same pedestrian. Trying to do so directly from the output of the classification module is not easy, simple heuristics do not work. It seems easier to try such fusion after the refinements and verifications explained before.

Tracking

Tracking can help to speed up the search of expected pedestrians, and also helps to reject false positives for being spurious or because if a potential pedestrian is tracked, then motion reasoning can be applied: the tracker itself can include reliable pedestrian dynamics that can be useful to take decisions at system level (e.g. pedestrian trajectory), it can be checked the gait pattern for potential crossing pedestrians, etc. All together means that it can be a very relevant help. In fact, once pedestrians are detected particular models can be build for their tracking (e.g. using color, which is not taken into account for classification, and, in fact, up to now neither too much for tracking pedestrians in PPS). Furthermore, tracking is essential to have pedestrian trajectories which is needed to take decisions at the application level (warn, brake, etc.).

Up to now, however, in the context of PPS the tracking module has not received as much attention as others like foreground segmentation or object classification. Each paper has its own proposal and no comparison with others is usually provided. Perhaps, here we can look to the surveillance context where tracking of people has more tradition. Probably because in surveillance the foreground segmentation and object classification is less hard (but not easy) than in PPS, thanks to work with a static camera. A review on *Multiple Object Tracking Under Occlusion* that uses people as examples can be found in [45]. It is true that *a priori* the dynamics to model in the context of PPS are also more complex than in surveillance, due to ego–motion, but on the other hand, vehicle information as speed and yaw rate can be read from the own car (CANBus) in order to reduce the additional uncertainty introduced by ego–motion.

Application

Information as the summarized in Fig. 2.11 is very useful to design the whole sensing process. For instance, different trade–offs between speed and risk of miss–detection due to erroneous foreground segmentation can be set for the different areas, different classifiers can also be used for different areas, etc.

Regarding the proposal in [74] it is clear that an extra measure in the worst case is to deploy external airbags. Another functionality for the the worst case consists in taking an evasive action, i.e. if the environment around the pedestrian is sensed too, then steering actions to avoid any impact could be possible. We think, however, this is too risky with the current state–of–the art technology. Other idea could be the use of an automatic hoot to warn the pedestrian.

Anyway, the application level is not trivial at all. Legal issues must be clarified and psychological aspects must be taken into account. For instance, too much alarms could make the driver to distrust the system and then do not believe actual ones. On the contrary, too much conform could produce in the driver an abusive sensation of security so that he/she drives too relaxed, without paying sufficient attention to the driving activities (we may think, for instance, that since the vehicles are more easy to drive, the driving speed has increased, and this is one of the main factors for accidents involving vehicles).

2.8 Summary

Although the most important techniques for pedestrian detection were presented in their corresponding module of this chapter, we include a table that summarizes the techniques used in each of the relevant systems of the field (Table 2.1). We have named them according to their main authors, and the information has been extracted from one or several several papers. The complete pedestrian detection systems are *Gavrila et al.* at DaimlerChrysler [49, 52], *Bertozzi, Broggi et al.* [9, 11], *Broggi, Bertozzi et al.* [17, 18], *Shashua et al.* at MobilEye [100], *Grubb et al.* [57], *Zhao et al.* [120] and *Soga et al.* [101].

In addition, four pedestrian classifiers are also summarized: *Zhu et al.* [121], *Papageorgiou, Poggio et al.* [83, 79], *Dalal et al.* [30] and *Szarvas at al.* [107].

Finally, in Table 2.2 we state the performance rates and main parameters for the classification module of each of the previous systems. In this case, filling the table is not so easy as the modules one because in many of the papers the implementation details are not present.

As it has been mentioned, many papers tend to not give implementation details about the setup parameters or the learning databases. This matter, together with the fact that there are not standard pedestrian databases available, makes difficult to perform enough reliable comparisons between classifiers, thus between detection systems in general. For example, some papers give classification results in False Positives Per Window (FPPW) rates (i.e. the testing examples consist in a set of sampled windows scanned along the input image), without providing the scanning parameters or window sizes (more discussion about this issue in Chapt. 5). We have named this testing methodology as *SCAN* in Table 2.2. Other systems like *Gavrila et al.* or *Zhao et al.* provide classification results by using the foreground segmentation module ROIs as a testing set (*FG Segment. ROIs*). Finally, in other systems, it is not applicable (N/A) to apply such a performance measure, e.g. systems exploiting symmetry.

Next, we expose some further considerations exposing issues not commented before in this chapter:

- Intelligent vehicles prototypes usually have information buses that can provide the detection system with interesting data, like current speed or steering direction. This data can enrich the high-level decisions of the system (application and even tracking modules). In the reviewed bibliography, this information is just exploited in the PROTECTOR system [49] by making use of the CANBus.
- Current classifiers generally work under the assumption of receiving a complete pedestrian image without occlusions or at least with not severe occlusions. Although many databases already include pedestrians carrying, for instance, bags or umbrellas, pedestrians with occlusion of important parts might not be detected. In this sense, component–based classifiers [79] can potentially solve this problem, so this seems an interesting line of investigation.
- There is not much information about error tolerance in the localization of pedestrians. [49] states 5% and 15% tolerance in x and z axes respectively, i.e. at a $20m$ far pedestrian the localization error is set to ± 1 and $\pm 3m$ in lateral and longitudinal position. This tolerance rates are used for two components independently: the application–specific (localization tolerance for the application) and the measurement–specific (true location of pedestrians). Hence, for field tests using the complete system, tolerances of $X = 10\%$ and $Z = 30\%$ are used.

After the exhaustive review of the literature, the summarized tables of some relevant systems, the discussion, and the further considerations exposed above, the reader can realize the big interest and difficulties in pedestrian detection, specially under the conditions proposed in the objectives section in Chapt. 1. Hence, although in this master thesis we focus on two modules rather than in a whole system, it is clear that there is still big room for future improvement in all of them.

Table 2.1: Comparison of the most relevant systems in pedestrian detection literature, highlighting their main parameters.

	Foreground Segmentation	Object Classification	Verification/ Refinement	Tracking	Other Comments
Gavrila et al. at Daimler Chrysler	Stereo+PSC	Silhouette + Chamfer System Texture + NNLRF	Stereo, gait pattern also tested	Kalman; particle filters; α-β tracker. Silhouette, Texture, Stereo, CAN data, isolated or together	Road surface estimation from stereo
Bertozzi, Broggi et al.	Symmetry+PSC	*(here another possibility was to think of the PSC as segmentation and symmetry as classification)*	Stereo+PSC+adhoc image filters, 3D curves matching as well as autonomous agents were also tested	Kalman. Grey-level, Stereo, isolated or together	No road surface estimation, only more uncertainty in the PSC
Broggi, Bertozzi et al.	Stereo (v-d) Symmetry+PSC	Silhouette of head and shoulders	Stereo+PSC+entropy		Road surface estimation from v-d
Shashua et al. at MobilEye	Texture+PSC	Components: Gradient Or.*Mag + RR-AdaBoost Different training per pose and illumination conditions	Multi frame after tracking: gait, classification goodness over time, etc., multi-class help suggested	*(used but not detailed)*	Ego-motion estimation from a single image
Grubb et al.	Stereo (v-d)	Gradient magnitude + Q.SVM Different training per pose (F/R-S)	Classification goodness over time with the help of tracking	Kalman. Stereo	Road surface estimation from v-d
Zhao et al.	Stereo + PSC	Gradient magnitude + NN			
Soga et al.	Stereo + PSC	FDF + G.SVM		EKF	
Zhu et al.		HOG/Variable Blocks + L.SVM-Cascade AdaBoost			
Papageorgiou, Poggio et al.		Holistic: Basic Haar + Q.SVM Components: Basic Haar + Q.SVM-L.SVM			
Dalal et al.		HOG/Fixed Blocks + L.SVM			
Szardas et al.		Intensity Image + CNN			

Table 2.2: Classification Level parameters and overall results for the same systems.

	Learning ROIs size	Training Set	Testing Set	Syst. Perf (DetRate / FP) in %	Class. Perf (DetRate / FP) in%	Speed	Detection Range	Other Comments
Gavrila et al. at Daimler Chrysler	$75-100$ pix wide	$1,000$ pos	FGS ROIs	90/?	85 DetRate 2 FP/frame (ChamferSys)	$4-10Hz$ PIV $2.4GHz$ HFOV	$5-25m$ $30°$	Car running at $30Km/h$
Bertozzi, Broggi et al.	N/A	N/A	N/A	?	?	?		
Broggi, Bertozzi et al.	N/A	N/A	N/A	?	?	?	$10-40m$	Assume a
Shashua et al. at MobilEye	12×36	$25,000$pos $25,000$neg	$15,244$ in total	$95/\sim 0$	93.5/ 8 FPR	$20-25Hz$ (EyeQ, 640×480)	$3-25m$ $47°$ HFOV	
Grubb et al.	?	$1,500$ pos $20,000$ neg	150 pos $2,000$ neg	83.5/ 0.4 FPPW	75/2 FPR	$23Hz$ PIV $2GHz$ MMX (320×240)	$5-30m$	It is not clear if neg samples come from foreground segmentation. Sys. running up to 20Km/h
Zhao et al.	30×65	$1,012$ pos $4,306$ neg	FGS ROIs	$85,2$/ 3.1 FPPW	85.4/ 0.05 FPR	$3-12Hz$ (PII-450MHz 320×480)	?	Class.Perf. refers to experiments using 254pos and 363neg DB
Soga et al.	52 pix high	$2,392$ pos $30,841$ neg	$1,200$ pos $25,601$ neg	83.3/ 0.007per frame	90/ 0.3 FPR	?	$\sim 30m$	Training set increased till $14,052$ by transf. the original set
Zhu et al.	64×128	$2,478$pos $12,180$neg	SCAN	-	90/ 0.01 FPPW	4Hz dense scan PC 1.4GHz (320×240)	-	
Papageorgiou, Poggio et al.	64×128	$1,848$pos $11,361$neg	SCAN	-	90/ 0.01 FPR	$10Hz$ PPC-200MHz (348×256)	-	
Dalal et al.	64×128	$2,478$pos $12,180$neg	SCAN	-	85/ 0.01 FPPW	$0.14Hz$ with Zhu's params	-	
Szarvas et al.	30×60	200 neg	SCAN	-	90/ 0.01 FPPW	$10-40m$	-	

Chapter 3

Acquisition system and database

3.1 Acquisition System

In this section we present the acquisition system setup and its parameters. The system has been mounted on two different sedans: a *Ford Escort Mk5b* and an *Audi A2*. Thus the height and pose of the camera once mounted can differ from one setup to another due to the different dimensions of the vehicles and the fact that we use two suction pads to install the camera in the front windshield (Fig. 3.1). The camera used is a Bumblebee, from Point Grey[1], which consists of two Sony ICX084 color CCD sensors. The parameters are the following:

- Focal length: 6mm
- Effective Field Of View: 43.07°(HFOV), 32.97°(VFOV)
- Resolution: 640×480 (no downsampling is made at any stage)
- Effective sensor: 4.736(H)×3.552(V) mm
- Stereo baseline: 12cm
- Provides a three independent color channels and two grayscale images corresponding to the two sensors. Color is arranged in a Bayer pattern.
- Connected to a car–battery powered laptop via IEEE–1394 port. Saving images (to a 7,300rpm internal or external HDD) reaches about 10fps.

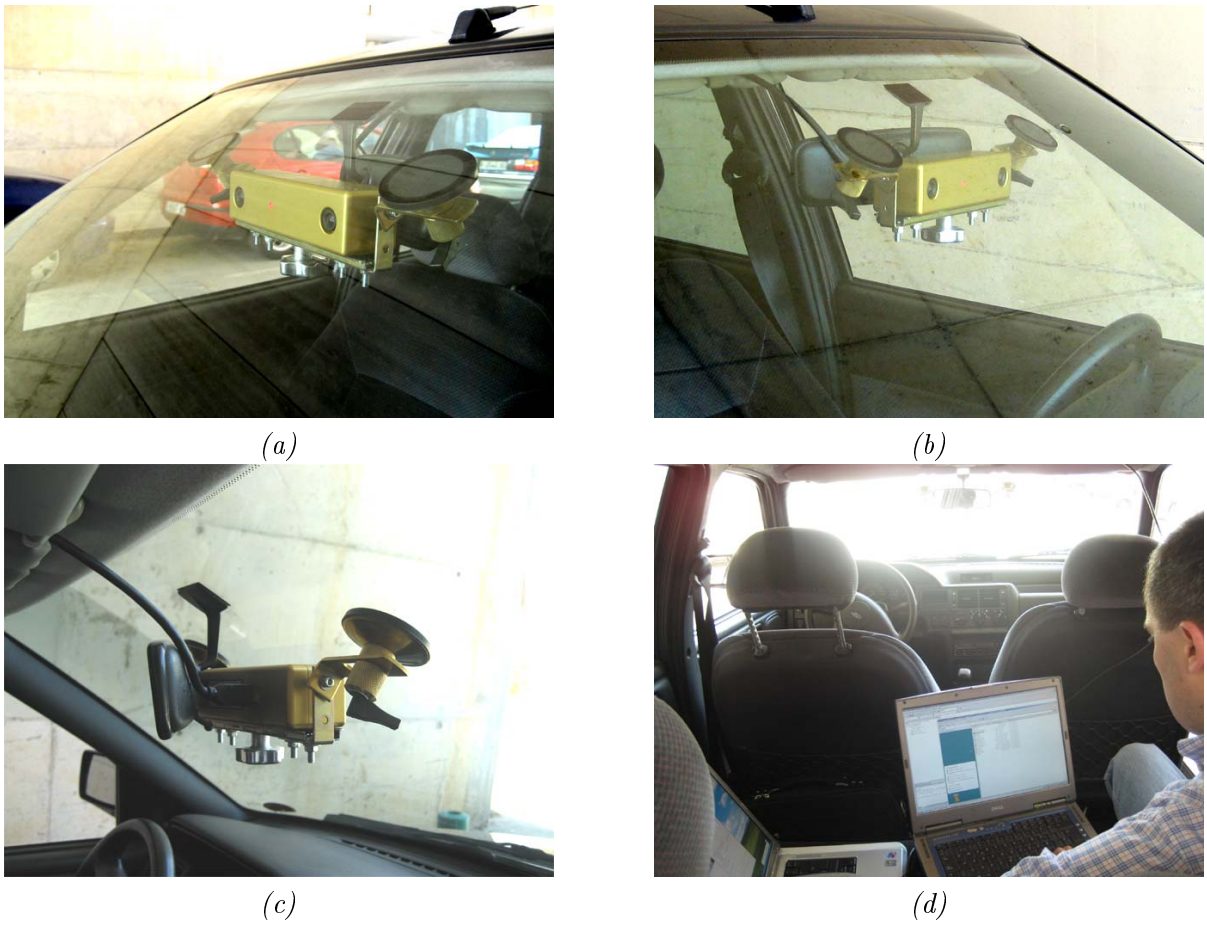
With the mentioned effective HFOV, if we assume a lane width of $3.5m$, which would represent the upper bound of the possible widths for urban scenarios, the nearest pavement zone captured is at $4.43m$, so supposing a pedestrian width of $0.85m$ (we will talk about pedestrian dimensions in Chapt. 4) we can state that the nearest detected pedestrian in the pavenent is at at

$$d = \frac{\frac{3.5m}{2} + 0.85m}{tan(\frac{43.07°}{2})} = 6.58m \ , \tag{3.1}$$

enough for our requirements (Fig. 3.2). Using a similar equation, we can also calculate the real height of the image column corresponding to this distance to be $3.88m$, again enough for our requirements.

[1] http://www.pointgrey.com

The maximum distance for detection is $50m$, where a pedestrian is about 12 pixels narrow in the image (Chapt. 4).



(a) *(b)* *(c)* *(d)*

Figure 3.1: Camera setup. *(a),(b)* Front and *(c)* back–view of the camera setup. *(d)* PC running the pedestrian detection application.

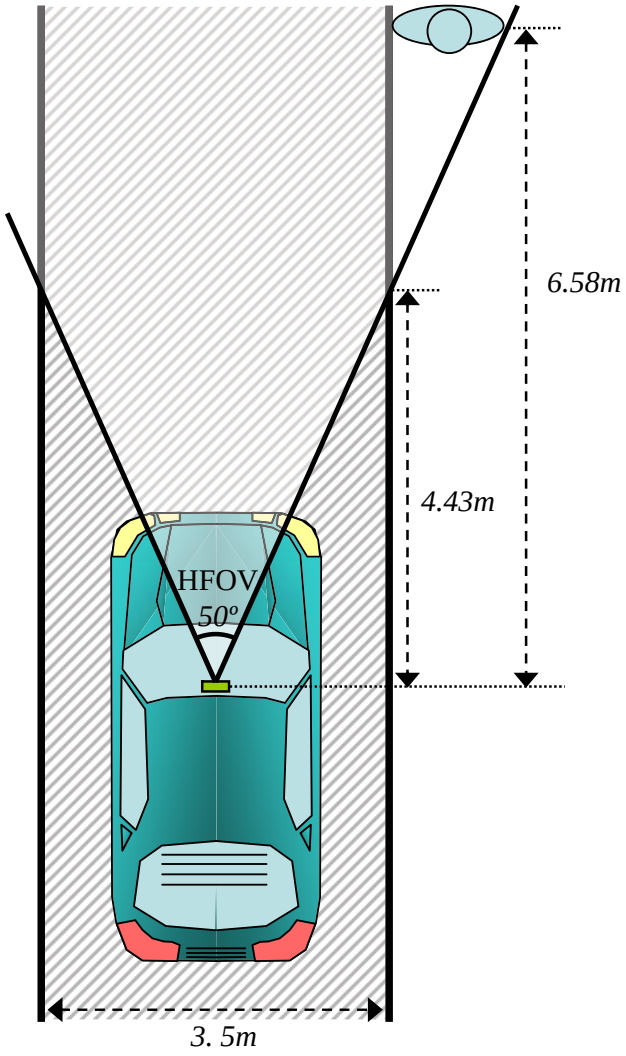


Figure 3.2: An horizontal field of view of $50°$ allows to see the the pavement at $3.43m$.

3.2 CER–01 database

CER–01 database was recorded in the Universitat Autònoma de Barcelona campus, the city of Cerdanyola (*CER* naming comes from here) and surroundings, using the aforementioned system.

Database samples were annotated from the camera's right grayscale channel[2], adjusting the selected windows to a projected grid laying on the 3D ground (see Sect. 4 for detailed explanation)—horizon line was manually adjusted in this selection step. Hence, no samples containing sky or building facades are likely to be selected. Samples have a 1:2 aspect ratio, and maintain the original dimensions of the image sequences. The database consists of $1,000$ positive and $5,000$ negative samples (no vertical reflection was used).

Variability is a crucial issue to be taken into account in order to construct a good model of our target. In this section, we present a list of features and properties whose diversity in several aspects is needed when designing a pedestrian database. Next, some examples illustraing this desired variability are shown.

[2]In our current development stage, color is just used to compute stereo in the pitch estimation step.

Multisize

In CER–01, samples are selected in the range from 5 to $50m$, thus coping with pedestrian dimensions from 140×280 to 12×24 pixels. No size normalization is made at the selection step. An obvious drawback when dealing with multisize samples is that a blurring effect appears depending on the size of the sample (actually it is the distance that varies), i.e. the ones at $\sim 50m$ seem to be blurred, thus affecting to the model learning (Fig. 3.3).

Figure 3.3: Multisize affects appearance of the samples (enhaced contrast for better visualization). *(a)* Far samples in the range of 40–$50m$. (b) Medium distance samples at $\sim 20m$. (c) Near samples at $\sim 10m$.

Background cluttering

When constructing the model from real example images, the learning algorithm shall discard the area out of the pedestrian shape as an intra–class feature. In order to do that, examples containing the most possible backgrounds should be selected. Fig. 3.4 illustrates several types of backgrounds.

Figure 3.4: Three categories of backgrounds (enhaced contrast for better visualization): *(a)* plain, *(b)* average textured and *(c)* cluttered. When performing classification, the first ones provide a clear outline of the pedestrian, so a priori, contour–based classifiers are likely to be favoured.

Illumination

Independently from the possible backgrounds, different illumination conditions lead to different results when extracting the features. For instance, with poor illumination, contours are not so clear and some body parts are missing. In Fig. 3.5 two possible cases are shown.

Figure 3.5: Illumination variability (enhaced contrast for better visualization). *(a)* Poor illumination conditions, specially in images with weak contrast, seem harder to classify. *(b)* and *(c)* illustrate scenarios with rich illumination, or at least, the minimum expected contrast. In this case, good illumination provides more useful information to perform the classification.

Pose

Pedestrians' pose is an intra–class variable that must be learnt by the classifier. Fig. 3.6 shows the three basic poses, all present in the database.

Figure 3.6: Three possible poses (enhaced contrast for better visualization): *(a)* Front–viewed *(b)* back–viewed and *(c)* side–viewed pedestrians.

Clothes

Diversity in clothes is also another important intra–class variable. For instance, Fig. 3.7 shows some examples, which are divided depending on the gender of the pedestrian. As expected, variability in women's clothes is higher than men's, going from trousers or short skirts, which generally allow legs localization, to long skirts and even chadors that cover most of the relevant expected features in a human shape. Of course, clothes can vary depending on the region of the world where we are.

Figure 3.7: Some examples of clothes in *(a)* men and *(b),(c)* women (enhaced contrast for better visualization).

CER–01 database includes samples with almost all the possible (and desirable) variations in a pedestrian database. However, in this database we discarded one interesting feature: occlusions. Hence, for the moment, pedestrians with strong occlusions are not taken into account in this data set.

Counter–examples

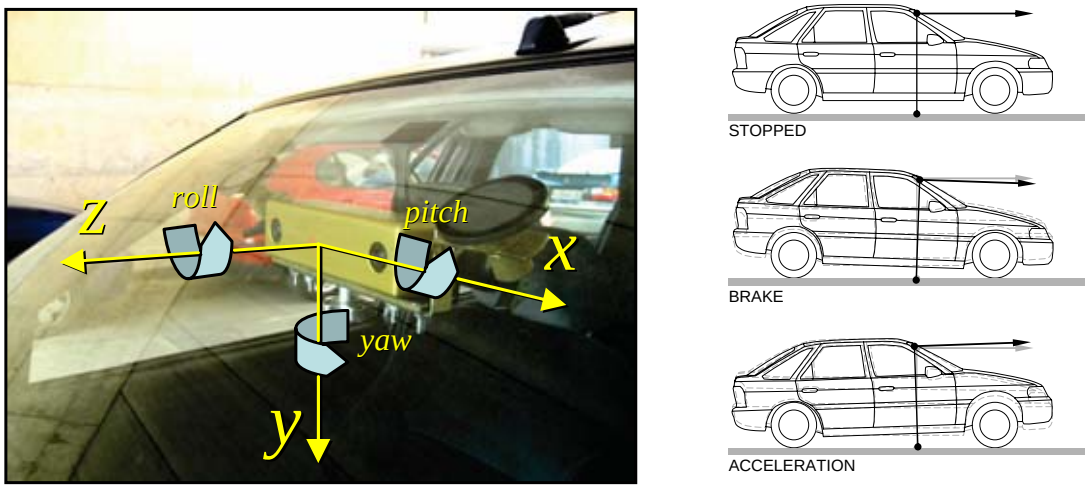
Counter–examples in CER–01 were selected from the ground area likely to contain pedestrians. Hence, they correspond to urban images containing any object or background in this area not being a pedestrian (Fig. 3.8).

Figure 3.8: Some counter–examples of the database (enhaced contrast for better visualization).

Chapter 4

Horizon Line Estimation

In our work, pedestrian detection is achieved by applying window classification techniques on a set of all the possible human–sized windows of the input image. In the current chapter, it is exposed why optimizing the number of windows is vital in our current system, and propose a stereo–based pitch and height estimation technique in order to automatically adjust the windows according to the current pose of the camera[1]. Fig. 4.1a illustrates the coordinate system of our camera and its variations in pitch angle and height when accelerating or breaking. As can be seen, yaw and roll angles can be assumed to be null in most of the traffic scenarios, but oscillations in pitch are too strong to be assumed null.



(a) *(b)*

Figure 4.1: *(a)* Coordinate system with the origin point in our camera. *(b)* Effect of the pitch angle and height of the camera when the vehicle is braking and accelerating.

[1]Along this master thesis, we use the *Horizon Line* name instead of just *camera parameters* since the former one tends to be more intuitive and easy to understand for the reader. On the other hand, they are equivalent concepts (with the camera parameters we can calculate the horizon line), so they can be used indistinctly.

With the pitch and height information, the road in the 2D image can then be scanned using back-projected pedestrian–sized windows from the 3D world to the image. In order to not extend too much this chapter, the equations needed to perform this scan and projection are explained in App. D.

4.1 Assumed pedestrian dimensions

In our system, pedestrian size constraints are similar to the assumed in [9]. This is, a pedestrian is defined as a person with height $H = 1.70m$ and width $W = 0.85m$ (we add a bigger margin in the width than [9], so that the information of extended arms and walking legs is not lost). Additionally, a standard deviation $\sigma = 0.1m$ must be assumed in order to cope with different sizes of pedestrians (Fig. 4.2). Te smallest window size is 12×24, which corresponds to a pedestrian at ~ 50meters [2].

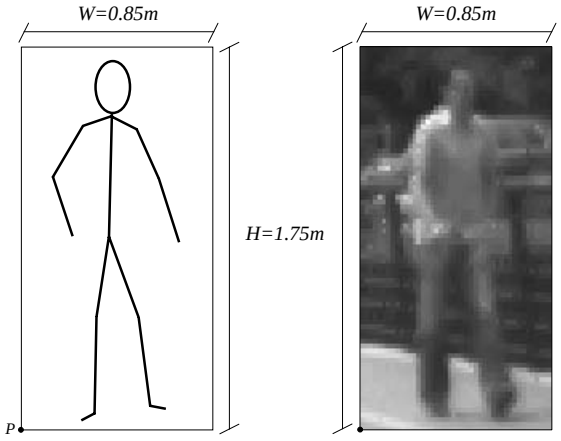


Figure 4.2: Standard pedestrian size. Given an image point P, the algorithm selects a $1:2$ rectangle corresponding to the dimensions W, H in the world. Size in pixels is calculated depending on the distance of P to the camera.

4.2 Previous approaches for defining the searching windows

The first intuitive approach to select the windows is to perform a brute–force search over the input image[3]. In this case, an exhaustive scan of all the possible windows at all the possible positions and sizes (maintaining the $1:2$ aspect ratio) would require 10^8 windows, and scanning just the half bottom part of the image with the 10% of the possible sizes would require 10^6 (Fig. 4.3).

In [82], the system shifts a 64×128 window over the image with increments of δ pixels, iteratively resizing the image from 0.2 to 1.5 times its original size (with increments of 0.1). Therefore, all possible

[2]Given an image subwindow of width $PedestrianImage_{width}$ (in pixels), the real distance from this window to the camera can be easily computed even with a single image. Let $f_x = f\frac{Image_{width}}{Sensor_{width}}$ be the focal length (in the $X-$axis), where $Image_{width}$ is in pixels and $Sensor_{width}$ is in millimeters. The distance is then defined as $d = f_x \frac{PedestrianReal_{width}}{PedestrianImage_{width}}$, where $PedestrianReal_{width}$ is in meters and $PedestrianImage_{width}$ is defined in pixels.

[3]All the numbers are calculated with a standard 640×480 input image, i.e.the size of our acquired images.

locations are scanned in a multi–scale fashion. In this case, near 3 million windows are selected for $\delta = 1$; about $675,000$ windows for $\delta = 2$; and $300,000$ windows for $\delta = 3$. Note that if the minimum size for a window is reduced to our system's 12×24 pixels dimensions, the total number of windows increases significantly.

(a) *(b)*

Figure 4.3: Brute–force search. *(a)* Windows of fixed size shift δ pixels to get a new position to scan. Some pedestrians are bounded in the selected windows, but others are not. *(b)* Same approach, but now variable size windows. Now pedestrians of different sizes, i.e. at different distances, can be bounded.

Obviously, this brute–force approach is not affordable in a system that aspires to reach real–time performance. In order to speed up the whole process, some authors propose to restrict the searching area to image locations determined from a priori knowledge of the current ground plane. Since it is not possible to find pedestrians in the sky area, and it is not useful to detect a pedestrian on the top of a tree, the scanning windows are restricted to the ground plane.

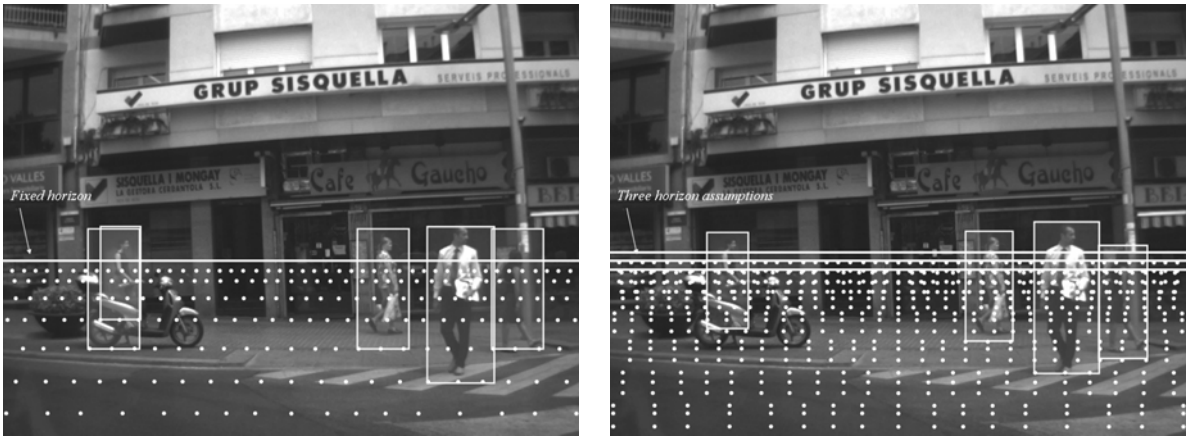
For instance, in [87], *Ponsa et al.* make an initial calibration that fixes a pitch angle that will be used to determine the 2D image position of any road 3D point. Hence, a 3D grid, sampling the road plane, is projected on the 2D image. The projected grid nodes are then used to define the bottom–left corners of searching windows (Fig. 4.4a). In that example, to cope with pitch and height variations, only a relatively small range of possible pitches and heights of the camera are explored. This approach is valid since the addressed problem is vehicle detection in highway scenarios, so interframe car accelerations and road imperfections are in general much lower than in urban scenarios.

Other works, like the pedestrian detector described in [9], use a constant pitch/height approach in urban scenarios. This is not a realistic assumption since camera pose is continuously modified not only by the mentioned factors, but also by speed bumpers, sudden changes in the road slope, etc (Fig. 4.4*(a)*).

In order to cope with different road orientations, [87] also tests three different camera pitch angles (See Fig. 4.4 *(b)*), improving robustness in vechicle detection in highway scenarios. Using the same approach in urban scenes would require a wider range of possible pitchs to test, since the orientation changes are bigger. Consequently, the number of windows would increase again too much to work in real–time.

As introduced earlier in Chapt. 2, several works [54, 4, 8, 19] use a image stabilization technique based on vertical optical flow analysis. It is based on the study of a row–wise histogram computed from the edges of the current image. Histograms from consecutive frames are used to compute their corresponding vertical offset. This approach, although very efficient in terms of computing time, has two important drawbacks. First of all, the image should contain several horizontal features, since the

whole process relies on the accumulation of horizontal edges. Secondly, current camera orientation is related to the previous frame, therefore, since relative errors can not be removed, the global error value increases with the time—the drift problem.



(a) *(b)*

Figure 4.4: Fixed pitch assumption. *(a)* Fixed pitch assumption, windows have different sizes depending on their corresponding position in the 3D world. *(b)* Three constant pitch assumption used in [87] to detect vehicles. Pedestrians are better fit at the cost of a higher number of ROIs.

Facing up to the problem of using the pitch to correctly adjust the windows to any scenario, [25] proposes a technique for estimating vehicle's yaw, pitch and roll. It is based on the assumption that some parts of the road have a constant width (e.g., lane markings). Similarly, [66] proposes to estimate camera's orientation by assuming that the vehicle is driven along two parallel lane markings. Unfortunately, none of these two approaches can be used for an urban scenario, since lanes are not as well defined as those of highways.

Having in mind the mentioned problem, *Lefée et al.* [64] propose a feature–based approach to compute 3D information from a stereo vision system. Features such as zebra crossings, lane markings or traffic signs painted on the road (e.g., arrows, forbidden zones) are used. Differently to classical approaches, this scheme classifies each pixel according to the grey values of its neighbours. Then, points lying on the road are used to estimate camera's position and orientation. Although presented results are quite promising, the traffic–based–feature detection constraint is one of the main disadvantages of this tecnique. Moreover, not every urban road contains features scuch as the aforementioned.

A different approach was presented in *Labayrade et al.* [63]. The authors propose an efficient technique able to cope with uphill/downhill driving, as well as dynamic pitching of the vehicle. It is based on a v–disparity (App. D) representation and Hough transform. The authors propose to model not only a single plane road (a straight line) but also a non–flat road geometry (a piecewise linear curve).

4.3 Proposed technique

In the sense of automatically estimating the pitch and height of the camera in order to adjust the searching windows, we have developed a plane fitting technique based on RANSAC and 3D data information. The proposed approach, presented in [93] and further explained in [94], is presented next.

It aims to compute camera's position and orientation, avoiding most of the assumptions previously mentioned. The technique consists of two stages. Initially, the original 3D data points are mapped onto

a 2D space. Then, a RANSAC based least squares fitting is used to estimate the parameters of a plane fitting of the road; at the same time, camera's position and orientation are directly computed, referred to that plane. Independently of the road geometry, the provided results could be understood as a piecewise planar approximation, due to the fact that road and camera parameters are continuously computed and updated. The proposed technique could be indistinctly used for urban or highway environments, since it is not based on a specific visual traffic feature but on raw 3D data points. In addition, no error is accumulated since it works in a frame–by–frame basis and no initialization is required.

3D data point projection and noisy data filtering

Let $S(r, c)$ be a 3D image obtained from a binocular stereo, with R rows and C columns, where each array element (r, c), $r \in [0, (R-1)]$ and $c \in [0, (C-1)]$, is a scalar that represents a surface point of coordinates (x, y, z), referred to the sensor coordinate system. Fig. 4.5*(left)* illustrates the extracted 3D data projected to the image plane. In Fig. 4.5 *(right)* figure, the points cloud is rotated to show the computed depth.

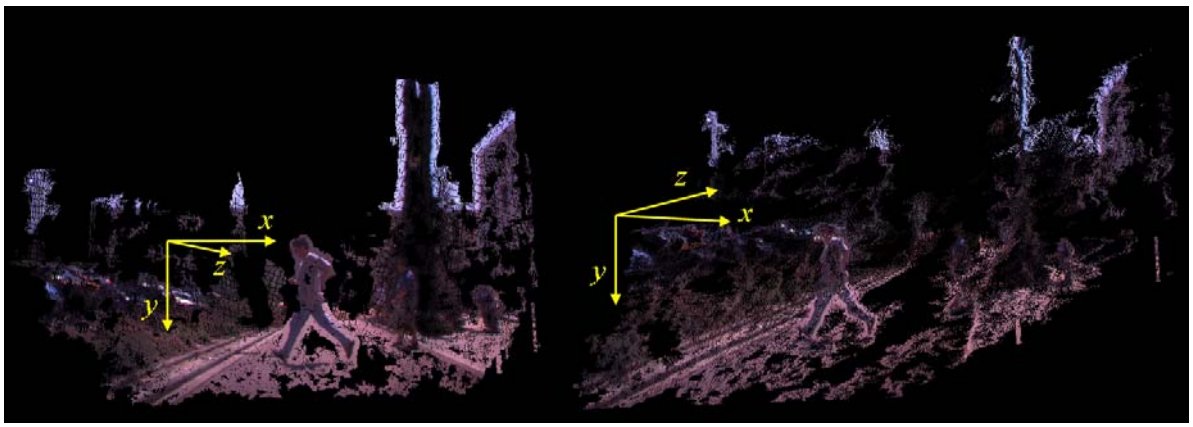


Figure 4.5: *(left)* Back–projection of the computed 3D points. *(right) Rotated view to make the depth visible*.

Notice that vertical variations between consecutive frames—due to road imperfections, car accelerations, etc.—will mainly produce changes on camera's position and pitch angle. In other words, yaw and roll angles are not so affected by those variations. Therefore, without loss of generality, they are assumed to be constant.

The aim at this stage is to find a compact subset of points, ζ, containing most of the road's points. Additionally, noisy data points should be reduced as much as possible in order to avoid both a very time consuiming processing and a wrong plane fitting.

Assuming constant yaw and roll angles, original 3D data points, $S(r, c)$, are mapped onto a 2D discrete representation $P(u, v)$, where $u = \lfloor S_y(r, c) \cdot \sigma \rfloor$ and $v = \lfloor S_z(r, c) \cdot \sigma \rfloor$. σ represents a scale factor designed as: $\sigma = ((R + C)/2)/((\Delta X + \Delta Y + \Delta Z)/3)$; and Δ is the working range in every dimension—on average $(34 \times 12 \times 50)$ meters. Every cell of $P(u, v)$ keeps a reference to the original $3D$ data point projected onto that position, as well as a counter with the number of mapped points. Fig. 4.6*(top)* shows a 2D representation obtained after mapping a 3D cloud—every black point represents a cell with at least one mapped point. Notice the large amount of noisy points highlighted on the figure.

Finally, points defining the ζ subset are selected as follow. Firstly, those cells of $P(u, v)$ containing less mapped points than a predefined threshold are filtered, by setting to zero its corresponding counter—

points mapped onto those cells are considered noisy data. The threshold value was experimentally set as a percentage of the maximum amount of points mapped onto a cell. On average, every cell of $P(u, v)$ corresponds to an area of (5.6×5.6) cm of the original cloud of points; the maximum amount of points mapped onto a cell is about 250 points. The threshold value was set 6% of that value—i.e., 15 points. After filtering noisy data, a selection process picks one cell per column. It goes bottom–up through every column and picks the first cell with more points than the aforementioned threshold. $3D$ data points mapped onto selected cells define the sought subset of points, ζ. Fig. 4.6*(bottom)* depicts the cells finally selected. The ζ subset of point gathers all the 3D points mapped onto those cells.

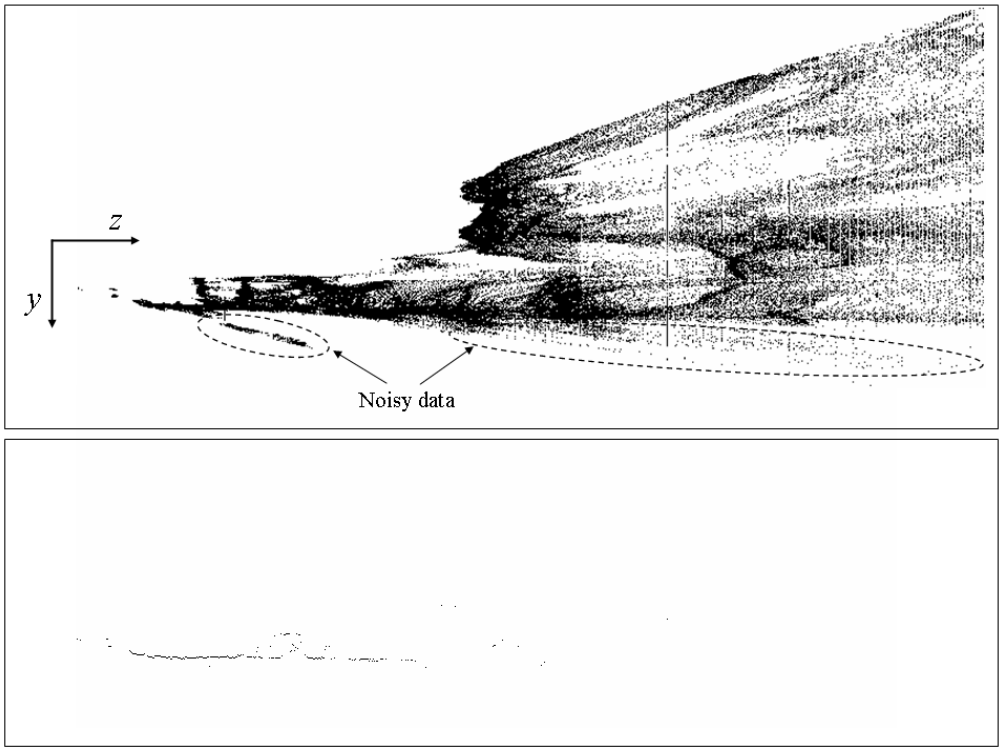


Figure 4.6: *YZ projection. (top)* Whole cloud of points projected on the YZ plane. *(bottom)* Selected points to be used by the RANSAC technique.

RANSAC based plane fitting

The outcome of the previous stage is a subset of points, ζ, where most of them belong to the road. In the current stage a RANSAC based technique [37] is used for fitting a plane $ax + by + cz = 1$ to those data. In order to speed up the process, a predefined threshold value for inliers/outliers detection has been defined (a band of $\pm$5cm was enough for taking into account both 3D data point accuracy and road planarity). An automatic threshold could be computed for inliers/outliers detection following robust estimation of standard deviation of residual errors [90].

The proposed plane fitting works as follows.

Random sampling: *Repeat the following three steps K times or up to the current plane parameters are similar to those ones computed from the previous frame (during the first iteration this second condition is not considered)*

1. Draw a random subsample of n different 3D points from ζ (lookking for a trade–off between CPU processing time and final result, n has been finally set to 6)
2. For this subsample, indexed by k, where $k = (1, ..., K)$, compute the plane parameters (a, b, c) by using the least squares fitting approach [114], which minimize the square residual error $(1 - ax - by - cz)^2$.
3. For this solution $(a, b, c)_k$, compute the number of inliers among the entire set of 3D points contained in ζ, as mentioned above using $\pm$5cm as fixed threshold value.

Solution:

1. Choose the solution that has the highest number of inliers. Let $(a, b, c)_i$ be this solution.
2. Refine $(a, b, c)_i$ using its corresponding inliers.
3. In case the number of inliers is smaller than 10% of the total amount of points contained in ζ, those plane parameters are discarded and the ones corresponding to the previous frame are used as the correct ones. In general, this could happen when 3D road data are not correctly recovered when occlusion or other external factor appears.

Finally, camera's position and orientation, referred to the fitted plane, are easily computed. Camera's position—height—is estimated from the plane intersection with the Y axis $(1/b)$ and the current plane orientation. Camera's orientation—pitch angle—is computed from the current plane orientation.

4.4 Experiments

The explained technique has been tested on different urban environments. Fig. 4.7 displays variations in the camera height and pitch angle, both referred to the current fitted plane, for an urban sequence containing a gentle downhill, vehicle's accelerations and a speed bumper.

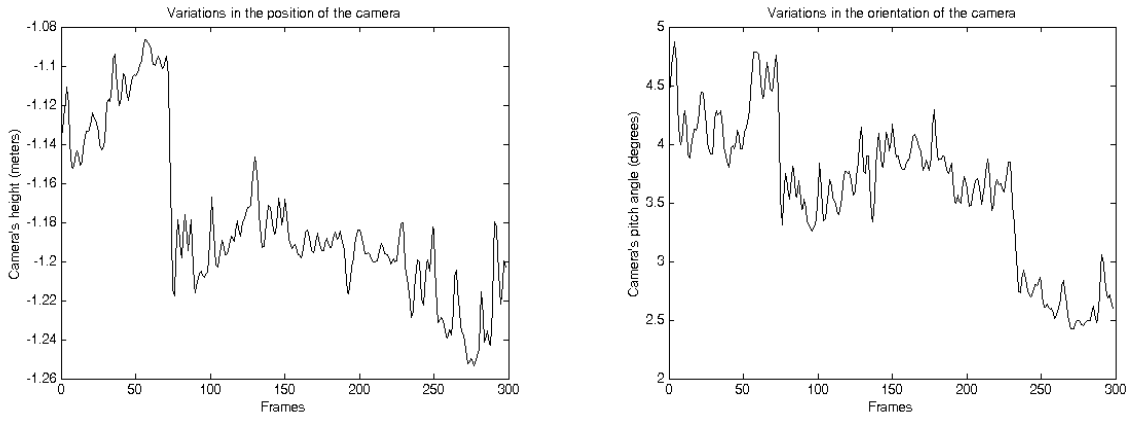


Figure 4.7: Variations in the camera height and pitch angle, related to the current fitted plane. Note that only 2fps are plotted.

A row–wise histogram based approach (HB), similar to the one proposed in [54, 4, 8, 19] has been implemented and compared with the proposed technique (PT). It consists of computing vertical offset between row–wise histogram of edges from consecutive frames. This vertical offset is referred to an

initial vanishing line position defined by the user. Fig. 4.8 depicts the vanishing line evolution in two short sequences, both corresponding to the same video sequence. Since the HB approach is affected by the drift problem, several initializations were performed along the whole video sequence. In the first case Fig. 4.8*(right–top)* the vanishing line computed by HB is driven to a wrong position since the scene contains a large amount of horizontal edges, in particular HB is affected by those belonging to the bridge. In the second case (Fig. 4.8*(right–bottom)*) an uphill driving scenario is presented. In this case, both techniques have the same behaviour in the first frames, but it can be noticed how the accumulation of errors in HB drives the vanishing line to a wrong position. Recall that in both cases HB has been manually initialized correctly; on the other hand, the current technique was able to process the whole video sequence (about 2 hours of recording) without any kind of initialization.

Finally, since ground truth is not known beforehand, several frames were chosen and used to validate the obtained results. In these cases, their corresponding vanishing lines were manually computed[4] (MC) and used as ground truth to compare ones automatically obtained by the PT and the HB approach. Fig. 4.8*(left)* presents two frames with their corresponding MC vanishing lines. Four MC vanishing lines per sequence are presented in Fig. 4.8*(right)*—frames 10, 20, 30 and 40. Notice that in both cases results obtained with the proposed technique are more similar to the real ones (MC) than those computed with HB.

Fig. 4.9 shows the most relevant frames of a short sequence that contains three road orientation changes. First the car adjusts a flat road (until the zebra crossing), then it enters an uphill (notice the slope variation in frame 0) and finally turns to flat again. It can be appreciated how the technique correctly adjusts the estimated horizon.

Fig. 4.10 illustrates the importance of having the right estimation of camera's position and orientation. note that any pitch angle has a corresponding vanishing line, namely horizon, since any ground point at infinite distance lays on this 2D image line. Thus, searching bounding boxes can be computed by using the vanishing line value as input information. The first three image pairs show the projected 3D grid and some search windows—just some illustrative examples are highlighted—when a fixed vanishing line is assumed. Notice the problems to fix a correct horizon to cope with all the possible road orientations. On the contrary, *(d)* images present the grid and nearest windows by automatically computing the vanishing line with the explained technique.

4.5 Summary

In this chapter we have presented a new technique to estimate position and orientation of the camera that is used to reduce the scanned windows passed to the classification module in our pedestrian detection system. Next, we comment the relevant points of the method:

- The proposed technique adjusts the horizon line by computing the current relative pitch and height between the camera and the ground plane. Results are promising, and provide a new solution without the constraints usual applied to the same problem—i.e. presence of lanemarkings, traffic–features, etc. The window reduction goes from the 10^8 of a full scan or 3×10^6 of [82] to just $2,000$ with our current scanning parameters (i.e.$0.5m$ step in x and z).

- The information provided by this technique is still useful even if a stereo–based foreground segmentation is used. This is, besides the fact that segmentation can make use of the 3D data already used by the technique (thus saving time), knowledge about horizon position can support the ROIs generation step in case of having blind zones or object occlusions with the stereo. In addition,

[4] By drawing two parallel lines in the 3D space and getting their intersection in the image plane

free–space techniques can also use the horizon line information to discard regions of the ground without a significant number of 3D points.

However, there are still problems to solve and issues that can be optimized. Next, we list some of the planned improvements for the technique:

- Ground truth. As can be seen in Fig. 4.8, the RANSAC based plane fitting technique tends to follow the manually computed pitch trends. However, the same plots illustrate that the results do not adjust *to the pixel* to the manual estimation. First, a technique to compute the correct real plane at each frame must be developed, since manually estimation can also be subject to errors. Then, this estimation should be used as ground truth to make all the comparisons between algorithms.
- Estability and reliability. In addition, the plane fitting is noisy in some complex scenes (hence the estimated pitch), perhaps due to 3D noisy data. Consequently, inestabilities can disturb the searching windows adjusting. In this matter, two ideas could be tested. First, the *three–pitch–constraint* [87] together with the plane fitting. Secondly, adding some temporal coherence to the current algorithm, like Kalman filtering, is a must, since per–frame estimation can produce noisy results when applied in grounds with complex 3D objects (e.g. litter bins, streetlights, kerbs, etc).
- Real time optimizations. The current implementation takes 250ms both in constructing 3D data and fitting the plane. This timing is quite slow, and should be improved to reach hard real time requirements.
- Complex ground profiles. Finally, some work must be done in roads where the profile is not a simple plane but a curve. In this cases, near ground points are correctly adjusted, but further ones are not.

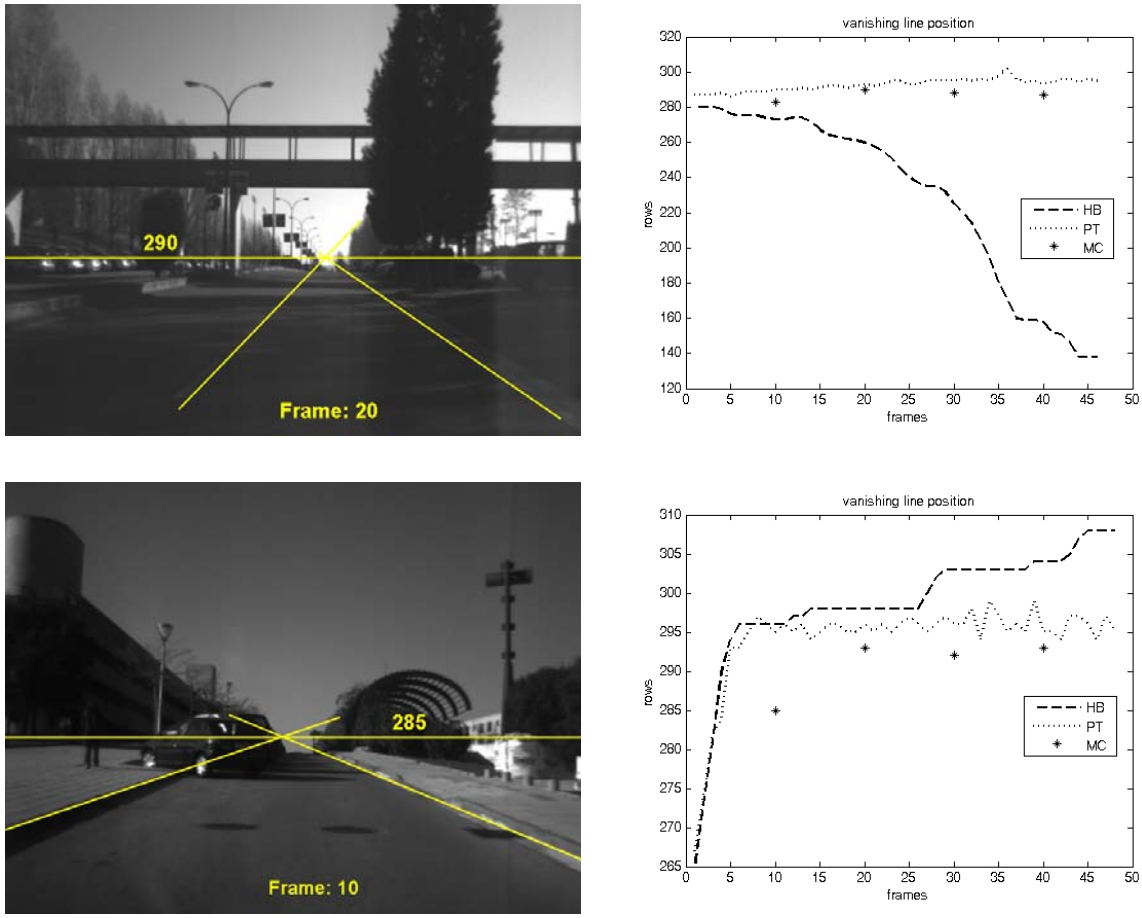
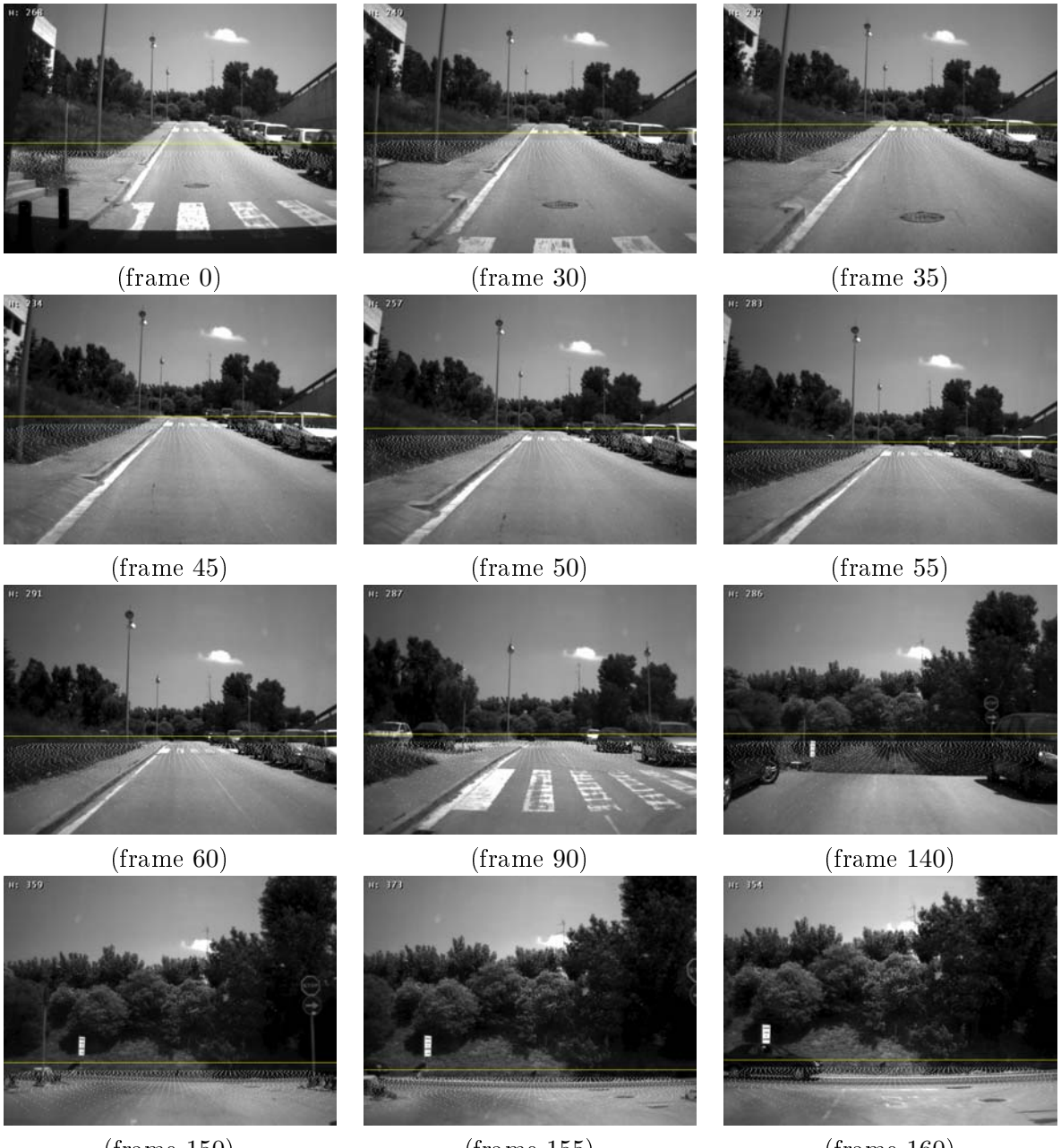


Figure 4.8: Illustrations of two different scenarios: flat road and uphill driving. Left: Ground truth manually computed for intermediate frames of the corresponding sequences. (Note: row 0 is located at the top of the images, i.e. an horizon line located at row 300 corresponds to a lower position in the image than an horizon at row 200). Right: Vanishing lines computed with the proposed technique (PT) and with a histogram based (HB) approach; additionally, four manually computed (MC) vanishing lines per sequence are presented.



(frame 0) (frame 30) (frame 35)

(frame 45) (frame 50) (frame 55)

(frame 60) (frame 90) (frame 140)

(frame 150) (frame 155) (frame 160)

Figure 4.9: Horizon estimation in a test sequence.

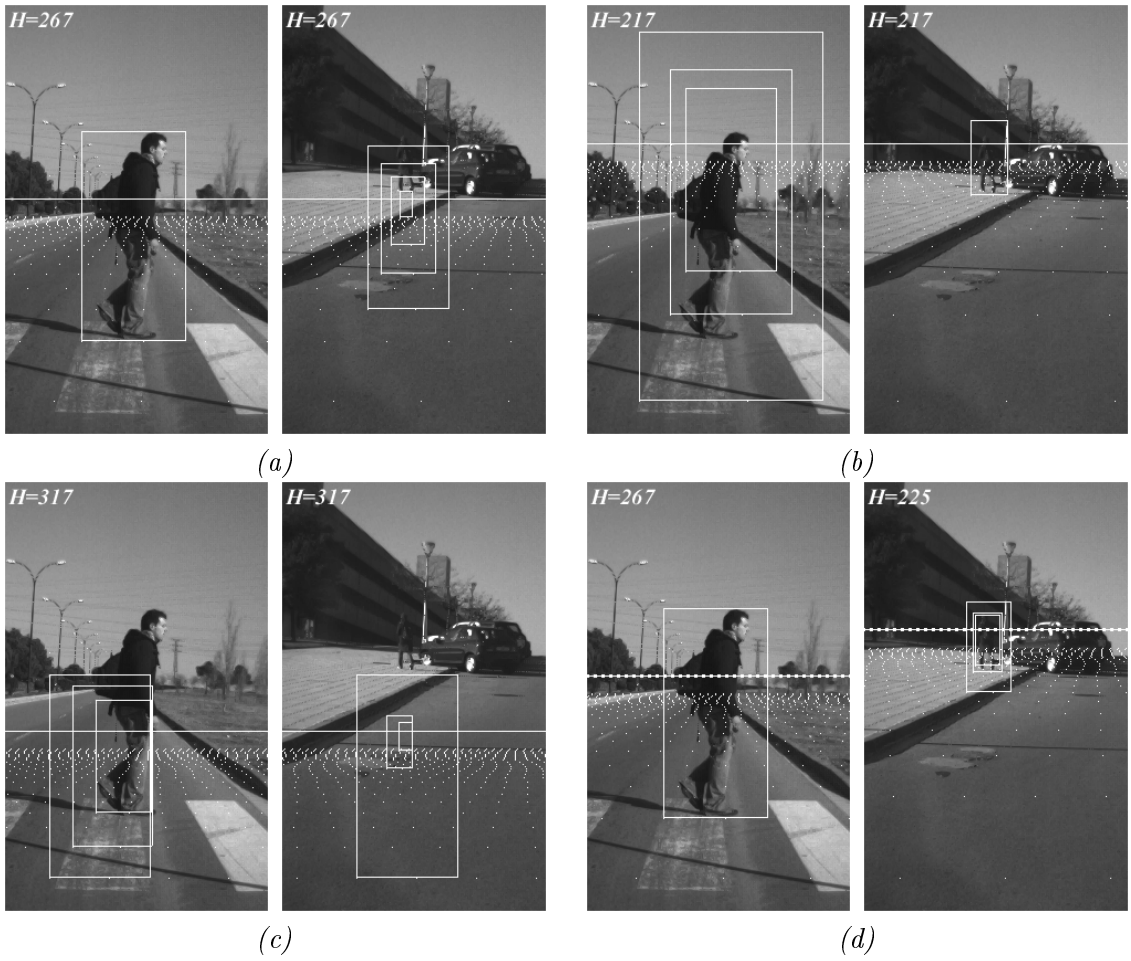


Figure 4.10: Searching bounding boxes using fixed and automatically computed vanishing lines. In all the cases highlighted boudning boxes correspond to the nearest to the pedestrian. *(a)* fixed vanishing line for an uphill driving scenario. *(b)* Fixed vanishing line assuming flat road. *(c)* Fixed vanishing line for a downhill driving. *(d)* Automatically computed vanishing line by using the proposed technique. Notice that, only in the latter case, the vanishing line position is correctly placed in both scenarios.

Chapter 5

Pedestrian Model

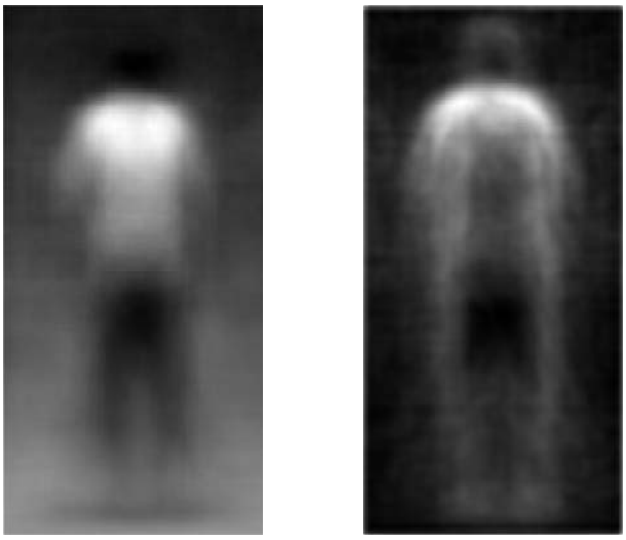
Modelling a standard pedestrian is not an easy task due to the high intra–class variability, i.e. different pedestrians can have very different appearance, as was seen in Chapt. 3. Perhaps the most intuitive way to construct the model is by manually selecting the features that define our idea of a pedestrian. For instance, Fig. 5.1 illustrates the average of all the pedestrians in CER–01 database using original grayscale and edge values. In this case, a pedestrian model consisting in the shoulders shape is the most clear visual feature. However, manual selection can be biased by human experience, so the best features (the most representative for pedestrians) may not be the chosen ones. In the appearance–based classification section in Chapt. 2 we exposed some common issues related to features. In Computer Vision, a feature can be defined as an attribute of the image data. In order to select the singular features that will define our model, there is a wide consensus around the use of automatic learning algorithms [79, 120, 84, 111, 30, 121].

The work presented in this chapter is focused on the appearance classification given an image window. Classification techniques first build a model by making use of training examples and a learning algorithm, and then use this model to take the class decision. The most used learning algorithms are mainly Support Vector Machines, Neural Networks and AdaBoost. In our case, the algorithm used to test the features performance is Real AdaBoost (App. A), because of its good performance in many similar works.

First, we test two Haar wavelets sets, Simple and Extended, the first one already tested in pedestrian detection [111], and both them used in face detection [112, 68]. Then we test edge orientation histograms (EOH), originally proposed in [65] for face detection. Having seen the results of these sets, we propose to use the combination of a Haar set together with EOH to improve the classifier performance. Additionally, we also test differential invariants and the local jet, and then the structure tensor orientation, which represent novel proposals in this task. Finally, the new histograms of oriented gradients (HOG) feature [30] has been analysed and compared to our proposal.

5.1 Performance Evaluation

All the tests in this chapter are thought to measure classification performance rather than detection performance. Hence, the tests are carried out using a testing subset extracted from the database. Contrary to works that present results based on the classification over all the possible detection windows in the input image, the *testing set approach* focuses the tests just on the studied issue, in our case features comparison for pedestrian targets. In this way, the test is independent from overlapping bounding boxes matters, thus independent from any multiple detections measures; and also avoids the appearance of a



(a) *(b)*

Figure 5.1: Average pedestrian from *(a)* original grayscale and *(b)* sobel edge images. All samples are normalized in size and aspect ratio and then the aritmetical average was calculated.

huge number of negatives that could drastically reduce the number of false positives in a quite artificial way.

The complete database consists of $1,000$ positive and $5,000$ negative samples, from which the experiments select 700 positive and $4,000$ negative samples for the training set, and 300 positives and $1,000$ negatives for the testing set, so that there is no intersection between both sets in an experiment.

Then, Real AdaBoost selects the 100 most representative features for the pedestrian model. This number is taken as a trade–off between computation time an good generalization for the studied sets. Table 5.1 shows the total number of features of each one of the studied sets, from which the 100 features are extracted.

In order to measure and illustrate the idoneity of the studied features for the pedestrian detection problem, *Receiver Operating Characteristic* (ROC) curves have been used. Appendix C presents a brief introduction to the most relevant aspects of ROCs and the rates used in this master thesis to measure the results. Each plot's legend includes the *Area Under Curve* (AUC) value of each feature, which is calculated in the plotted curve range (i.e. from 0.0 to 0.1 FPR) and not with the full ROC space. The aim of using this range and not the whole ROC space is to just take into account the range where a real pedestrian classifier can work, i.e. at low false positive rates. At 0.1 FPR, the curves are almost stabilized and do not provide further improvement in the rest of the ROC, so using the proposed range seems to be the most reliable option. Obviously, the higher the AUC value, the better the classifier.

In the case of CER–01 database results, the plotted curves correspond to the average of 4 different experiments, where the training and testing set are selected randomly with no intersection. If not specified, a $\sigma = 1$ gaussian smoothing is applied as a preprocessing to each sample. The minimum size of the sample (classification window) is 12×24 pixels (Sect. 4), so all the features are scaled to this proportional window size (see Sect. 5.2.4) . Additionally, the feature selection process discards features whithout a mimum area of 2×2.

Fig. 5.2 illustrates the methodology we follow to make the tests.

Table 5.1: Total number of features for each of the studied sets.

Feature set	Features
Simple Haar	38,721
Extended Haar	51,434
Simple Haar with fixed size feature	2,365
Differential Invariants and local jet	46,817
Structure Tensor (4 bins)	83,490
Edge Orientation Historams (4 bins)	83,490
Edge Orientation Historams (9 bins)	500,940
Combination SHaar and EOH (4 bins)	122,211
Combination SHaar and EOH (9 bins)	539,661
Combination EHaar and EOH (4 bins)	134,924
Combination EHaar and EOH (9 bins)	552,374
HOG (4 pix blocksize, 9 bins)	1,980

5.2 Haar Wavelets

In 1997, inspired by works like [73] and [103], *Oren et al.* [82] introduce Haar wavelet templates to capture the structural similarities between various instances of a class. These wavelets represent a fast and simple way to calculate region derivatives at big scales by means of computing the average intensities of concrete subregions.

5.2.1 Feature definition

A feature of this set is defined as the difference of intensity between two defined areas (white and black), in a given position inside a region R Fig. 5.3):

$$Feature_{Haar}(x, y, w, h, type, \alpha, R) = E_{white}(R) - E_{black}(R) \ , \tag{5.1}$$

where (x, y) is the bottom–left position in the template; w, h are rectangle's width and height; *type* is one of the configurations listed in Fig. 5.5; and $\alpha \in \{0°, 45°\}$, if the set includes 45° rotated templates. $E_{white}(R)$ and $E_{black}(R)$ represent the sum of intensities of white and black areas of the template respectively.

5.2.2 Template sets

The set proposed in [82] defines three types of template configurations (Fig. 5.4a) that capture changes in intensity along the *(a)* horizontal direction, *(b)* vertical and *(c)* the diagonals (this set is referred in this master thesis as *Basic Haar Set*). The template is then moved and scaled over a region R, so different features are computed by using a single template. In order to achieve a good resolution when scanning R with the template, [82] proposes the use of the *overcomplete dictionary*, also named *quadruple density dictionary* (Fig. 5.4*(b)*). Hence, a better spatial resolution provides more accurated and relevant features.

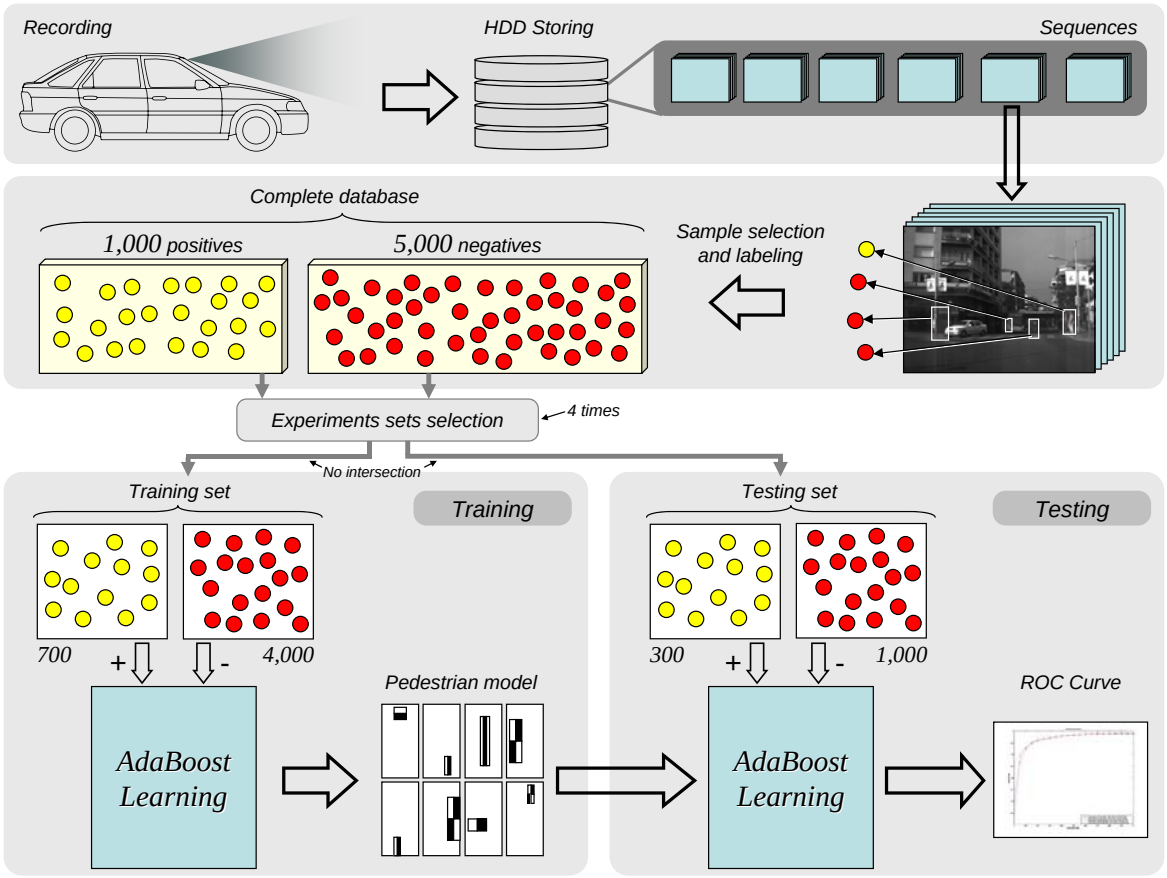


Figure 5.2: Experiments methodology. The results (ROCs) presented in this chapter correspond to the average of four different experiments ($n = 4$), each one with a training and testing set (with no intersection between them).

Later in 2001, *Viola and Jones* [112] contribute with some new improvements to the idea, but now in the face detection problem. Since image regions can have different dimensions, normalization is required to stablish equivalence between the features computed in each example. In addition, images must be normalized according to their variance in order to minimize the effect of different lighting conditions. This is achieved by multiplying the feature values rather than multiplying the pixels, once the current median, variance and dimension of the region are known.

In the same paper, two new templates are introduced: horizontal and vertical lines. Mathematically, they represent the second order partial derivatives in x and y, respectiely. The set containing the three original basic templates and these two new ones will be referred as *Simple Haar set* (Fig. 5.5).

Another improvement of the set is presented by *Lienhart and Maydt* in 2002. Here, the Simple set is extended with center–surrounding templates, two new versions of the horizontal and vertical line templates, and the 45° rotated versions of all them (Fig. 5.5). This new set (from now on *Extended Haar Set*)) represents an improvement over the Simple set used by Viola and Jones when used in face detection [68].

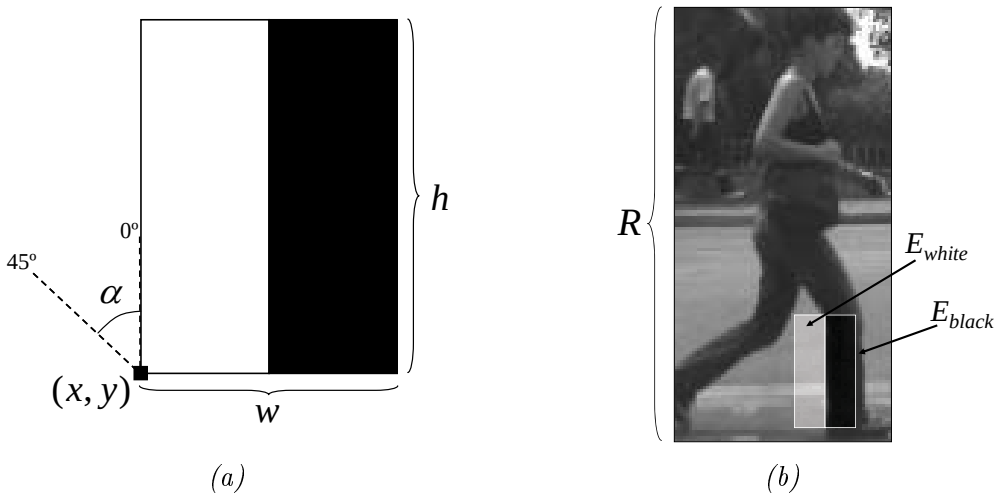


Figure 5.3: *(a)* Haar feature example with parameters (x, y, w, h, (a), 0°, R), where (a) template can be seen in Fig. 5.5. *(b)* Example of a feature computed in a real image region containing a pedestrian.

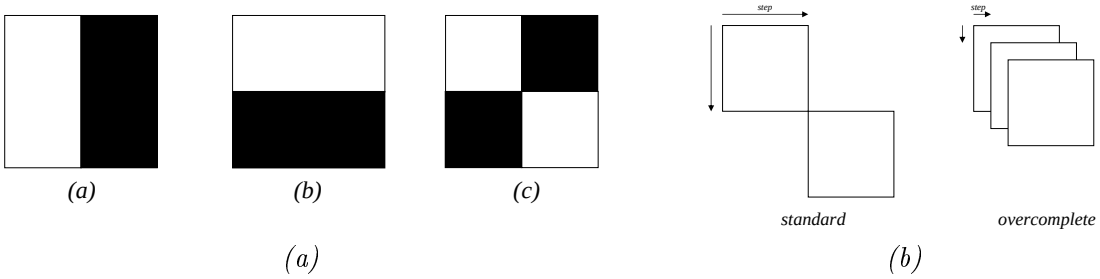


Figure 5.4: Haar features. *(left)* Basic Haar set used in [82]. *(right)* Overcomplete scanning used in the same paper.

5.2.3 Integral Image representation

Perhaps the most interesting contribution of *Viola and Jones* [112] is the *Integral Image*: an intermediate representation used to compute the rectangle features in a very fast way. The integral image at location x, y contains the sum of the pixels above and to the left of x, y, inclusive:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y') \ ,$$

where $ii(x, y)$ is the integral image, and $i(x, y)$ is the original image. Fig. 5.6 illustrates the construction and benefits of using this representation when dealing with rectangle area features.

Lienhart and Maydt also present a method to calculate a 45° version of the integral image (needed to calculate the features that make use of the rotated templates) in just two passes over all pixels (we refer to [68] for more details).

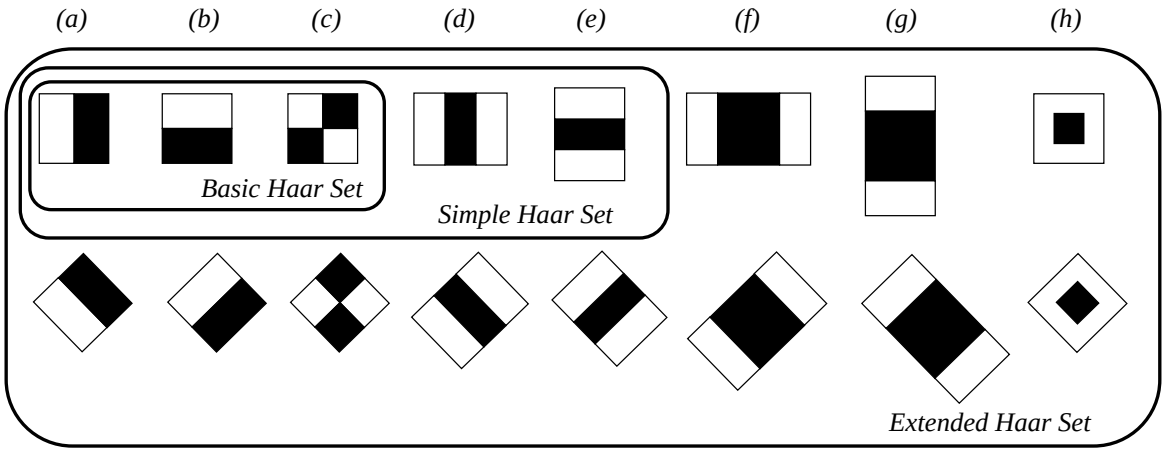


Figure 5.5: The described sets have been incrementally extended.

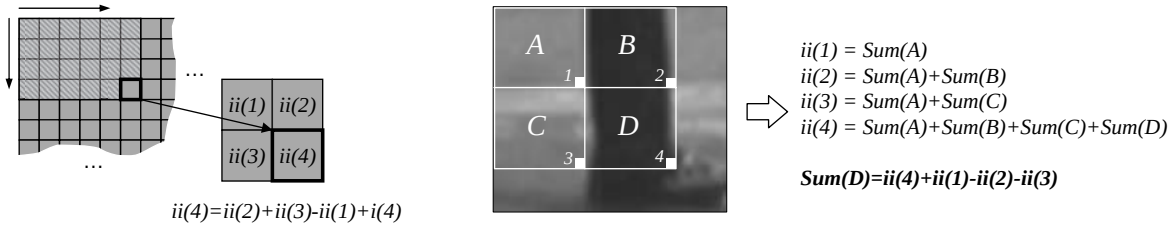


Figure 5.6: *(a)* Integral image (*ii*) can be computed in one pass over the original image. Each *ii* pixel uses the values of its left and top neighbours and the value of the original image, so the image is constructed incrementally. *(b)* Retrieving rectangular area sum (*Sum()*) using the integral image. Summed area *D* can be computed by four array references to the integral image.

5.2.4 Feature scaling

Since the classifier is applied on windows with different sizes, the technique used in [112] has been adopted: scaling the features, not the image. The integral image computes the sum of any image rectangle with arbitrary size in a constant time, so computing the feature values in different sized windows has always the same cost.

In this way, a 3×3 feature for the standard 12×24 window size (canonical window) is scaled to 6×6 if the current window is 24×48. From now on, when talking about a $w \times h$ pixels feature, the reader must notice that this measure is referred to the minimum window, defined as 12×24 pixels for CER–01 database, so when classifying a 60×120 pixels window, the feature is scaled to $5w \times 5h$ size (Fig. 5.7).

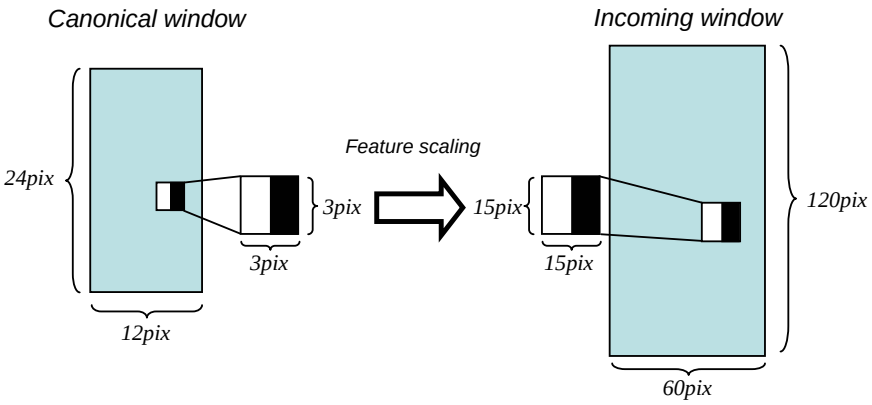


Figure 5.7: Feature scaling. In order to apply the features to an incoming window, the original feature size and position (which is always noted for a canonical window of 12×24) is scaled proportionally to the new window.

5.2.5 Tests

Fig. 5.8 illustrates the performance of Simple and Extended Haar sets. The Extended set works better than the Simple set in almost all the curve. For a 0.01 *False Positive Rate* (FPR), which represents one negative labeled as positive for one hundred samples, the improvement in correct detections is about 4%.

Fig. 5.9 shows the 16 first chosen features by the Simple Haar set. Feature 1 is very illustrative, since it corresponds to the head region, which classifies positives with a confidence of 0.70.

Finally, Fig. 5.10 shows the sample distribution according to the final Real AdaBoost rule when using 100 features in one experiment. It can be seen how positive samples tend to have positive classification values, and negatives get negative values. When plotting the ROC, the decision threshold is moved along the x axis, hence increasing the number *False Negative Rate* (FNR) but decreasing the *False Positive Rate* (FPR). Fig. 5.11 shows some of the pedestrians with higher value (best classified pedestrians). The best classified ones seem to be front/rear pedestrians with non–textured backgrounds.

5.2.6 The importance of global features

In [30], Dalal & Triggs present a new feature set to perform pedestrian detection. This new feature—discused in 5.7—works on a local fashion by patching the image with small square filters, and extracting useful gradient information. In their work, they present several performance plots where the new feature is is compared with some other features, for instance Haar Wavelets, but always using small and square features like *Papageorgiou et al.* did [79], not global multisize as it is used in our work.

In this section it is exposed why applying Haar Wavelets in such small local features can drastically reduce their performance, thus not exploiting all their strength. Fig. 5.12 illustrates this behavior. By taking again a 0.01 FPR it can be seen that the detection rate decreases in more than 10% when fixing the size of the patches to 2×2 and 3×3 (with displacement $\Delta = 1$)[1].

It is clear that using global features is very important. Fig. 5.9 prooves our hypothesis, as it can be seen that the chosen features tend to be big and not small regions.

[1] The original paper [30] used 9×9 and 12×12 features for a 64×128 pixels window, so the closest approach

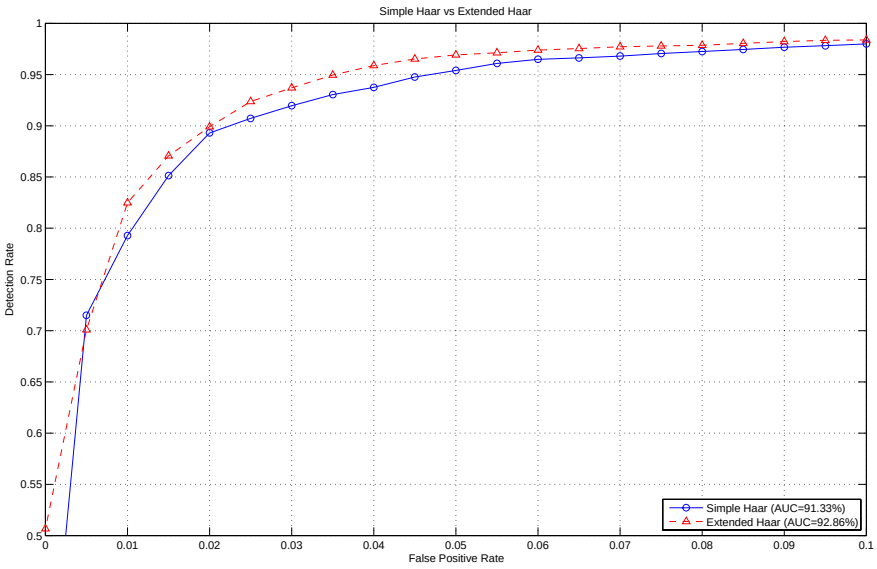


Figure 5.8: Simple Haar versus Extended Haar performance.

would be to use 2×2 and 3×3 for our windows.

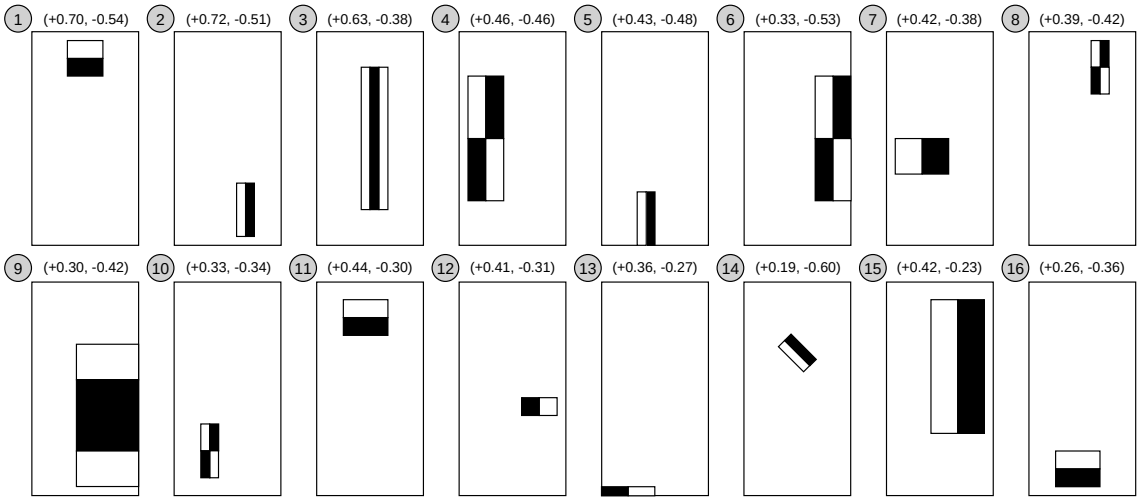


Figure 5.9: 16 first Extended Haar chosen features. The two numbers over each window represent the Real AdaBoost weak rule confidence when classifying positive and negative with the given feature. It is worth to mention that the first 8 chosen features are the same for Simple and Extended Haar sets.

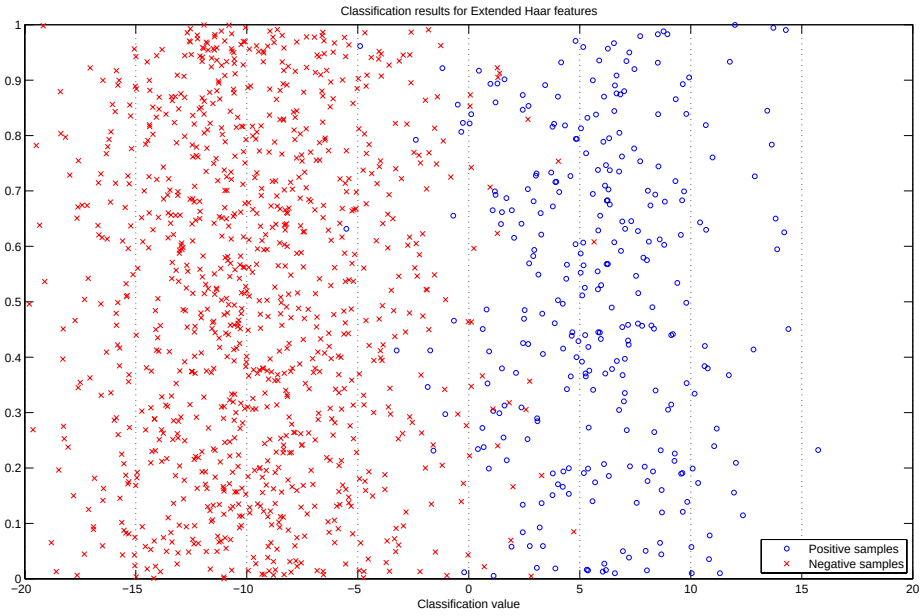


Figure 5.10: Classification of CER–01 samples using one Real AdaBoost strong classifier based on Extended Haar. (Samples are randomly distributed on the $y-$axis for better visualization)

Figure 5.11: Positive samples with higher classification value (contrast enhaced for better visualization).

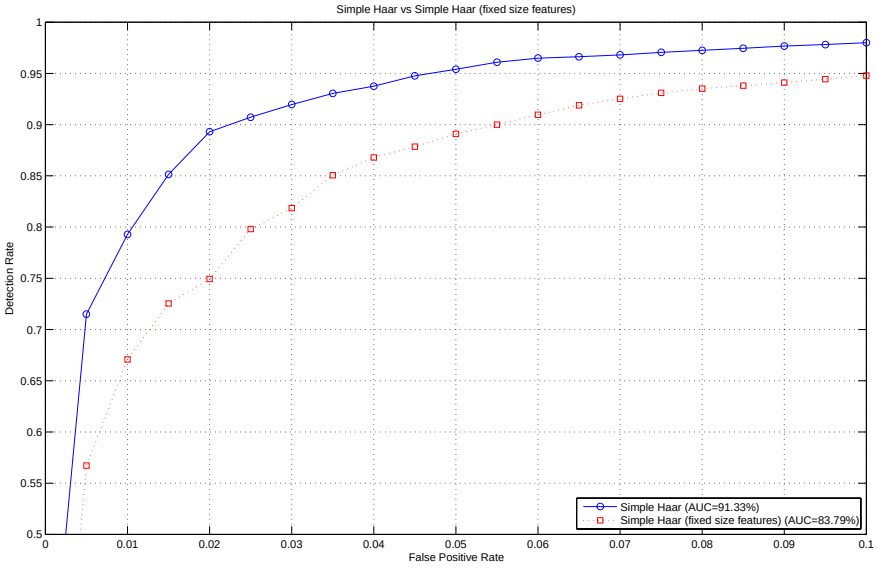


Figure 5.12: Simple Haar versus Simple Haar with fixed feature size.

5.3 Differential Invariants and local jet

Haar wavelets (Simple Haar set) have been widely used in the literature to solve classification tasks [82, 112]. Consequently, their performance at capturing structure information from a certain class has been extensively tested. Although these features are at first sight very different from the usual gradient–based features, Simple Haar features actually encode gradient changes in the image. For instance, feature *(a)* in Fig. 5.5 computes the gradient value in the horizontal axis. Hence, the feature is clearly equivalent to compute the first order partial derivative in x at a given scale defined by the width and height of the filter. Feature *(e)* in the same figure captures horizontal lines, thus approximating the second order partial derivative in y.

In this section we propose to take advantage of the values of the Simple Haar set features both to use them as the local jet until order 2 and to calculate several differential invariants [60]: gradient magnitude, Laplacian, isophote curvature and flowline curvature. The features needed are already calculated: I_x and I_y are the first order partial derivative in x and y axes, respectively, so they are equivalent to *(a)* and *(b)* features of the Simple Haar set. In the same manner, I_{xx} and I_{yy} are the second derivative (*(d)* and *(e)*). The cross derivative I_{xy} corresponds to the diagonal feature *(c)*. The equations of these invariants are listed below.

$$\begin{aligned}
&\text{GradientMagnitude} = \sqrt{I_x^2 + I_y^2} \\
&\text{Laplacian} = I_{xx} + I_{yy} \\
&\text{Isophote Curvature} = \frac{2I_x I_y I_{xy} - I_x^2 Iyy - I_y^2 Ixx}{(I_x^2+I_y^2)^{3/2}} \\
&\text{Flowline Curvature} = \frac{I_x I_y (I_{yy} - I_{xx}) + I_{xy}(I_x^2 - I_y^2)}{(I_x^2+I_y^2)^{3/2}}
\end{aligned} \tag{5.2}$$

In their original context [60], the features would be computed in the gaussian space–scale. In our approach, the scale of the feature is given by the size of the filters.

5.3.1 Test

Fig. 5.13 illustrates the performance of the differential invariants and the local jet. As can be seen, some improvement can be achieved (about 4% at 0.01 FPR). The performance is quite similar to Extended Haar set, but the number of features added are just 4, with no need of rotation features nor rotated integral image computation.

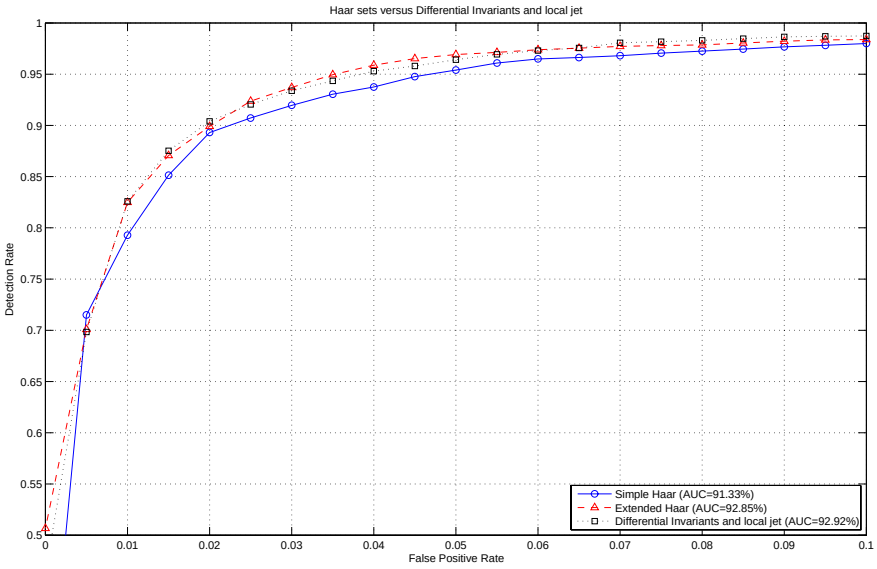


Figure 5.13: Haar wavelets sets versus Differential Invariants and local jet

5.4 Edge Orientation Histograms Features

In 2004, Levi and Weiss present a face detector based on orientation histograms [65]. These features seem also to be interesting for our work, since pedestrians usually present strong edges in zones like the legs or the trunk.

5.4.1 Preprocessing

First, edges are extracted by convolving two Sobel masks with the image I:

$$G_x(x,y) = Sobel_x * I(x,y) \tag{5.3}$$

and

$$G_y(x,y) = Sobel_y * I(x,y) \ , \tag{5.4}$$

where $Sobel_x = [-1,0,1]$ and $Sobel_y = [-1,0,1]^T$. Hence, the modulus (i.e. strength) of the edge at pixel (x,y) is defined as:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \ . \tag{5.5}$$

In the original paper, a fixed value between 80 and 110 (over 255) is used to threshold the edges image. In our implementation, an adaptive threshold that depends of the variance of the image has been used, since imposing a fixed one can supress important data in many situations.
Next, orientation of the edge is computed as follows:

$$\theta(x,y) = arctan\left(\frac{G_y(x,y)}{G_x(x,y)}\right) + \omega \ , \tag{5.6}$$

where ω is the range of all the possible orientations starts, e.g. if $\omega = -\pi/8$, the possible orientations are inside the range $(-\pi/8, 7\pi/8]$, since the orientations are unsigned ($\pi/2$ is the same as $-\pi/2$).
Then, the orientations are divided into K bins, denoting the value of the k_{th} bin as:

$$\psi_k(x,y) = \begin{cases} G(x,y) & \text{if } \theta(x,y) \in bin_k \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

Therefore, a pixel in bin $k_n \in K$ contains its gradient magnitude if and only if its orientation is inside k_n's range, otherwise is null.

At this point, every image bin contains the edge magnitude of the different orientations. Integral Image representation is also useful in this case, since the features use the sum of all the pixels of two given orientations. Thus, the sum of pixel values of bin $k \in K$ in the image region R is defined as

$$E_k(R) = \sum_{(x,y)\in R} \psi_k(x,y) \ . \tag{5.8}$$

5.4.2 Feature definition

An Edge Orientation Histogram feature (EOH) is defined as the relation between two orientations of a given region. The feature value between orientations k_1 and k_2 of region R is

$$Feature_{EOH}(x,y,w,h,k_1,k_2,R) = \frac{E_{k_1}(R) + \epsilon}{E_{k_2}(R) + \epsilon} \ , \tag{5.9}$$

where ϵ is added for smoothing purposes. If this value is above a given threshold, it can be said that orientation k_1 is dominant to orientation k_2 for R, which can be exploited as a weak hypothesis in the Real AdaBoost algorithm.

5.4.3 Bin interpolation

Our implementation uses an interesting improvement presented in [30]: bin interpolation. Instead of giving all the gradient magnitude value to a concrete bin (Eq. 5.7), the value is bilinearly interpolated between the neighbouring bin centers.

Let θ be the pixel's orientation and ω be the first angle in the orientations range. Then, the distance from the current orientation to its bin's center, $\tilde{k_i}$, is

$$d = \theta - \omega - \tilde{k_i} \ . \tag{5.10}$$

Then, the interpolation is just

$$\begin{aligned} k_i &= (1 - |d|) \cdot G(x, y) \\ k_j &= |d| \cdot G(x, y) \end{aligned} \tag{5.11}$$

where k_j corresponds to the nearest bin to θ.

Fig. 5.14 illustrates the benefits of using interpolation for Edge Orientation Histograms when using $K = 4$ bins. At 0.01 FPR, the detection rate increases in about 1%.

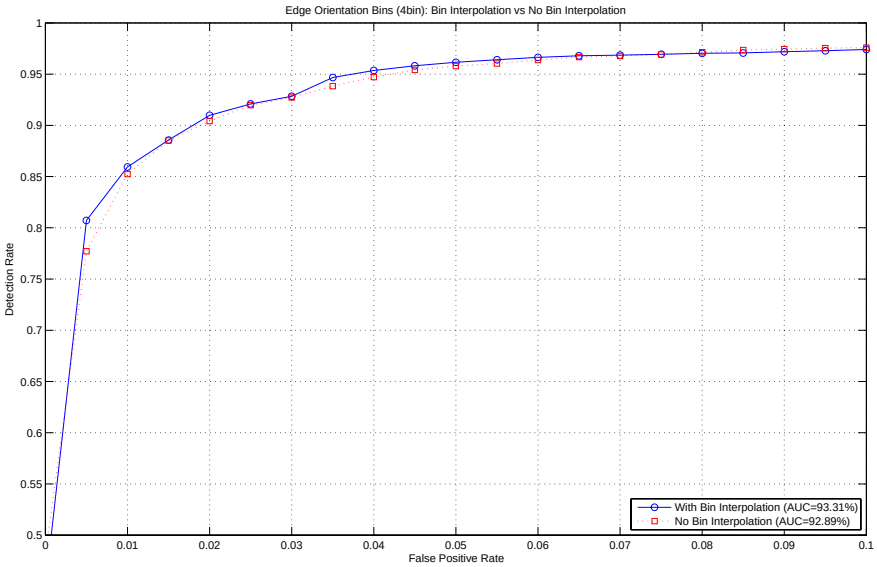


Figure 5.14: Interpolation versus No Interpolation

5.4.4 Tests

In Fig. 5.15, it can be seen how EOH features are comparable to the Haar sets, even improving the detection rate for low FPR. For example, for $FPR = 0.01$, the detection rate for EOH (4 bins) is 7% better than Simple Haar, and 4% better than the Extended Haar set. This improvement cannot be explained easily. The most intuitive idea is that orientation–based features generalize better, since they use more concrete information than just graylevel differences.

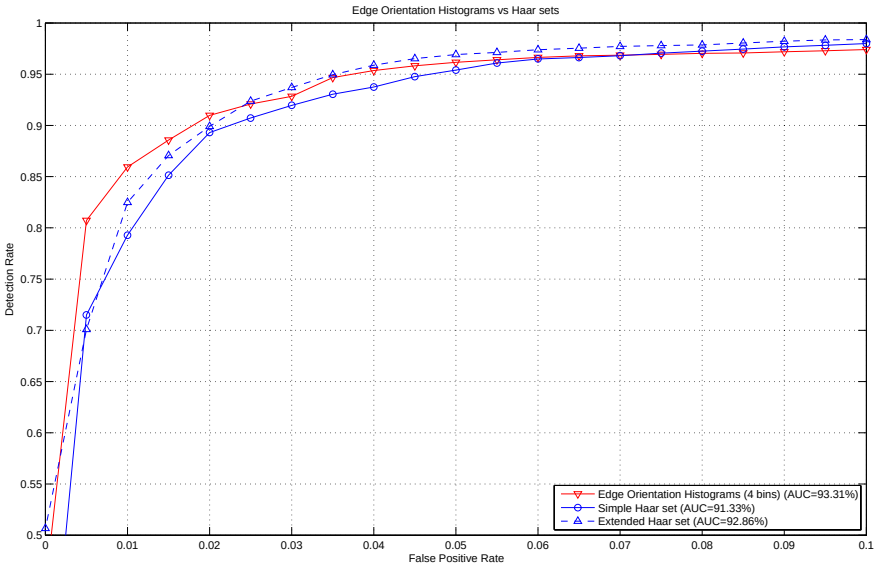


Figure 5.15: Edge Orientation Histograms versus Simple and Extended Haar sets.

Fig. 5.15 shows the resulting ROC curves for $K = 4$ (as the original paper by *Levi and Weiss* [65] proposes) and $K = 9$ (as *Dalal and Triggs* [30] propose in their HOG feature). Here, we found no explanation of the lower performance at low FPR rates when using 9 bins rather than 4. By examining the samples, the most relevant conclusion is that the best classified samples (i.e. the ones with higher positive confidence) are the ones with simple not–textured backgrounds as seen in Fig. 5.18. EOH tends to fail when classifying samples with cluttered and textured backgrounds. Here, contrary to the Haar sets best classified examples, side–viewed pedestrians have also a high confidence value, so the generalization seems to be higher.

Finally, like in the Simple Haar set, the 16 first chosen EOH features (for 4 bins) are presented (Fig. 5.17). In this case, the first one captures the leg's edges orientations, which means that the proportion between vertical and horizontal edges in the leg's area tend to be high in pedestrian samples. That's why the confidence when classifying with this feature is so high, $+0.81$ and -1.33 for positives and negatives respectively.

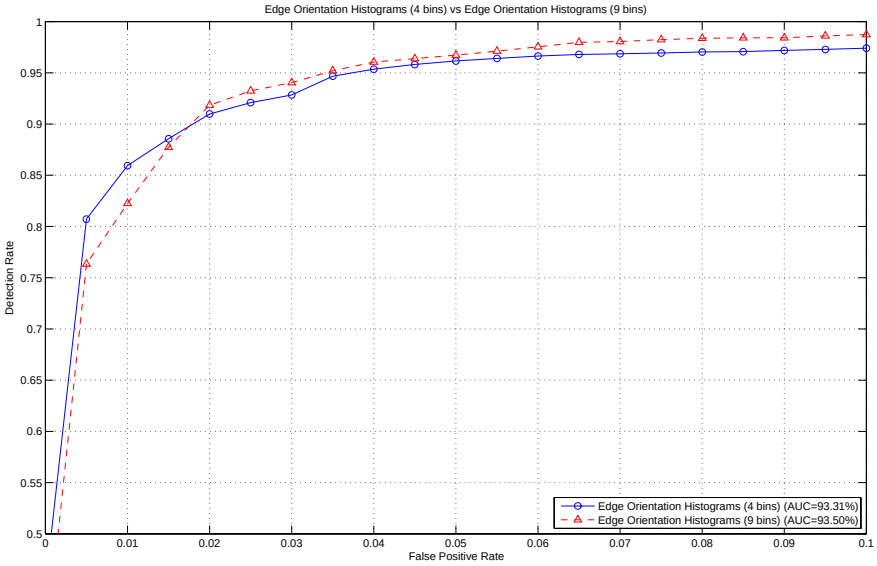


Figure 5.16: Edge Orientation Histograms: 4 bins versus 9 bins

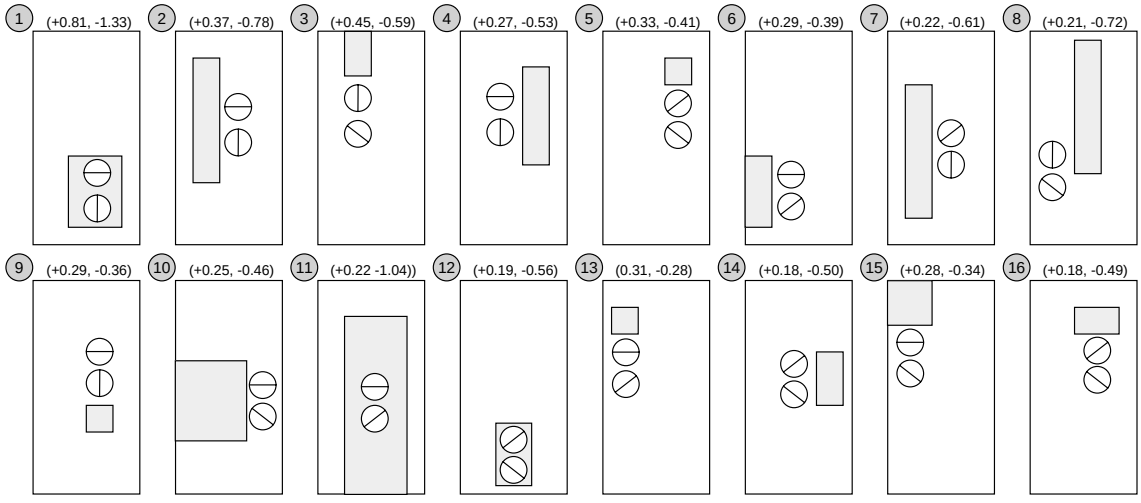


Figure 5.17: 16 first chosen features by EOH4. The grey box represent the feature location in the classification window, with the orientations used, and the two numbers over each window represent the Real AdaBoost weak rule confidence when classifying positive and negative with the given feature.

Figure 5.18: Positive samples with higher classification value (contrast enhaced for better visualization).

5.5 Structure Tensor

In this section, another feature based on gradient orientations is presented. In the search for a feature that filters out spureous pixel orientations and maintains the dominant orientations of neighborhoods, the *structure tensor* [70] has been tested (Fig. 5.19).

The target is to determine the dominant orientation of a neighborhood, and not the orientation of a concrete pixel. Then, the problem is reduced to calculating the orientation of a set of vectors, where each vector represents the angle of a single pixel (Fig. 5.19).

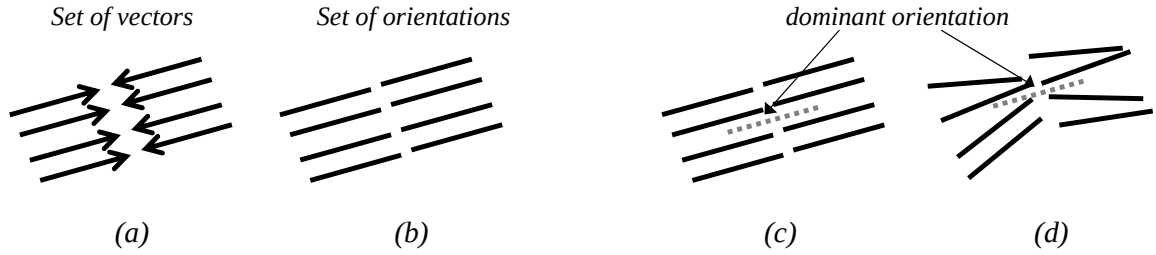


Figure 5.19: Dominant orientation given a set of vectors. *(a)* Set of vectors where the arrow head defines the direction. *(b)* Set of orientations without direction. *(c)(d)* The dashed grey line represents the dominant orientation given a set of vector orientations.

Given a set of n vectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$, each vector of dimension d, the dominant (average) orientation (ignoring their sign because just orientation is needed) is calculated by using the eigenvector corresponding to the maximum eigenvalue of the symmetric and semi–positive definite $d \times d$ matrix:

$$M = \sum_{j=1}^{n} \mathbf{u}_j \cdot u_j^t \ . \tag{5.12}$$

The matrix $\mathbf{M}$ is called structure tensor. Notice that, since this tensor is symmetric and positive definite, its eigenvalues can be sorted such that $\lambda_1 \geq \lambda_d \geq 0$ and its eigenvectors are orthonormal. The angle between the x-axis and the orientation u' is the dominant orientation θ.

Next, the analytic computation of the eigenvector corresponding to the greatest eigenvalue: $u_1 = (u_1^x, u_1^y)$ for the 2D image case.

Let us again note the image as I, which is convoluted with a gaussian G_{σ_D} (where σ_D is the *differenciation scale*) to form I_{σ_D}. Then, the first derivative in x and y axes result in $I_{\sigma_D,x}$ and $I_{\sigma_D,y}$. Finally, the structure tensor $\mathbf{S}$ is constructed with the elements $s_{11} = (I_{\sigma_D,x})^2_{\sigma_I}$, $s_{22} = (I_{\sigma_D,y})^2_{\sigma_I}$ and $s_{12} = (I_{\sigma_D,x})_{\sigma_I}(I_{\sigma_D,y})_{\sigma_I}$, where σ_I corresponds to the *integration scale* of the gaussian that is applied to $(I_{\sigma_D,x})^2$, $(I_{\sigma_D,y})^2$ and $I_{\sigma_D x} I_{\sigma_D y}$ (Fig. 5.20):

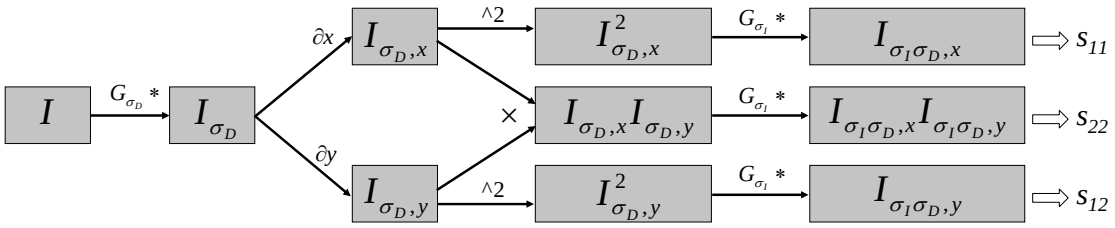


Figure 5.20: Process used to construct the structure tensor matrix given an input image.

$$\mathbf{S} = \begin{pmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{pmatrix} . \tag{5.13}$$

Supposing that $\mathbf{S}$ is not a diagonal matrix, the steps are the following:

$$\begin{aligned} \Delta_1 &= s_{11} - s_{22} \\ \Delta_2 &= s_{12} \\ \lambda_\Delta &= (\Delta_1)^2 + (\Delta_2)^2 \\ \Delta_3 &= (\Delta_1 + \sqrt{\lambda_\Delta})/\Delta_2 \\ \Delta_4 &= \sqrt{1 + (\Delta_3^2)} \\ u_1^x &= \Delta_3/\Delta_4 \\ u_1^y &= 1/\Delta_4 \end{aligned} \quad . \tag{5.14}$$

Finally, the resulting orientation is noted as $\theta = \arctan\left(\frac{u_1^y}{u_1^x}\right)$. The feature is defined as the relation between two orientations for a given area, the same approach as Edge Orientation Histograms.

5.5.1 Tests

Fig. 5.21 illustrates the performance of features based on Structure Tensor Orientation, by using $\sigma_I = 3$, $\sigma_I = 5$ and $\sigma_I = 10$, compared with Edge Orientation Histograms (σ_D was fixed to 1). Taking into consideration the 4–9 bins performance in previous tests, in this case plots where made just for the 4 bins case. As can be appreciated, although for some FPR ranges the ROC curves seem to give higher detection rates, there is not an significant improvement in all the plot when using the tensor, even when increasing its integration scale. We have some hypothesis regarding this behavior:

- First, the way we are computing the Edge Orientation features—i.e. making the relation of two orientation bins—can already provide the *dominant orientation* behavior we were looking for, specially with big sized features. Let us assume we have a feature with almost all the pixel orientations pointing a similar direction, say bin k_2, and just a few of them pointing to the perpendicular, k_0. In this case, the relation of the two orientations bins gives a high value, since the number of pixels in k_2 is much bigger than pixels in k_0. Hence, the features are computed by also taking into account the *dominant orientation* in the feature area. Of course, this is specially valid for features with big area, which are indeed the chosen ones as stated in previous figures.
- Second, variations in the integration scale can affect in a different manner to different samples in the database depending on their size. Since features are extracted from the original samples, with no resizing—remember Sect. 5.2.4—, changing σ_I on a 100×200 sample has different behavior than on a 12×24 sample, which are the extreme cases. Thus, although the tendency of the ROC seems clear with our current test, in order to perform more accurate experiments with structure tensor, methods to avoid the exposed problem should be developed, e.g. resizing all the samples to a fixed size, using a different σ_I scale depending on the sample, etc.

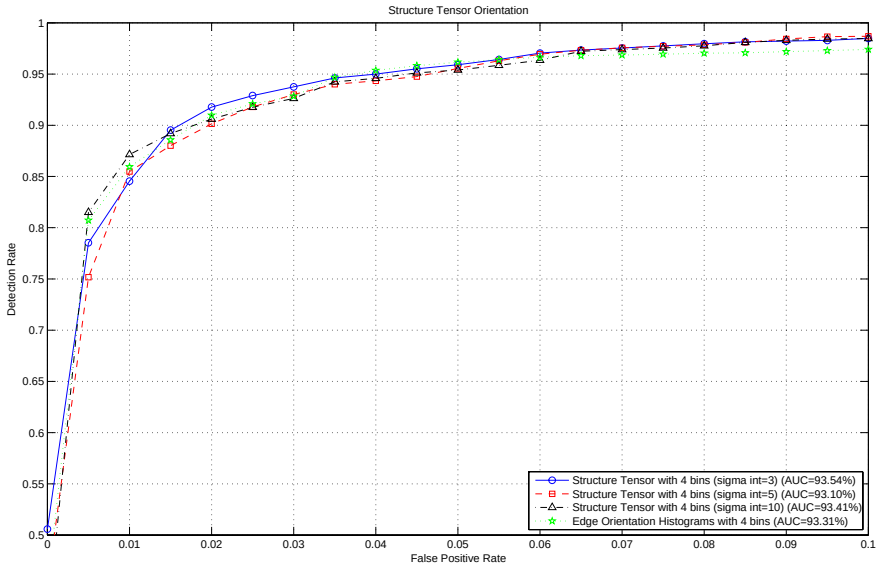


Figure 5.21: Structure Tensor Orientation performance, compared to Edge Orientation Histograms.

5.6 Combining feature sets

In this section, we propose to combine different feature sets in order to improve the results obtained by single ones alone [53]. Fig. 5.22 illustrates the great performance improvement when combining features. For $FPR = 0.01$, the combination represents an improvement in correct detection of 15% over Simple Haar and about 8% over Edge Orientation. This can be explained by checking two figures. First, Fig. 5.23 shows the chosen features in the combined set. As can be noticed, the two first features are also the first ones (which may have the highest confidence) in the Edge Orientation and Simple Haar sets respectively. Hence, the confidence of rules in the combined set is necessarily higher. This fact is revealed in a second graphic, Fig. 5.24. In this classification plot, the samples are distributed on the $x-$axis by their final classification value. As can be noticed, the margin achieved when using the combined set is higher than when using a single set (compare to the Extended Haar set classification in Fig. 5.10), i.e., the range of the distribution is wider thanks to the aforementioned higher confidence of weak rules.

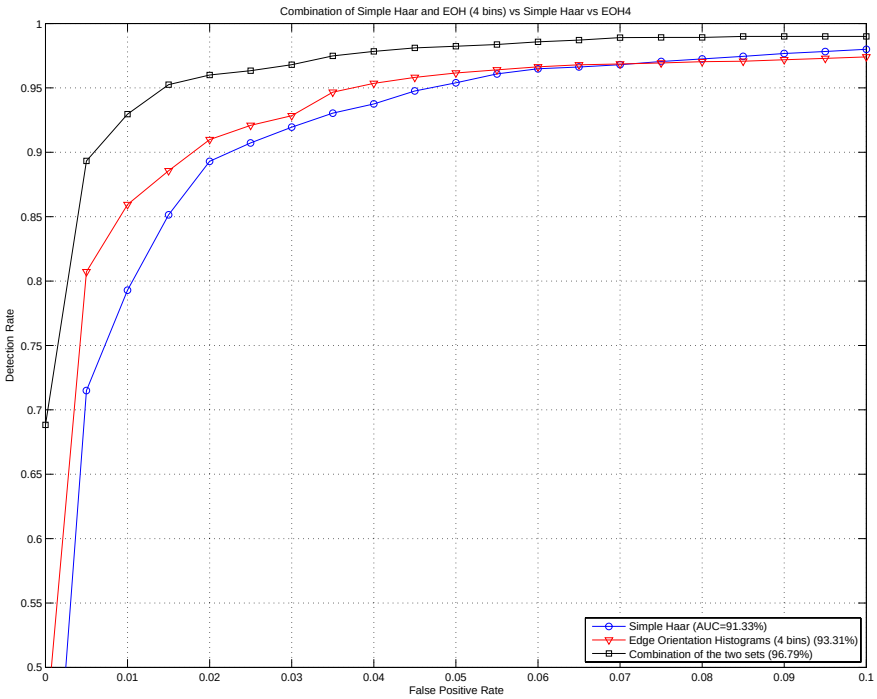


Figure 5.22: Benefits of the combination of Simple Haar and Edge Orientation Histograms (4 bins) sets.

However, not all the combinations of two sets provide a higher performance in terms of detection rates. Fig. 5.25 illustrates the ROC curve when combining Haar sets together with Edge Orientation Histograms. As can be seen, their performance is almost the same in all the cases, i.e., there is not a relevant uniform improvement (additionally, all them are around 96% of Area Under the Curve). First, the sets containing 9 bins Edge Orientations seem to suffer from the same down–performance at false False Positive Rates than when used alone (Fig. 5.16), whilst 4 bins ones perform worse when $FPR \geq 0.02$. Hence, it can be said that the behavior of a given set alone can is reflected in the

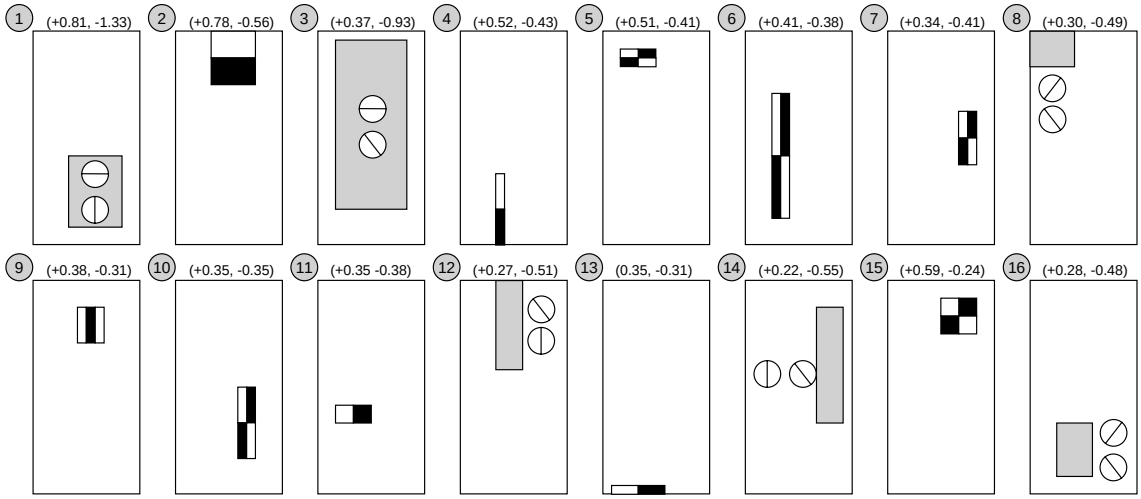


Figure 5.23: 16 first chosen features by the Combined Simple Haar and EOH (4 bins) classifier.

ROC when combined with other sets. This matter can be appreciated in the previous work [53], which presented ROC curves by averaging 2 experiments and not 4. In that paper, the peaks of single sets alone are clearly reflected when combined with others. Secondly, and now focusing on the 4 bins case, it can be seen that although Extended Haar together with EOH gives better results in the low FPR zone, the overall performance is not much higher than Simple Haar and EOH as it could be expected. Consequently, it can be stated the combinated sets' curve is not the sum of the single sets' curves alone. Contrary, the resulting curve seems to reflect the degree of complementarity between the two combined sets, i.e., if two given sets extract different information from the image, their combination would suppose a higher improvement than two sets that extract similar information when used alone. For instance, in the case of Simple Haar and EOH, the later ones can capture rotated features not present in the Simple Haar set. On the other hand, since EOH already capture orientations, the Extended Haar does not contribute with much information with its 45°, so the combined ROC curve is similar to the Simple Haar one.

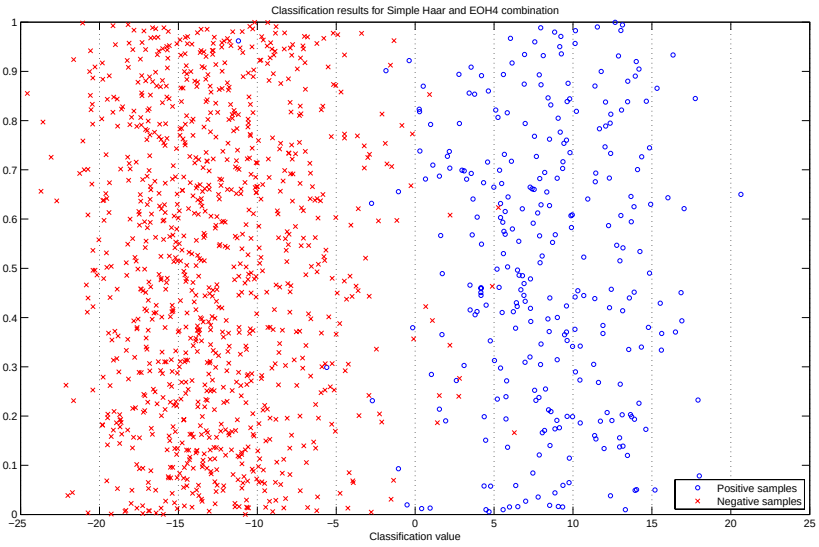


Figure 5.24: Classification of CER–01 samples using one Real AdaBoost strong classifier based on Simple Haar and EOH (4 bins) features. (Samples are randomly distributed on the $y-$axis for better visualization)

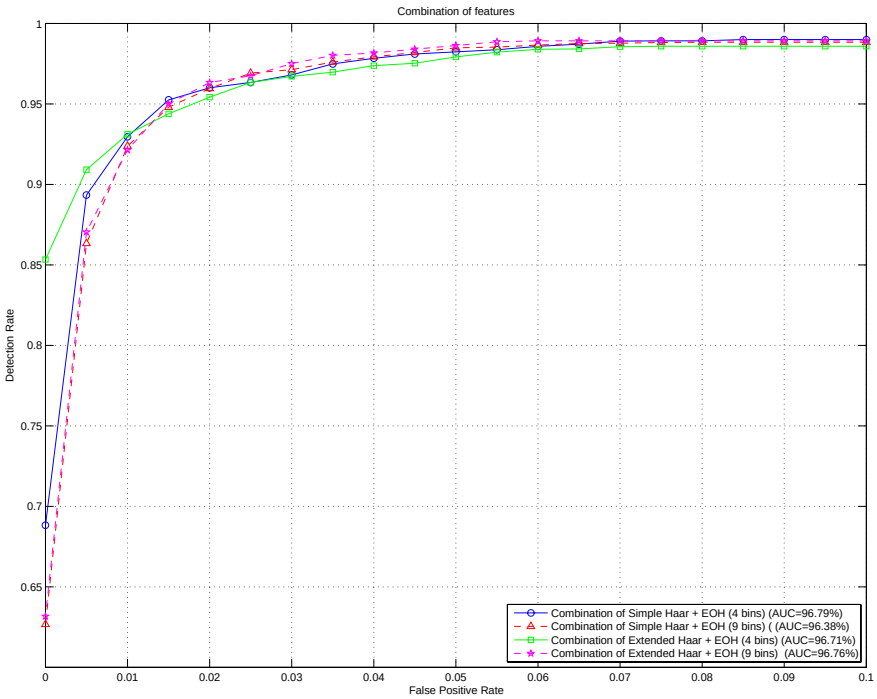


Figure 5.25: Different combinations between the two Haar sets and Edge Orientation Histograms.

5.7 SVM–based Histograms of Oriented Gradients

A recent paper by *Dalal and Triggs* [30] presents a novel feature named *Histograms of Oriented Gradients* (HOG). The feature is similar to the well–known SIFT descriptor [71]. The strong point of this feature, as the authors state, is the block normalization process.

5.7.1 Feature definition

Feature computation scheme can be seen in Fig. 5.26. No smoothing to the image is recommended according to the authors, since blurring decreases the performance of the detector. First, a Sobel mask extracts the gradient information, like in EOH. Then, pixel orientations are divided in bins and interpolated as explained in the previous EOH section. Next, the strong point of these features in front of EOH: normalization. In HOG, a feature is defined as a block, more or less like Haar and EOH, but now each block is divided in several cells, each one containing a histogram of orientations of the pixels it contains. Finally, the histograms of the cells included in the same block are attached and normalized, obtaining a vector of values that is passed to a Support Vector Machine (Appendix B) classifier.

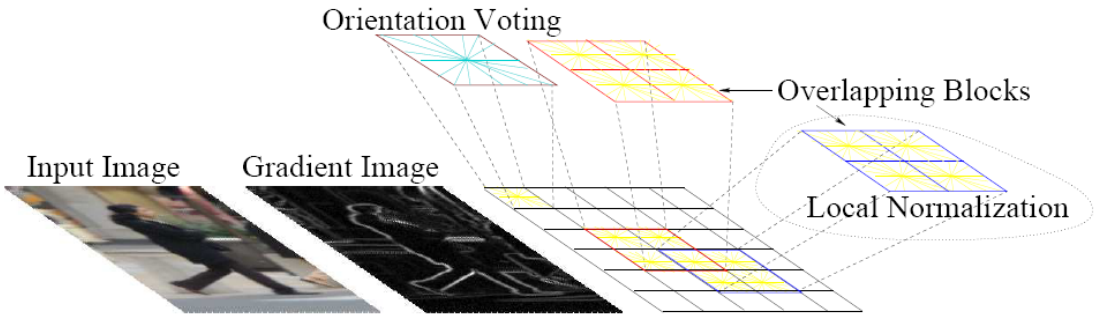


Figure 5.26: Histograms of Oriented Gradients computation [30].

5.7.2 Tuning the parameters

The original parameters presented in [30] correspond to the optimized classifier for the INRIA database, which contains samples at a higher resolution and nearer to the camera than our CER–01. Fig. 5.27 illustrates some of them. Photos are mostly taken from digital cameras at very high resolution, so no problems like blurred or very poorly illuminated pedestrians are likely to appear.

In our case, to be fair with these features, we have selected the best parameters of HOG features for our CER–01 database, fixing some others like cells per block or type of normalization. We used 2×2 cells/block, since increasing this number with our small width samples could lead to very small cells but very big blocks, hence not being loyal to the described features in [30]. We used the L2–norm for the normalization, as the authors suggest. And the overlapping is fixed to 1/2 with a stride of 1 cell as they propose. For the tests, SVMLight[2] is used with a linear kernel function with $C = 0.01$, exactly as the original paper.

[2] `http://svmlight.joachims.org`

Figure 5.27: Examples from INRIA database [30].

Fig. 5.28 illustrates the ROCs for the parameters selection. In our case, the best parameters are 9 bins with 4×4 pixels of block size. These parameters are very similar to the ones that [30] proposed for their tests with their databse. For instance, a 4×4 block size in a CER–01 sample (it is important to have in mind the aforementioned feature scaling matter), corresponds to a 2×2 pixels cell size, for a 12×24 image. If we used INRIA database sample dimensions, the 2×2 cellsize would correspond to a 10.6×10.6 pixels cellsize in INRIA, thus the feature size is similar to the proposed in the original paper. In addition, 9 bins also correspond to the selected by in their work.

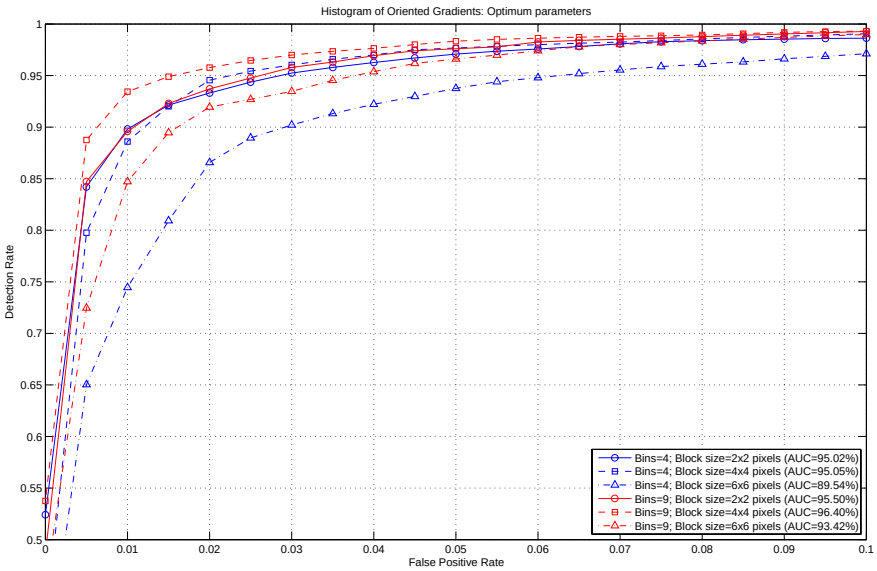


Figure 5.28: Chosing the best parameters for HOG features using SVM as the learning algorithm. Clearly using 9 bins together with 4×4 block size seems the best option.

5.7.3 Comparing HOGs with our best classifier

Having selected the best parameters for HOG features using our database, next we compare it to our best classifier. For this purpose, we get the combination of Simple Haar and Edge Orientation Histograms with 4 bins (Fig. 5.22)—which is the one with best performance—, and the Histograms of Oriented Gradients with the optimum parameters—i.e. 9 bins and 4×4 block size. The resulting ROC is presented in Fig. 5.29. Results are quite suprising, since their performance is almost the same. These curves lead us to extract some conclusions and expose new questions:

- First, if the two features can achieve the same performance, then the important point of comparison is computing time when classifying[3].
  - In the case of HOGs, the SVM classifier used 55 blocks, each one corresponding to a $36D$ vector. The time wasted in computing them is splitted in the following operations:
    - Let us assume we are using one integral for each orientation bin to speed up the process, thus we have **9 integral images** (this is done once for the whole input image).
    - To compute a HOG block, 9 access to an integral image bin are needed to get the 4 cell's orientation sums. Since there are 9 bins, $9 \times 9 = 36$ integral image access are made in total for a single block. Hence, $36\frac{access}{block} \times 55\frac{blocks}{sample}$ =**1,980 memory access per sample**.
    - Then, the values are normalized using $L2 - norm$. This implies 36 multiplications, 36 sums, one square root and 36 divisions per block, to get $\mathbf{v} \leftarrow \mathbf{v}/\sqrt{\|\mathbf{v}\|^2 + \epsilon^2}$. These operations are done for each one of the 55 blocks, so **1,980 sums, 1,980 multiplications, 1,980 divisions and one square root are needed to compute the normalization for one sample**.
    - The SVM classification works by computing the distance to the separating hyperplane, which basicly corresponds to **multiplication and sum operations**—one for each feature.
  - Combination of Simple Haar and EOH. Here, just the 100 AdaBoost-selected features are computed. In this case, the operations are the following:
    - **5 integral images are computed**: 1 for the original image and 4 for each edge orientation bin (this is done once for the whole input image).
    - To compute one Simple Haar feature, 8 array access to the integral image and a difference operation are made—except from diagonal features, where the needed access are 12 and an additional sum operation si required. With EOH, the value is always calculated by 8 access and one division operation. The worst case here seems to compute EOH features, since divisions are much more time-consuming, so $8\frac{access}{feature} \times 100 features$ =**800 access to integral images and 100 divisions** are needed, at worst, per sample.
    - Our Real AdaBoost classification works with threshold rules as weak classifiers, so the final classification needs **100 sum operations and one final comparison** to get the decision.

[3]It is clear that training our AdaBoost-based classifier takes very longer than training the SVM-based, just because we select features from a more than 100,000 instead of directly working with 1,980. However, methods to reduce the number of training features without losing performance are being tested for future use, e.g. following the technique mentioned in [121]. In addition, we are more interested in optimizing the execution speed than in the training.

With these numbers we can conclude that our approach is much faster than *Dalal*'s, despite getting similar performance rates. However, more experiments need to be done, mainly with other databases.

- Second, it is not clear whether the benefit in HOGs comes from the use of SVM or from the features themselves. Fig. 5.30 illustrates the ROC when using Real AdaBoost to classify HOG features—again with 9 bins and 4×4 blocks—, by also taking the $1,980$ features that SVM used. The plot shows that, although when using SVM version higher detection rates can be achieved, with the Real AdaBoost one we still get good detection at low FPR—which are the useful rates. On the other hand, if we use 100 HOG features, like in all our other experiments, the performance drastically decreases. Hence, the benefit seems to come from the big number of features used and not from the classification algorithm. However, again more experiments are needed to provide stronger conclusions in this matter.

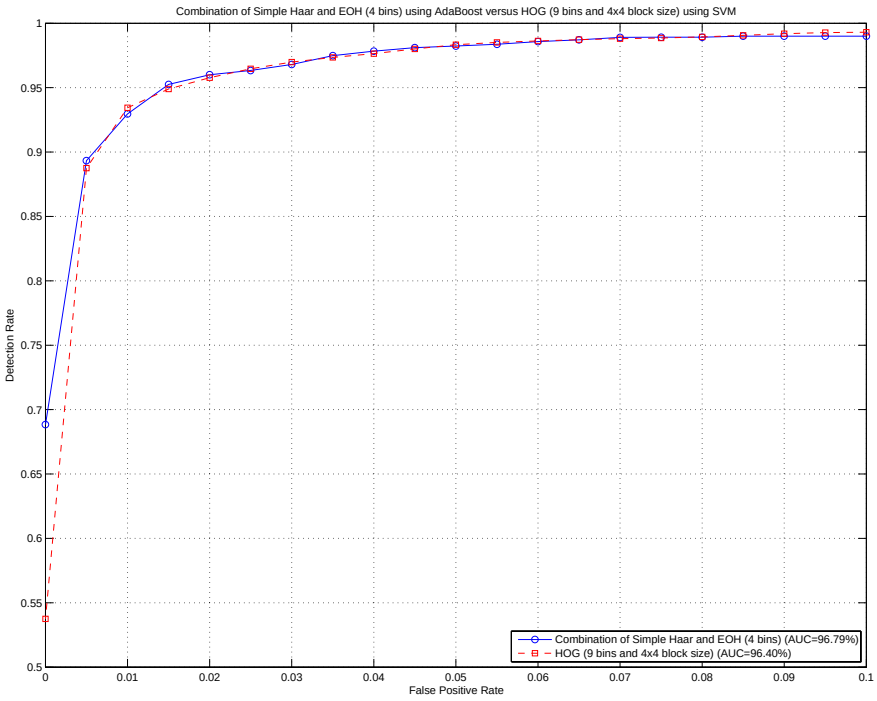


Figure 5.29: Comparison between of our best AdaBoost–based classifier (combination of Simple Haar and EOH) with the best SVM–based HOGs.

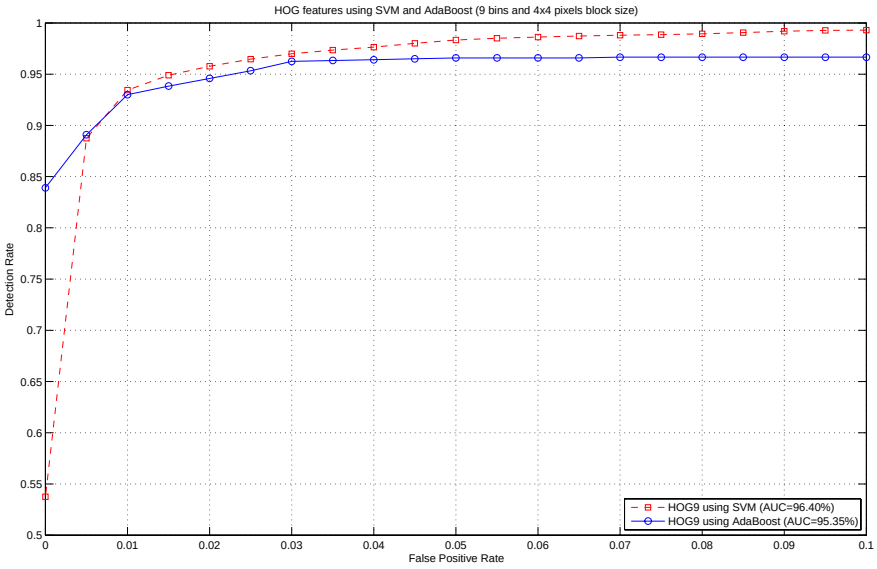


Figure 5.30: HOGs using SVM and Real AdaBoost learning algorithm and classifier.

5.8 Guidelines on Pedestrian Database construction

Having used the CER–01 database for our experiments, we can extract several conclusions to design future pedestrian databases. The principal aim when designing such databases is to reach the most representative set as possible, so the learnt model consists in the most general description of the target. Next, some guidelines are listed:

- The recording must always be focused on the application. In our case, an onboard pedestrian detector requires onboard recorded images, not photographs at full resolution. In this way, we can say that the most similar is the recording setup to the testing setup, the better the results will be when detecting.
- Multisize. It is vital if one wants to detect pedestrians in a big range of distances. As seen before, the blurring effect that appears in far pedestrians can be disturbing to a model that does not take into account such pedestrians.
- Illumination. This is perhaps the most annoying effect. Automatic gain and exposure when recording provides high contrast in the image when passing under dark zones, which is vital, but on the other hand, saturation in a concrete part of the image can lead to lose contrast in other parts. In our CER–01 database there are lots of images with very poor contrast, thus internal details of pedestrian are lost. One problematic scenario appears when driving against the sun–light, because the image tends to saturate in the top regions and then the darker zones below get poorly illuminated. Another similar case appears when driving in streets where trees and furniture project shadows on possible targets, but the rest of the image is correctly illuminated. In this case, pedestrians' edges are difficult to find if their clothes are dark. These drawbacks perhaps could be solved with an improved automatic gain algorithm, or more sofisticated maxcon techniques.
- Model variability. This point is obvious: the more variability in the training, the more general the model will be. This includes the pose, clothes and size of pedestrians.
- Background. Similarly to the last point, background diversity provides a better generalization.
- Occlusions. This point depends on the target of the detector. In our case, we do not deal with strong occlusions, but if the system is required to detect such cases, it is clear that some examples of this class are needed.
- When selecting the pedestrians, commonly known as *annotation step*, the fitting box should not always maintain the same proportions, but also contain the pedestrian in the same area (being always aware of the variability of pedestrians dimensions).

And finally, some tips and further work that would be interesting to take into account, but are not so closely related to the database design.

- A very interesting (and useful) way of constructing the best database using the smallest set of examples is to clusterize the available samples in intra–classes, and then select the most representative ones from each group. This technique would reduce significantly the number of examples, thus reducing also the required training–time. In addition, samples not providing useful information—because there are very similar samples—would not be selected.

Our CER-01 database does accomplish most of the points: the recording focused on the application, since the sequences for training are taken under the same possible conditions as the testing ones; multisize is the strong point of our database; and illumination, model and background variability is already present in our database. Nevertheless, an hypothetic CER-02 database should include even more variability in points like illumination and model variability, specially under more complex weather scenes. Besides,

the number of samples should be increased to reach the numbers of the relevant pedestrian detection systems exposed in Chapt. 2.

5.9 Summary

- As we stated in [53], any Haar set combined together with an edge orientation histogram feature set improves the performance of these sets alone (Fig. 5.25).
- Global features do count. Fig. 5.12 illustrates the importance of using global features, contrary to using local small-sized Haar wavelets [30]. This issue is explored in a recent paper by *Zhu et al.* [121].
- The structure tensor does not provide substantial benefits in performance rates in comparison with EOH features, apart from its higher cost in computing time. Our hypothesis is that dominant orientation is already being taken into account when using EOH. However, experiments using just samples of similar sizes are left as future work in order to get more reliable results. Additionally, methods different from the simple orientation relation of two bins should be investigated.
- The classifier using the combination of Simple Haar and Edge Orientation Histograms achieves the same performance as *Dalal*'s Histograms of Oriented Gradients, but having a much lower computational time when applied. We show this in Fig. 5.29. However, further research must be made in this matter, mainly testing with other databases.

Chapter 6

Results

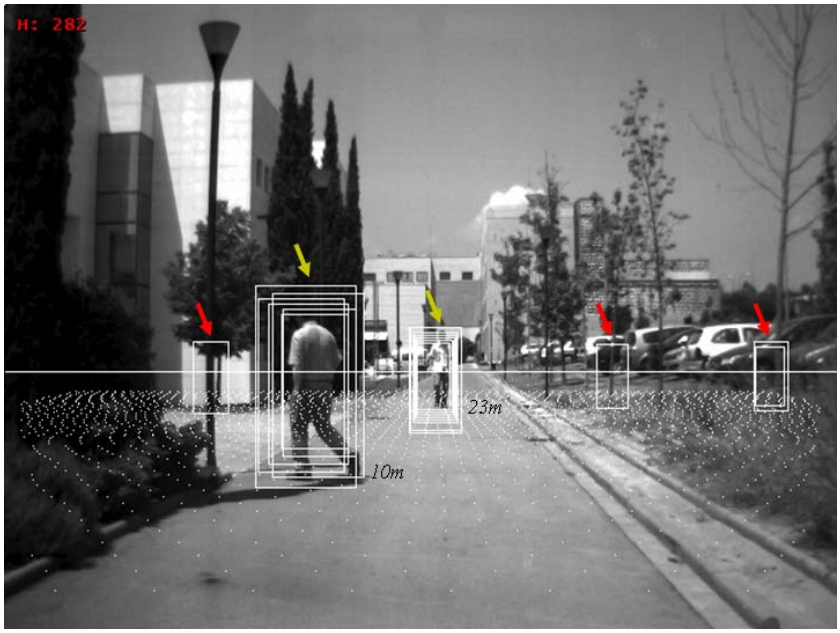
In this chapter we present the results obtained with the current system. Figures 6.1–6.5 show some snapshots of real videos processed with our system. Notice that the sequences were taken in different days than the sequences used for the classifier training (this not only implies having different scene conditions but also the acquisition system can be mounted in a different orientation and pose inside the vehicle). Here, the pitch and height of the camera is automatically estimated using the technique presented in Chapt. 4 so the ground plane is sampled by classification windows—white dots correspond to the bottom–left–corner—using 0.5m intervals. Then, windows classification has been performed using the Real AdaBoost classifier based on 100 features of the combined Simple Haar and Edge Orientation Histograms (4 bins) set of Chapt. 5.

The system has been first tested on a static scenario—car was stopped—with moving pedestrians *(a)(b)* in order to evaluate the system in a nearly controlled—if an open scenario can be defined so. The other sequences present the real case, moving host vehicle and moving pedestrians, in real urban environments. We have marked the correct detections with a green arrow, and false positives with a red one. The number next to each cluster of detections corresponds to the distance manually calculated for these windows (using the equations to calculate distances from window width in Chapt. 4).

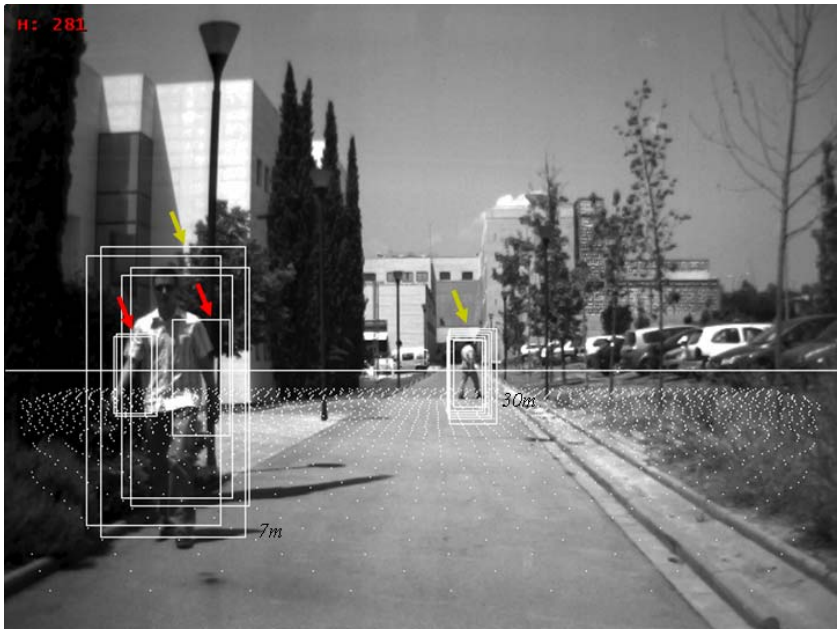
Next, we expose some issues after analyzing the results.

- The estimated camera pose, hence the horizon line, seems to be correctly estimated also in hard urban scenarios, despite of the presence of vehicles or objects in the road. The searching windows then tend to adjust correctly the posible pedestrians in the scene. Bad estimations seem to correspond to the presence of different ground planes—e.g. asphalt and pavement—, roads with not linear but curved profiles, and also roads where yaw orientation is not null *(d)*.
- As can be seen in *(a)* and *(b)*, the estimation is not the same in the two cases despite of having a static ground. This issue was already stated in Chapt. 4 summary, when talking about the estability problems. The cause of this noisy estimations is the random nature of RANSAC. Future research in this line does clearly point at implementing some kind of temporal estabilization method, for instance with a Kalman filter.
- Regarding the detection rate, the number of false negatives is very low, in the examples we have plotted dashed ellipses around them *(d),(e),(j)*. In most of the cases, missdetections come from a wrong pitch estimation which supplies not useful searching windows. In this matter, the three–horizon approach together with the pitch estimation could help, but the time spent in the classification would be multiplied by three. In this case, the error of the plane fitting could be used to adjust the other two tested horizon lines.

- False positives are usually related to vertical objects *(a)*, or windows that hold zones with similar edge orientation than pedestrians, e.g. see the lower part of the FP in *(f)*. In the same shot *(f)*, the green arrow marked with a star corresponds to a person driving a motorbike. In this case, we consider this detection as a correct positive due to the similarities with a pedestrian shape. However, this can be discused.
- As can be noticed, each pedestrian is normally classified as positive in several detection windows. This issue, known as multiple detection, should be solved in the verification/refinement module by grouping overlapping windows. This grouping must make use of the confidence and distance of the windows. In our case, we have not wasted much time in solving multiple detections, since simple heuristics do not work without a robust refiner to bound the pedestrian. Moreover, we found no mention in the reviewed works about this step, so we leave this as future work together with techniques to refine the final bounding box.
- In our case, we used one classification cascade, while other similar human–classifiers like *Zhu et al.* [121] use 18 for Haar features and 30 for HOG.
- The current system works at 2fps, spending about 250ms with the pitch estimation and other 250 for the classification. This rate is quite low for the real–time requirements we pursue. However, our efforts were not focused on time, thus these rates are likely to be improved in the future.

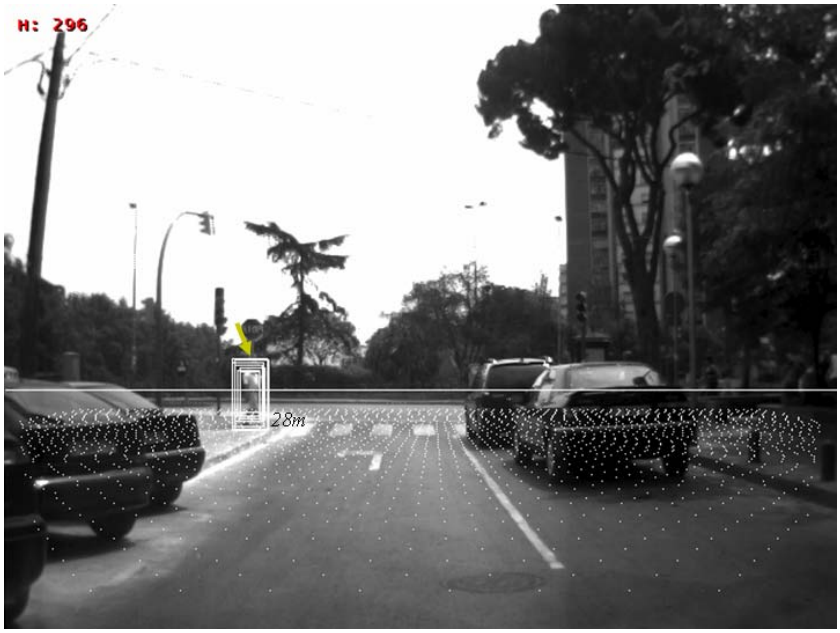


(a)

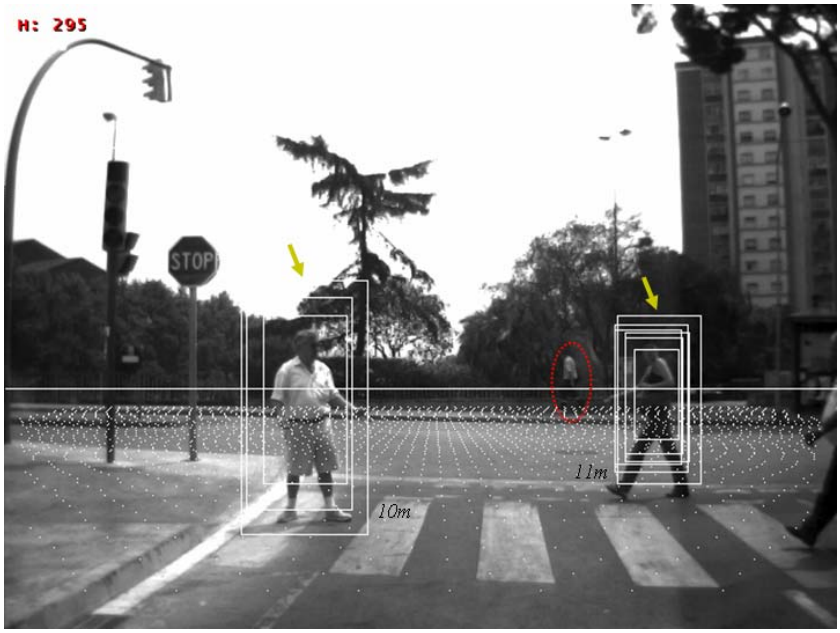


(b)

Figure 6.1: System results.



(c)



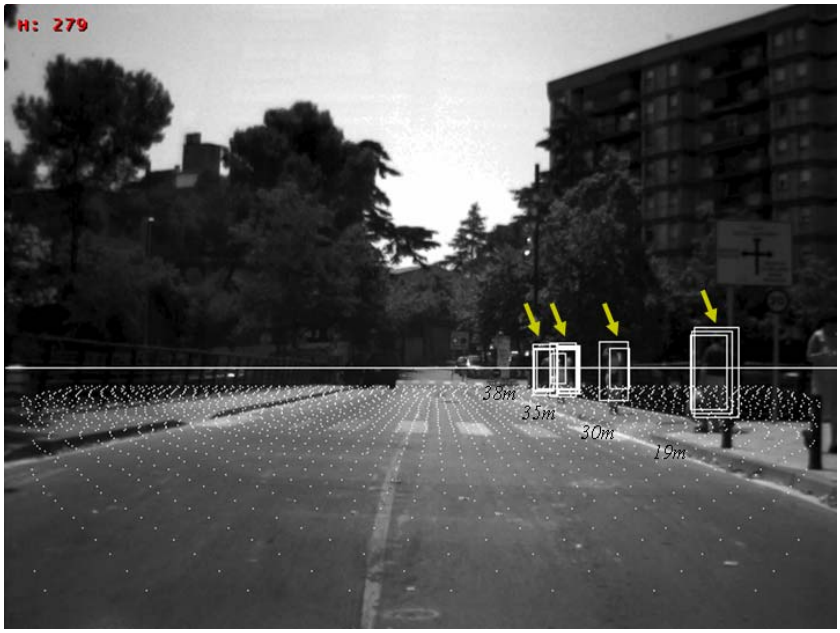
(d)

Figure 6.2: System results.

(e)

(f)

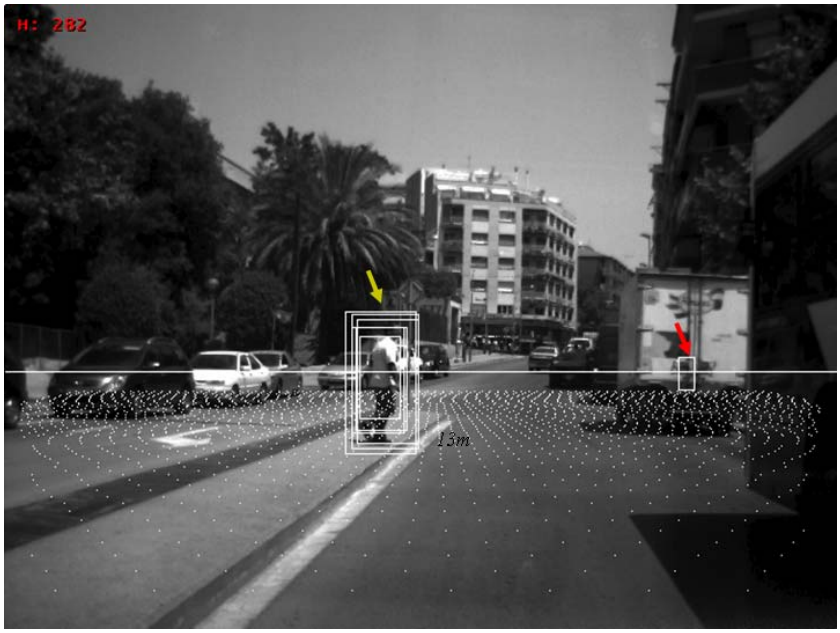
Figure 6.3: System results.



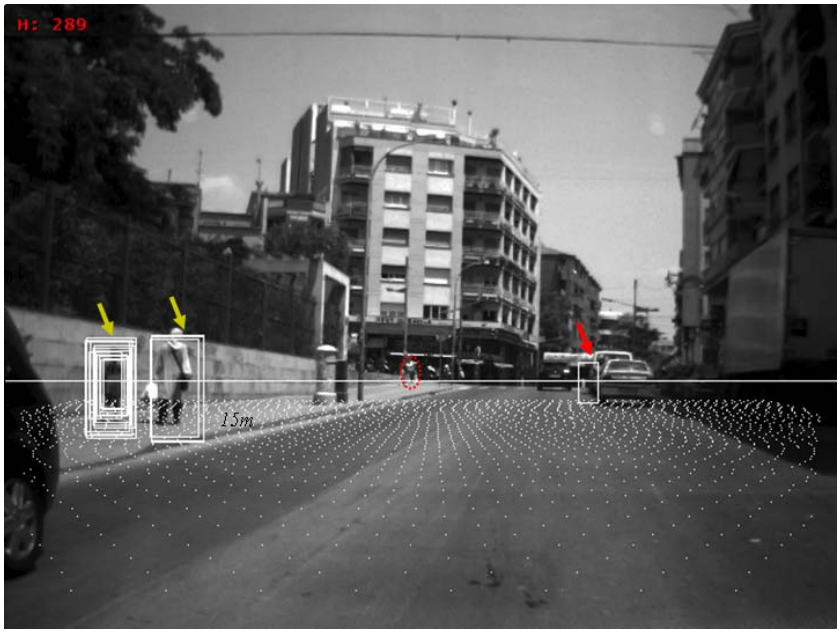
(g)

(h)

Figure 6.4: System results.



(i)



(j)

Figure 6.5: System results.

Chapter 7

Conclusions and Future Work

In this chapter we present the main conclusions of this master thesis, which mainly consist in a compilation of the chapters' summaries, together with some considerations about the global pedestrian detection system. Additionally, we enumerate some issues regarding the planned future work.

The main conclusions, in form of presented results to the objectives proposed in Chapt. 1 are the following:

- An exhaustive review of the state of the art helped us to propose a general architecture for a complete pedestrian detection system. This common architecture in modules not only represents the base to construct our future system but also helps us to compare the different proposals in the literature, highlighting the strong points and drawbacks of each one.
- We constructed a pedestrian database with our own acquisiton system, which represented mounting a stereo–camera on a vehicle and recording several hours. The encountered problems when mounting such a system have the compensation of making experiments with a good pedestrian database composed of quite hard examples in scenarios where advanced driver assistance systems are supposed to work. Such database also helped us to distinguish the different variables that affect a pedestrian, such as illumination, class variability or distance, and extract relevant information in order to design future databases.
- A stereo–based pitch–estimation technique has been presented as a method to reduce the number of image regions to explore. The proposed algorithm adjusts the horizon line dynamically by computing the pitch and height of the camera relative to the ground plane. The method has been published in a journal and in a congress proceedings:
  - A.D. Sappa, D. Gerónimo, F. Dornaika and A. López. On–board camera extrinsic parameter estimation. *IEE Electronic Letters*, 42(**13**), pp. 745–747. 2006.
  - A.D. Sappa, D. Gerónimo, F. Dornaika and A. López. Real time vehicle pose using on–board stereo–vision system. In *International Conference on Image analysis And Recognition*. Póvoa de Varzim, Portugal. 2006.

  The technique provided good results, i.e., there is a significant reduction in number of searching windows when used (from 10^8 of a full scan or 3×10^6 of [82] to just $2,000$ with our current scanning parameters), without the usual constraints applied to the same problem. However, this technique must be improved specially in terms of computing time, estability and reliability, and study the behavior in specific complex ground profiles.

- Regarding the pedestrian model chapter, we made a study of the features that work best when constructing a classifier. We have tested, with good results, features that provided high performance in face detection, like Extended Haar or Edge Orientation Histograms. We have also implemented and tested novel features for pedestrian detection like the differential invariants and local jet or the structure tensor orientation. Besides, we made experiments using the recently proposed histograms of oriented gradients (HOG) [30] with our CER–01 database, which represents testing the feature under real ADAS conditions, not with high–resolution photographs as the original paper did.

  As a main conclusion, we can say that the combination of Haar features and Edge Orientation Histograms in our Real AdaBoost classifier provides better performance rates than with the sets alone when detecting pedestrians. In addition, this combined set reaches similar performance rates than HOGs but require less computation time, so further research with these features seems an intersting option for future work.

- We also published a paper about the current system specially focusing on the classification stage:

  – D. Gerónimo, A.D. Sappa, A. López and D. Ponsa. Pedestrian detection using AdaBoost learning of features and vehicle pitch estimation. In *Proceedings of the International Conference on Visualization, Imaging and Image Processing*. Palma de Mallorca, Spain. 2006.

Along the chapters some issues have been labeled as future work, next we enumerate them:

- Apart from the improvements mentioned for the pitch–estimation, which helps to reduce the number of searching windows, we plan to investigate other approaches in order to restrict even more the candidate windows in the foreground segmentation module. The main techniques to be also implemented would be free–space analysis and stereo–segmentation.

- Regarding the classification step, research should be focused on several aspects. First, to make a deeper study on *Dalal*'s Histograms of Oriented Gradients [30] and on the combination of Haar and Edge Orientation Histograms, so to extract the benefits of both and speed up the more time–consuming parts. Second, to test component–based approaches, in order to solve occlusions problems, and also multi–class techniques, so the different object detectors in ADAS (e.g., vehicles, road signs, etc.) could share features. Third, to design a database not only by making use of the guidelines presented in Chapt. 5, but also incorporating new optimizations such as dividing front/rear pedestrians from side-viewed ones, making clusters of similar samples, etc. Finally, implementing concepts like AdaBoost cascades would significantly improve the final detector performance by reducing the number of false positives.

- Finally, it is worth to mention that the work presented in this master thesis belongs to just two modules of our proposed architecture. Thus, other modules must be developed in any level of complexity to have the complete system. First, it is mandatory to develop a method to group multiple detections and refine the final bounding box (verification/refinement module). Second, a tracking module would help to filter out spurious false positives and improve robustness of positive detections (tracking module).

Appendix A

AdaBoost

Boosting refers to the general problem of making accurate prediction rules by combining rough and moderately inaccurate rules–of–thumb [43]. *Adaptive Boosting* algorithm, formerly known as AdaBoost, was first introduced by *Freund and Schapire* in 1995 and its idea is to construct a *strong classifier* by linearly combining several *weak classifiers*.

A.1 Algorithm description

AdaBoost's input consists of a training set, whose samples are previously labeled as positive or negative; and the output is a set of weak rules that conform the final strong rule that will classify a sample.

Initialization

Let $S = \langle (x_i, y_i) \rangle_{i=1,\dots,m}$ be the training set, where $x_i \in \mathcal{X}$, being $\mathcal{X}$ the samples space, $y_i \in \mathcal{Y}$, being $\mathcal{Y} = \{-1, +1\}$ the labels space, and m the number of samples. $\mathcal{F}$ is the set of available features to build the classifier.

Each sample has its own weight, in order to boost the learning of samples that are missclassified. Samples are initialized with different values depending on their label, i.e. $w_i = \frac{1}{2\#p}$ if $y_i = +1$ and $w_i = \frac{1}{2\#n}$ if $y_i = -1$, where $\#p$ is the number of positives, $\#n$ the number of negatives, hence $m = \#p + \#n$.

Loop

Each iteration t of the algorithm provides a new feature, specifically the one that best classifies the samples according to their weights in the current iteration. The number of iterations, T, can be fixed at the beginning or make it depend on the pursued error[1].

[1] In our current implementation, T is a fixed consant.

For $t = 1, ..., T$

- Normalize weights of the samples so the sum is 1:

$$w_{i,t} \longleftarrow \frac{w_{i,t}}{\sum_{k=1}^{m} w_{k,t}} \quad . \tag{A.1}$$

- For each feature f_j in a subset $\mathcal{F}'$, where $\mathcal{F}' \subseteq \mathcal{F}$, generate an hypothesis h_j and classify samples in S. In general, h has the form $h : \mathcal{X} \mapsto \mathbb{R}$. The classifier h_j is applied according to the following equation:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) \leq p_j \Theta_j \\ 0 & \text{otherwise} \end{cases} \quad , \tag{A.2}$$

  where $\Theta_j \in \mathbb{R}$ is the threshold that divides positive and negative samples, and $p_j \in \{-1, +1\}$ is the parity that indicates the direction of the inequality sign, all for current feature f_j.

- Choose classifier h_j with lowest error ϵ_j as the new *weak* rule, being

$$\epsilon_j = \sum_{i=0}^{m} w_i \left| h_j(x_i) - y_i \right| \quad . \tag{A.3}$$

- Update the weights of the samples:

$$w_{i,t+1} = w_{i,t} \exp(y_i h_t(x_i)) \quad , \tag{A.4}$$

  so the weight of missclassified samples increases.

Final classifier

The final strong rule is defined as

$$H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} (log \frac{1}{\beta_t}) h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases} \quad . \tag{A.5}$$

i.e., 1 if classified positive and 0 if negative.

We present two figures in order to graphically show how AdaBoost works. Fig. A.1 illustrates a single iteration of the aforementioned loop. Once all the possible hypothesis are generated, the one with least error is chosen.

Fig. A.2 presents the main loop, where samples are re–weighted depending on their classification using the chosen hypothesis h_j. If a given sample is missclassified by the current h_j, the algorithm increases its weight so next iterations focus their efforts on classifying it correctly. On the contrary, if a sample is correctly classified, its weight for next iterations must decrease.

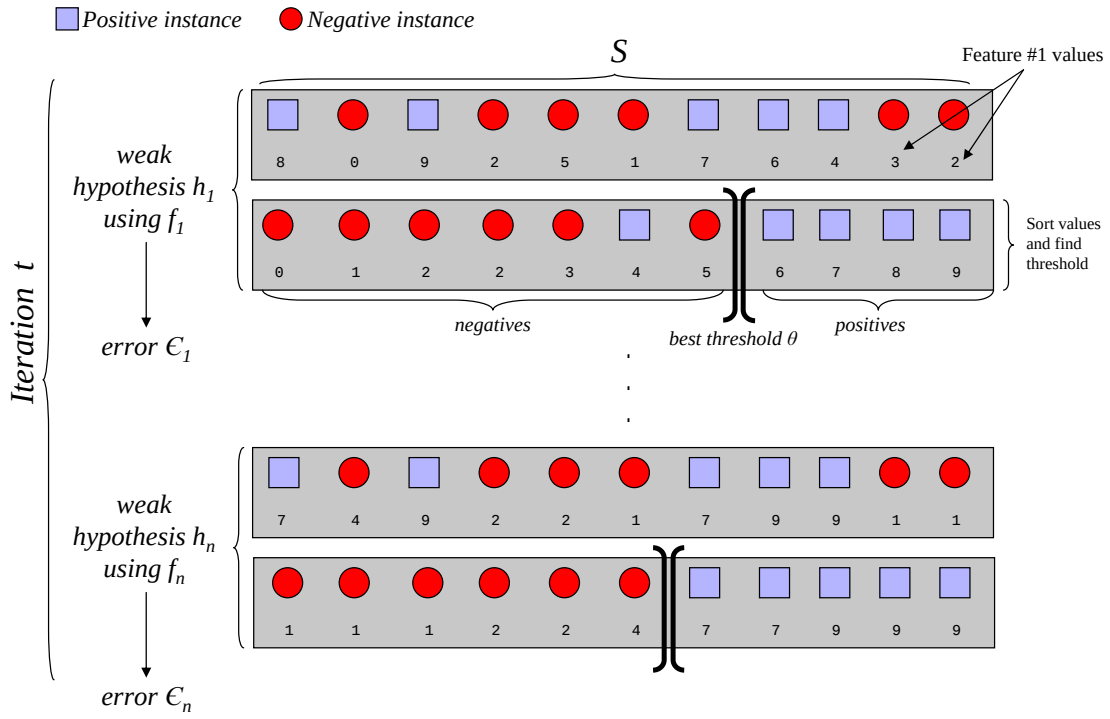


Figure A.1: *Hypothesis generation*. Positive (squares) and negative (circles) samples are sorted depending on the value of the computed feature. Then, the threshold that best divides the values is used as a parameter of the weak hypothesis.

A.2 Real AdaBoost

The learning algorithm used in our classification module corresponds to the *Real AdaBoost* version presented in [97]. The main difference from the original proposal are the associated confidence values in $\mathbb{R}$ for each weak hypothesis, contrary to the restricted range $[-1, +1]$ of the exposed implementation by Freund and Schapire.

Hence, Eq. A.2 would be replaced by the following equation:

$$h_j(x) = \begin{cases} c_\oplus & \text{if } p_j(\overline{f_j(x)} - \Theta_j) \leq 0 \\ c_\ominus & \text{otherwise} \end{cases} , \tag{A.6}$$

where $\Theta_j \in \mathbb{R}$ is the threshold that divides positive and negative samples, $p_j \in \{-1, +1\}$ is the parity that indicates the direction of the inequality sign, $\overline{f_j(x)} = f_j(x)$ if $abs_j = 0$ or $\overline{f_j(x)} = |f_j(x)|$ if $abs_j = 1$. $c_\ominus$ and $c_\ominus$ represent the confidence of the weak rule when classifying positive and negative samples respectively, and are defined as:

$$c_\oplus = \frac{1}{2} ln \left(\frac{W_+^+ + \epsilon}{W_-^+ + \epsilon} \right) \tag{A.7}$$

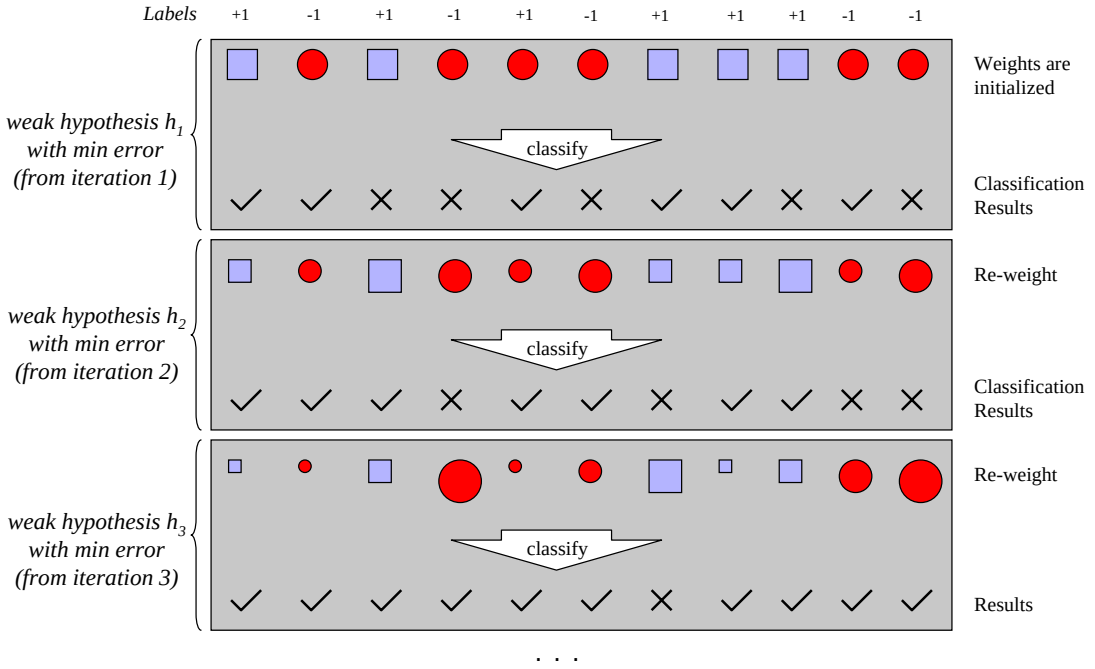


Figure A.2: *Main loop*. The results of the classification using a weak rule h_i are used to re–weight the samples.

$$c_{\ominus} = \frac{1}{2} ln \left(\frac{W_{+}^{-} + \epsilon}{W_{-}^{-} + \epsilon} \right) \quad , \tag{A.8}$$

where W_s^+ and W_s^- represent the real positive and negative samples respectively, classified by h_j as $s \in positive, negative$. For instance, W_-^+ represents the weight a positive sample classified as negative (i.e. a false negative). ϵ is added to the factors in order to smooth the results (avoid infinite magnitude in case W_-^s or W_+^s are very small or zero).

Then, the weak rule (replacing Eq. A.3) must be now defined as:

$$\epsilon_j = \sum_{i=0}^{m} w_i \left| sign(h_j(x_i)) - y_i \right| \quad . \tag{A.9}$$

Finally, the final strong classifier in Real AdaBoost is reduced to the rule:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} h_t(x) \right) \quad , \tag{A.10}$$

so if $\text{sign}(H(x)) > 0$, the sample will be classified as pedestrian, and as non–pedestrian in any other case. In addition, $|H(x)|$ represents the confidence of the classification.

A.3 Theoretical basis

Since 1997, AdaBoost has been extensively used in classification problems. The great advantage of AdaBoost over other classification algorithms is the low error achieved. Generally, as the complexity of a classifier increases, its generalization ability tends to degrade. In this case, it is said that the complexity of the classifier is so big that the classifier overfits the data. However, contrary to other algorithms, if the complexity of an AdaBoost classifier is increased, the error rate on the training set maintains its generalization properties. And surprisingly, when classifying the testing set, AdaBoost can even perform better than with the training set.

In 1998, *Schapire, Freund, Barlett and Lee* propose *margin theory* [96] to explain the aforementioned behaviour. The classification margin[2] for an example is defined as the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label. Hence, the bigger the margin is, the more confident the classification will be.

A recent work by *Rudin, Daubechies and Schapire* has demonstrated that, opposite of what everybody thought, AdaBoost does not maximize this margin to the limit [91]. We won't describe these theories, since they are out of the scope of this master thesis. Nevertheless, loads of works present in the literature make relevant that although AdaBoost does not converge to the maximum possible margin [91], but it empirically produces maximum margin classifiers with low generalization errors.

A.4 The Cascade Scheme

In [112], Viola and Jones define the *Attentional Cascade*. When several strong classifiers are available, the system can be trained so that each of them represents a level of a decision cascade that discards false positive samples missclassified by previous levels.

This scheme works like follows (see Fig. A.3.). First, all the available positive and negative samples are fed to the first level of the cascade, which makes a classification using its strong rule. This rule classifies some samples as negatives, which are directly discarded, and some others as positives, which are passed to the next level of the cascade, if there is one. The next level acts in the same way, thus reducing the false positives rate of the global classifier. One interesting approach when building the cascade is the capacity to train new negative samples in order to be attached as new levels. Note that positive samples cannot be added in new levels, so it is important to include all them at the beginning.

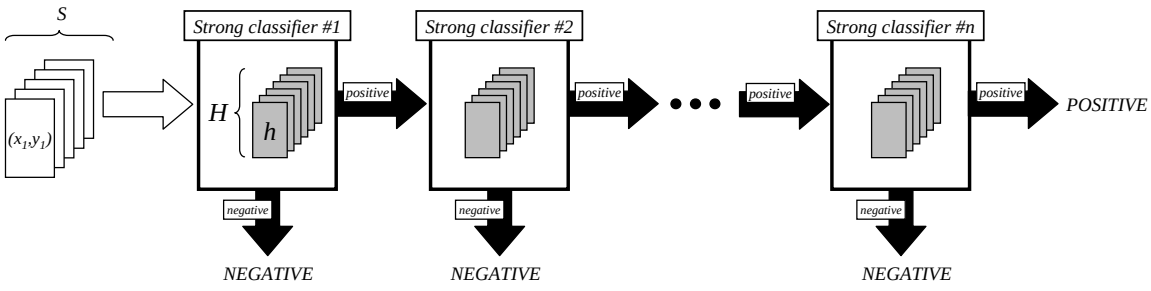


Figure A.3: *Cascade scheme.* Each level discards negative classified samples, and passes the positive classified ones to the next level.

[2]Note that this margin has no relation to the margin defined in Support Vector Machines.

Appendix B

Support Vector Machines

Support Vector Machines (SVM) were invented by Vladimir Vapnik in AT&T Bell Labs in the 90s. The goal of SVM—like any other learning tastk—is to, given a finite amount of training data, find the best rule that generalizes by balancing the accuracy on a particular training set and the ability of the machine to learn any training set without error.

The following explanation will mainly follow the tutorials by *Burges* [22] and *Chen et al.* [24], and just focuses on the Linear SVM case. For more indepth information refer to the mentioned papers.

B.1 Algorithm description

Let $S = \langle(\mathbf{x}_i, y_i)\rangle_{i=1,\dots,m}$ be the training set, where $x_i \in \mathcal{X}$, being $\mathcal{X}$ the samples space, $y_i \in \mathcal{Y}$, being $\mathcal{Y} = \{-1, +1\}$ the labels space, and m the number of samples.

Suppose there exists some *separating hyperplane* that divides positive and negative samples (Fig. B.1 *(a)*). The points $\mathbf{x}$ that lie on the hyperplane satisfy $\mathbf{w}\cdot\mathbf{x}+b = 0$, where w is normal to the hyperplane, $|b| / \|\mathbf{w}\|$ is the perpendicular distance from the hyperplane to the origin, and w is the Euclidean norm of $\mathbf{w}$. Let d_+ and d_- be the shortest distance from the separating hyperplane to the closest positive and negative examples respectively. Hence, it can be supposed that all the training data satisfies the two equations

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1 \tag{B.1}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1 \quad , \tag{B.2}$$

which can be combined into

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i \quad . \tag{B.3}$$

Then, by considering the Eq. B.1, it can be seen that points lying on the hyperplane H_1 accomplish $\mathbf{x}_i \cdot \mathbf{w} + b = 1$, where $\mathbf{w}$ is its normal and $|1 - b| / \|\mathbf{w}\|$ is the perpendicular distance from the origin. Similarly, Eq. B.2 defines the hyperplane $H_2 : \mathbf{x}_i \cdot \mathbf{w} + b = -1$, with the same normal $\mathbf{w}$ and perpendicular distance from the origin $|-1 - b| / \|\mathbf{w}\|$. Margin is then defined as $d_+ = d_- = 1/\|\mathbf{w}\|$ and the margin $2/\|\mathbf{w}\|$. Finally, the problem is reduced to maximizing the margin, thus minimizing $\|\mathbf{w}\|^2$.

In order to perform this minimization, the problem is formulated to use Lagrangians. We introduce positive Lagrange multipliers α_i, with $i = 1, \cdots, m$ for each of the inequality constraints B.1B.2. In this

way, the Lagrange functional

$$L_P = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot textbf w + b) + \sum_{i=1}^{m} \alpha_i \tag{B.4}$$

is minimized with respect to $\mathbf{w}$ and b and maximized with respect α_i.

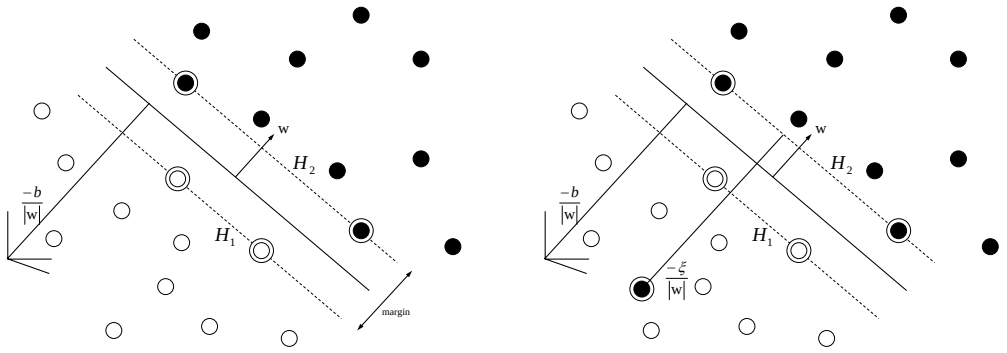
Let us now suppose the samples are not linearly–separable (Fig. B.1*(b)*). In this case, slack variable $\xi_i, i = 1, \cdots, m$ is added to the equations B.1B.2, resulting in

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - xi_i \quad \text{for } y_i = +1 \tag{B.5}$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + xi_i \quad \text{for } y_i = -1 \tag{B.6}$$

$$\xi_i \geq 0 \quad \text{for } \forall i \quad . \tag{B.7}$$

In this way, $\sum_i \xi_i$ is an upper bound on the number of training errors. Details of Lagrangian formulation given this new variable can be found in [22].



(a) *(b)*

Figure B.1: Linear separating hyperplane. Samples are classified with an hyperplane in a *(a)* separable case and *(b)* non–separable case. Support vectors are encircled.

Appendix C

Receiver Operating Characteristic (ROC) Curves

C.1 Introduction

ROC curves were originally developed in the middle of the 20th Century in order to calculate the capabilities of radar systems to distinguish radio signals contaminated with noise from actual targets. Later on, ROCs became popular in the field of medical diagnosis. For instance, ROCs were employed to compare different disease tests, and take better medical decisions. Recently, the pattern recognition field, and artificial intelligence in general, are making use of ROC curves to compare the algorithms and features used in classification problems.

Formally, a ROC curve is a graph that plots the performance of a two class classifier at different operating points. The curve illustrates the trade–off between correct and incorrect classifications while varying a decision rule, thus providing a richer description than the simple classification accuracy.

Next, we present a brief introduction to the main ROC concepts when working with a two class dichotomizer, which is the problem we are facing in this master thesis. Refer to [36, 110] in order to get a more extense explanation and deeper information about ROC curves.

C.2 Classifier performance

Formally, an instance x has a label $\ell \in \{p, n\}$ defining it as positive or negative (ℓ is known as the *actual class*). When performing the classification, the algorithm makes a prediction about the incoming instance, so x is mapped to $y \in \{p, n\}$ (hence y is the *predicted class*).

At this moment, there are four possible cases when classifying an instance:

- if $\ell = P$ and $y = p$, i.e., x is positive and it is classified as positive, it is counted as a *True Positive*,
- if $\ell = P$ and $y = n$, i.e., x is positive and it is classified as negative, it is counted as a *False Negative*,
- if $\ell = N$ and $y = n$, i.e., x is negative and it is classified as negative, it is counted as a *True Negative*,

- if $\ell = N$ and $y = p$, i.e., x is negative and it is classified as positive, it is counted as a *False Positive*.

Having this in mind, the result of classifying a set of instances can be summed up in a two–by–two matrix, known as *confusion matrix* (See Fig. C.1).

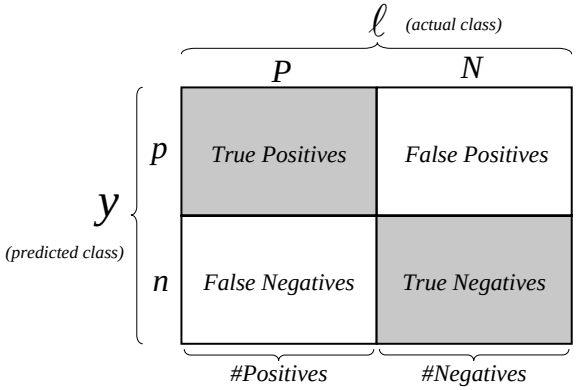


Figure C.1: The confusion Matrix illustrates the possible cases when classifying an instance. Grey cells represent correct decisions, i.e., the desirable classification, and white cells represent missclassification.

Once the matrix is constructed, the metrics needed to study the performance of the classifier can be calculated. Two main numbers are computed: *True Positive Rate* (TPR), known as *Detection Rate* or *sensitivity*, and the *False Positive Rate* (FPR), known as *False Alarm Rate*:

$$TPR = \frac{TP}{\#Positives}$$

$$FPR = \frac{FP}{\#Negatives}$$

The TPR is often referred as *hit rate* or *recall*, and measures how good the classifier labels the actual positives as positives. Other used measures are *specificity* (similar to the sensitivity, but now it measures the ability to label the actual negatives as negatives), *precision* (how likely is a positive labelled instance to be an actual positive) and *accuracy* (the proportion of correctly classified samples, both positives and negatives):

$$specificity = \frac{TN}{FP + TN} = 1 - FPR$$

$$precision = \frac{TP}{TP + FP}$$

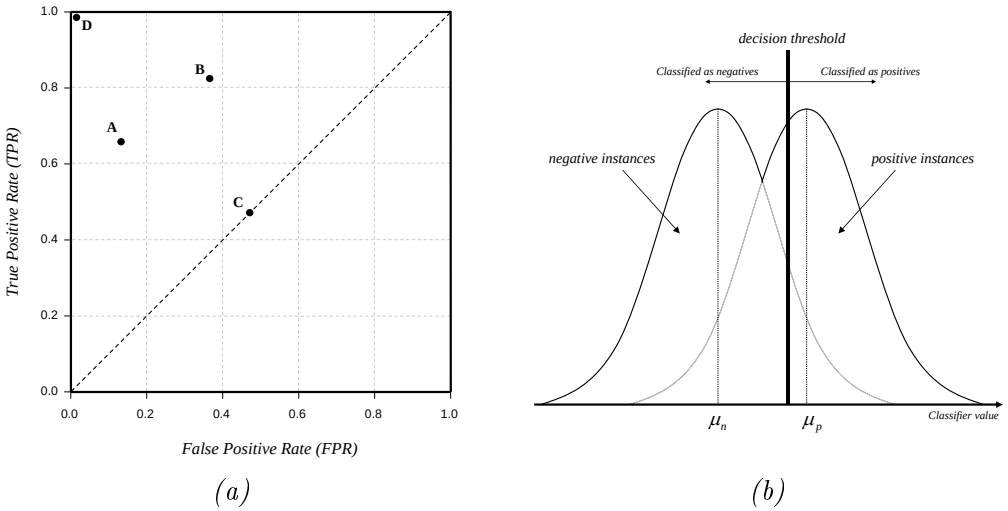
$$accuracy = \frac{TP + TN}{\#Positives + \#Negatives}$$

C.3 ROC Curves

A ROC graph illustrates the tradeoff between benefits (true positives) and costs (false negatives) in a two–dimensional plot. Graph *(a)* in Fig. C.2 shows some basic concepts of ROC space. For instance, point $(0, 0)$ represents a classifier that always labels instances as negatives. Such a classifier won't misslabel a negative, so the FPR is 0, but on the other hand there are no true positives. On the other side of the plot, point $(1, 1)$ won't miss a true positive, but its FPR is 1, since everything is labeled as positive. The dashed line in the diagonal from $(0, 0)$ to $(1, 1)$ represents the strategy of random classifiers, so it's the minimum performance expected. Classifier C is an example of them. Clasifier A illustrates a classifier with $TPR = 0.65$ and $FPR = 0.1$, thus tending to produce few false positives at the cost of not missclassifying negative instances. On the contrary, B with $TPR = 0.8$ and $FPR = 0.4$ elevates the number of correct true positives at the price of discarding less negatives. Hence, it can be clearly seen that D is the best classifier, where all positives are classified as positives, and all negatives are discarded producing a null false positive rate.

Oftenly, the classifier produces a binary result, i.e. positive or negative, so the performance of the classifier is plotted in the ROC space as a single point. In other cases, as our Real AdaBoost classifier, the result of the classifier is a score that rates the probability of the instance to pertain to one of the two classes. In this case, the decision is obtained by thresholding the posible scores, so the instances above this point are classified as positives and the ones below as negatives (or vice versa).

Fig. C.2 (b) shows this technique, exposing its relation with the aforementioned performance rates. Let us suppose a normal distribution of positive and negative instance sets with medians μ_p and μ_n respectively, and with equal variances. In order to label the instances, the classification decision is made by assuming a given threshold, which is formerly known as *operating point*. If the threshold is moved to the right of the x-axis (classification value), the proportion of false positives decreases, since the decision is more restrictive (conservative). In this case, the number of true positives will also decrease. On the other hand, if the threshold is moved to the left, the decision is less restrictive so the TPR increases, though the FPR also increases.



(a) *(b)*

Figure C.2: *(a)* A basic ROC graphs illustrating the performance of some classifiers. *(b)* Normal distributions representing the two instances sets, and the results of moving the decision threshold of the classifier.

In order to obtain the ROC curve, the decision threshold of the classifier is moved along the possible scores. Hence, the curve is plotted by interpolating the operating points of the thresholds. Fig. C.3

illustrates this.

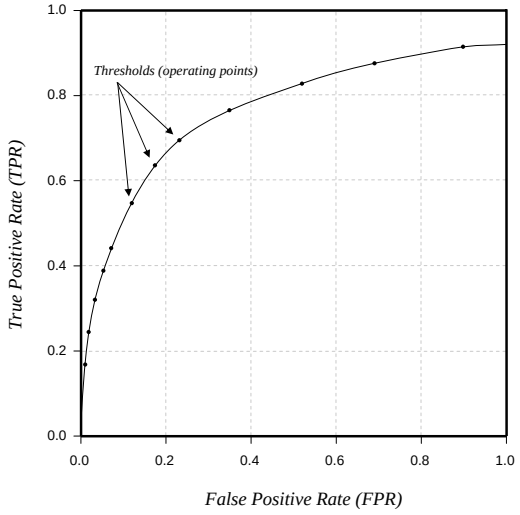


Figure C.3: ROC Curve. Operating points are interpolated in order to get the curve.

C.4 Area Under ROC

Although the two–dimensional plot is interesting to study the performance of a classifier, when two classifiers must be compared, this representation is not much illustrative. Area Under ROC (AUC) reduces this performance representation into a single scalar value. Although it is possible that a classifier with high AUC value performs worse than one with low AUC in a certain region of the ROC space, in practice the AUC is widely accepted as a good measure to compare ROCs [15]. It can be easily computed by trapezoidal integration, i.e., summing the trapezoids between the contiguous operating points of the curve:

$$AUC = \sum_i \left(\beta_{i-1}\Delta\alpha + \frac{1}{2}\Delta\beta\Delta\alpha \right) \quad , \tag{C.1}$$

where $\Delta\alpha = \alpha_i - \alpha_{i-1}$ and $\Delta\beta = \beta_i - \beta_{i-1}$. And finally dividing the result by the total possible area so the final value is normalized between 0 and 1 (see Fig. C.4).

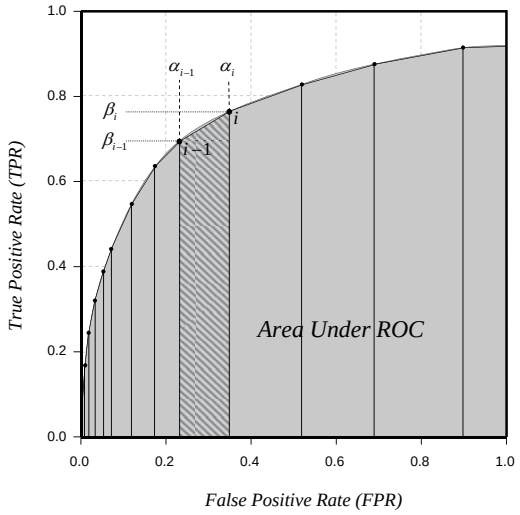


Figure C.4: Area Under ROC can be calculated by trapezoidal integration. Notice that this approach always underestimates the actual AUC, since the points are connected by lines and not by concave curves.

Appendix D

Stereo

D.1 Main concepts

Stereo is a process used to get the perception of depth and distance to objects by using two—or more—cameras. It is a passive technique, since it requires no interaction with the scene. There are other methods to recover the 3D shape or depth maps of the scene, but they are active since they project some kind of energy to the scene, like structured lighting or laser scanning.

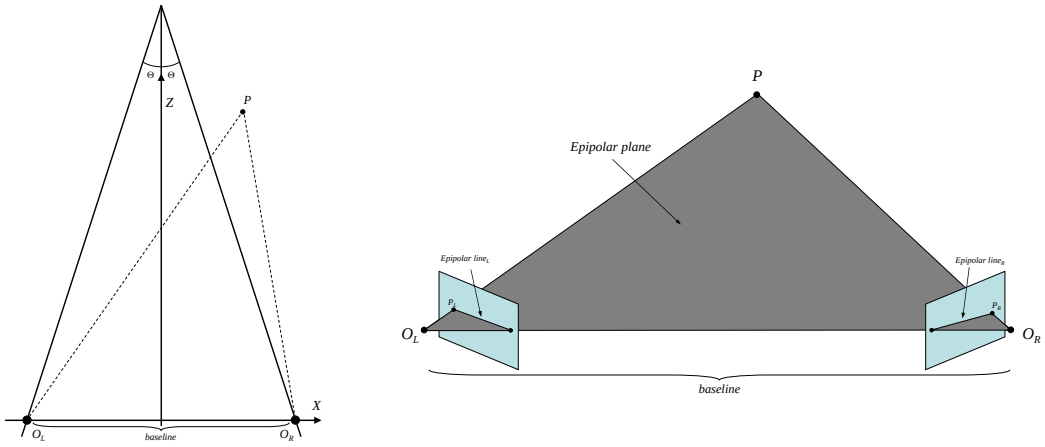
Binocular vision is based on the use of two cameras, so a previous calibration is needed to have their parameters and reach precise results. In Fig. D.1 there is a representation of the geometry concepts of stereo.

The reliability of stereo algorithms always depends on finding a good correspondence between points of left and right images. Since the correspondence is the problematic and time–consuming part of the algorithm, some constraints have to be taken:

- Epipolar line constraint. If binocular camera has the sensors at the same height from the ground and $Y-$axes are the same. By taking this assumption, epipolar lines in both images correspond to the same horizontal line in the two image planes. Hence, a point in left image must be found just in the same line of right image, so searching area for correspondence is reduced.
- Continuity of disparities is also a useful constraint. A disparity is a matching between two correspondent points P_L and P_R, and it is usually assumed that disparity variation between local neighbourhoods is smooth. However, this assumption cannot be guaranteed for all the pixels of the image, due to the different distance of objects from the camera, which makes the disparity in their borders to change quickly.
- Uniqueness constraint adds the assumption that every pixel in one image can only correspond to just one pixel in the other image. This constraint is not useful in our problem, because the background in a non controlled scenario can have several pixels with same intensity and even textures with the same structure in different places of the image.

Once some constrains are been taken, a correspondence analysis can be made to match points in both images. There are two basic approaches to make the similarity analysis between two pixels, or more generally image subwindows:

- Intensity–based. Corresponding pixels in a stereo–pair has normally the same intensity values. One of the first problems we encounter is that there are lots of possible matching pixels if we just



(a) *(b)*

Figure D.1: Stereo geometry. *(a)* Two cameras of the same focal length f, with their focal points located at O_L and O_R, at a distance b, called baseline. Parallel Y-axes and same tilts θ for both cameras are assumed in this example. *(b)* The epipolar plane is the plane formed by camera optical centers O_L and O_R, and a point P in the world. The epipolar lines (i.e., the intersection of the epipolar plane and the image planes) define the places where the point can lay in the two images (if and only if Y-axis of both cameras are parallel and are at the same height).

take into account their intensity values. That's why it is worth to define a block of pixels, i.e. a window of size 5×5 or 7×7, and then compare the similarity of them.

There are several techniques in this point, the most intuitive one is to search for an identical window (i.e. with the same intensity values in the pixels of the block) in an interesting area (restricted to the constraints we stated) of the other image. Color information can also be useful to do the block matching based on intensity. For more information about these algorithms and some refinements, we encourage to take a look at chapter 4 of [61], dedicated to *Static Stereo Analysis*, which is being used as a reference to make this section of the appendix.

- Feature–based. This is a more refined method that avoids some of the ambiguities we can find in intensity based ones. In addition, they can require less computing time in case the feature extraction is fast enough. The previously stated reference [61] exposes an analysis based on zero–crossing vectors, and another one based on color features.

We'll also mention a method based on both intensity and structure to measure similarity. In [42], Franke and Kutzbach define an encoding scheme to classify the structure of every pixel's neighbourhood (top, down, left and right nearest pixels), and then search pixels with the same structure along the same row of the corresponding image.

Once we have the matching between pixels or blocks in the two images, depth is found by triangulation. We describe the simplest case. If we assume that the camara coordinate systems are only translated parallel to each other, i.e. cameras' focus do not converge in a point, the equation to find the Z is simple:

$$Z = \frac{f \cdot b}{P_L - P_R} \ , \tag{D.1}$$

where f is the focal length, b the baseline, and P_L and P_R the mentioned projection of the world point P in the two image planes.

D.2 V–disparity representation

With the aim of extracting the road profile from a given image—Chapt. 4 exposes the importance of this data for pedestrian detection and ADAS in general—, in 2002 *Labayrade et al.* [63] propose the v–disparity representation. We think it is worth to give a brief explanation of this technique since our future works in pitch estimation should be focused on improving its results.

V–disparity makes use of a disparity map I_Δ computed from a stereo image pair (Fig. D.2 *(middle)*). Such map can be calculated via epipolar lines constraints or any other structure–matching technique previously mentioned. Then, for a row i in I_Δ, the disparity value Δ_M corresponding to point M in this row is plotted to the (Δ_M, i) position of $I_{v\Delta}$, the v–disparity image (Fig. D.2 *(right)*).

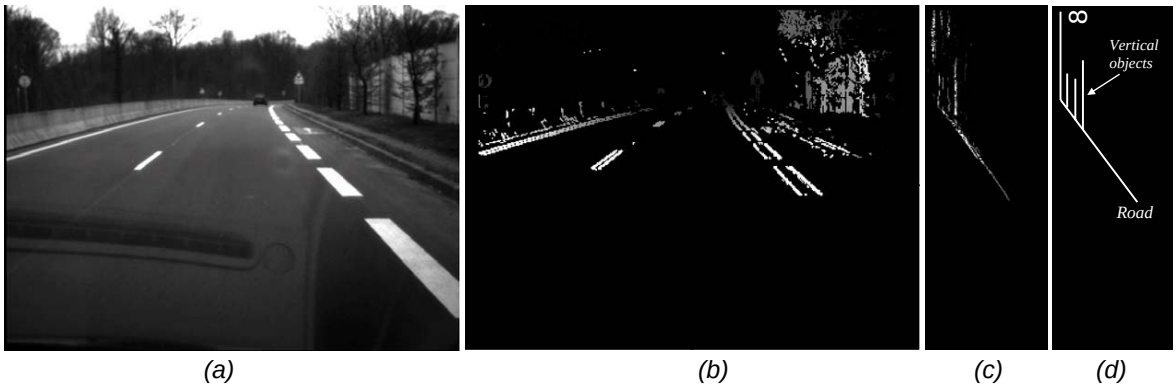


Figure D.2: *(a)* Original, *(b)* disparity and *(c) v–disparity images*. *(d)* meaning of lines in the representation.

Once $I_{v\Delta}$ is constructed, a Hough transform can be used to find the line that corresponds to the longitudinal profile of the road, and even approximating a piecewise curve in case of non–flat roads. As can be seen in Fig. D.2 *(right)*, this representation is also useful to detect obstacles in the road, in the sense that obstacle's disparity along the vertical axis is different than the road's one. Thus, they appear as vertical lines in $I_{v\Delta}$.

The authors state $40ms$ to compute $I_{v\Delta}$ and compute the tyre–road contact points with a PIV $1.4GHz$ PC using input images of 380×289. Moreover, this technique has been used in pedestrian detection systems by *Broggi et al.* [18] or *Grubb et al.* [57] with good results.

D.3 Distance computation using camera parameters

Once camera parameters pitch and height have been estimated (for instance using the technique in Chapt. 4), or fixed for a concrete scenario, the real distance from the camera to the road points in the image can be calculated by using the equations D.2 and D.3, as proposed in [88]:

$$x_I = x_0 + \frac{f^x x}{z\ \cos(\theta) + (-h + y)\sin(\theta)} \ , \tag{D.2}$$

$$y_I = y_0 + \frac{f^y((h - y)\ \cos(\theta) + z\ \sin(\theta))}{z\ \cos(\theta) + (-h + y)\sin(\theta)} \ , \tag{D.3}$$

where (x, y, z) is the 3D point to be projected, $(x_0, y0)$ is the center of the image projection, (f^x, f^y) the focal length on x and y direction, θ the pitch angle of the camera relative to the ground orientation, and h the height of the camera. Then, x_I and y_I are the image coordinates where the 3D point is projected.

In our work, a window scan along the 3D road (at $y = 0$) is made in intervals of $0.5m$ in both x and z directions (the 3D road sampled points correspond to the lower left corner of our 2D window), and then adjusting the windows sizes according to the distance to the camera using the same equations.

Appendix E

Glossary

ADAS Advanced Driver Assistance System

EOH Edge Orientation Histograms

HFOV Horizontal Field of View

HOG Histograms of Oriented Gradients

PPS Pedestrian Protection System

PSC Pedestrian Size Constraints

SVM Support Vector Machine (L: Linear, Q: Quadratic, G: Gaussian)

VFOV Vertical Field of View

Bibliography

[1] L. Andreone, F. Bellotti, and A. D. Gloria and R. Laulett. SVM–based pedestrian recognition on near–infrared images. In *Proceedings of the IEEE International Symposium on Image and Signal Processing and Analysis*, pages 274–278, Zagreb, Croatia, 2005.

[2] APALACI: Advanced Pre–crash And Longitudinal Collision Mitigation System with Pedestrian Classification Ability. `http://www.prevent-ip.org`.

[3] Annual report on traffic accident statistics 1998. Technical report, Institute for Traffic Accident Research and Data Analysis, Tokyo, Japan, 1999.

[4] P.H. Batavia and D.E. Pomerleau and C.E. Thorpe. Overtaking vehicle detection using implicit optical flow. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 729–734, Boston, USA, 1997.

[5] C. Bellotti, F. Bellotti, A. D. Gloria, and L. Andreone and M. Mariani. Developing a near infrared based night vision system. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 686–691, Parma, Italy, 2004.

[6] M. Bertozzi, A. Broggi, and A. Lasagni and M. Del Rose. Infrared stereo vision–based pedestrian detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 23–28, Las Vegas, USA, 2005.

[7] M. Bertozzi, A. Broggi, and A. Fascioli and P. Lombardi. Vision–based pedestrian detection: Will ants help? In *Proceedings of the IEEE Intelligent Vehicles Symposium*, volume 1, pages 1–7, Paris, France, 2002.

[8] M. Bertozzi, A. Broggi, M. Carletti, A. Fascioli, T. Graf, and P. Grisleri and M–M. Meinecke. Ir pedestrian detection for advanced driver assistance systems. In *Proceedings of the Pattern Recognition Symposium*, volume 2781, pages 582–590, Magdeburg, Germany, 2003.

[9] M. Bertozzi, A. Broggi, R. Chapuis, F. Chausse, and A. Fascioli and A. Tibaldi. Shape–based pedestrian detection and localization. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 328–333, Shangai, China, 2003.

[10] M. Bertozzi, A. Broggi, A. Fascioli, and T. Graf and M. Meinecke. Pedestrian detection for driver assistance using multiresolution infrared vision. *IEEE Transactions on Vehicular Technology*, 53(6):1666–1678, 2004.

[11] M. Bertozzi, A. Broggi, A. Fascioli, A. Tibaldi, and R. Chapuis and F. Chausse. Pedestrian localization and tracking system with Kalman filtering. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 584–589, Parma, Italy, 2004.

[12] M. Bertozzi, A. Broggi, P. Grisleri, and T. Graf and M–M. Meinecke. Pedestrian detection in infrared images. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 662–667, Columbus, USA, 2003.

[13] E. Binelli, A. Broggi, A. Fascioli, S. Ghidoni, P. Grisleri, and T. Graf and M–M. Meinecke. A modular tracking system for far infrared pedestrian recognition. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 758–763, Las Vegas, USA, 2005.

[14] R. Bishop. *Intelligent Vehicle Technologies and Trends*. Artech House, Inc., 2005.

[15] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[16] A. Broggi and M. Bertozzi and A. Fascioli. Self–calibration of a stereo vision system for automotive applications. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3698–3703, Seoul, Korea, 2001.

[17] A. Broggi, M. Bertozzi, and A. Fascioli and M. Sechi. Shape–based pedestrian detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Dearborn, USA, 2000.

[18] A. Broggi, A. Fascioli, I. Fedriga, and A. Tibaldi and M. Del Rose. Stereo–based preprocessing for human shape localization in unstructured environments. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 410–415, Columbus, USA, 2003.

[19] A. Broggi, A. Fascioli, P. Grisleri, and T. Graf and M–M. Meinecke. Model–based validation approaches and matching techniques for automotive vision based pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, San Diego, USA, 2005.

[20] A. Broggi, R.I. Fedriga, A. Tagliati, and T. Graf and M. Meinecke. Pedestrian detection on a moving vehicle: an investigation about near infra–red images. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 431–436, Tokyo, Japan, 2006.

[21] A. Broggi, P. Grisleri, and T. Graf and M–M. Meinecke. A software video stabilization system for automotive oriented applications. In *Proceedings of the IEEE Vehicular Technology Conference*, pages 2760–2764, Stockholm, Sweden, 2005.

[22] C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2:121–167, 1998.

[23] C-Y. Chan and F. Bu. Literature review of pedestrian detection technologies and sensor survey. Technical report, Institute of Transportation Studies, University of California at Berkeley, 2005.

[24] P.H. Chen and C.J. Lin and B. Scholkopf. A tutorial on ν–support vector machines. *Applied Stochastic Models in Business and Industry*, 21(2):111–136, 2005.

[25] P. Coulombeau and C. Laurgeau. Vehicle yaw, pitch, roll and 3D lane shape recovery in vision. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 619–625, Versailles, France, 2002.

[26] R. Cutler and L. Davis. Real–time periodic motion detection, analysis, and applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 23–25, Ft. Collins, USA, 1999.

[27] R. Cutler and L. Davis. Robust periodic motion and motion symmetry detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 615–622, Hilton Head Island, USA, 2000.

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

[29] E. Dagan, O. Mano, and G. Stein and A. Shashua. Forward collision warning with a single camera. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004.

[30] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 886–893, San Diego, CA, USA, 2005.

[31] T. Dang and C. Hoffmann. Stereo calibration in vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 268–272, Parma, Italy, 2004.

[32] Drive smart in the dark! night driving tips. Technical report, Nebraska Highway Safety Program and the Lincoln–Lancaster County Health Department, 2001. `http://nncf.unl.edu/eldercare/info/seniordriving/nightdrive.html`.

[33] Y. Fang, K. Yamada, Y. Ninomiya, and B.K.P. Horn and I. Masaki. Comparison between infrared–image–based and visible–image–based approaches for pedestrian detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 505–510, Columbus, USA, 2003.

[34] Y. Fang, K. Yamada, Y. Ninomiya, and B.K.P. Horn and I. Masaki. A shape–independent method for pedestrian detection with far–infrared images. *IEEE Transactions on Vehicular Technology*, 53(6):1679–1697, 2004.

[35] B. Fardi and U. Schuenert and G. Wanielik. Shape and motion–based pedestrian detection in infrared images: a multi sensor approach. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 18–23, Las Vegas, USA, 2005.

[36] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[37] M. Fischer and R. Bolles. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6):381–395, 1981.

[38] United Nations Economic Commission for Europe. Statistics of road traffic accidents in europe and north america, 2005.

[39] National Center for Statistics and Analysis of the National highway Traffic Safety Administration. Traffic safety facts 2003: Pedestrians, 2003. `http://www-nrd.nhtsa.dot.gov/pdf/nrd-30/NCSA/TSF2003/809769.pdf`.

[40] U. Franke and D.M. Gavrila. Autonomous driving goes downtown. *IEEE Intelligent Systems*, 13(6):40–48, 1999.

[41] U. Franke and A. Joos. Real–time stereo vision for urban traffic scene understanding. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Dearborn, USA, 2000.

[42] U. Franke and I. Kutzbach. Fast stereo based object detection for Stop & Go traffic. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 339–344, Tokyo, Japan, 1996.

[43] Y. Freund and R.E. Schapire. A decision–theoretic generalization of on–line learning and an application to boosting. *Journal of Computer Science and System Sciences*, 55(1):119–139, 1997.

[44] Kay Ch. Fuerstenberg. Pedestrian protection using laserscanners. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 115–119, Vienna, Austria, 2005.

[45] P.F. Gabriel, J.G. Verly, J.H. Piater, and A. Genon. The state of the art in multiple object tracking under occlusion in video sequences. In *Advanced Concepts for Intelligent Vision Systems*, Ghent, Belgium, 2003.

[46] D.M. Gavrila. Multi–feature hierarchical template matching using distance transforms. In *Proceedings of the International Conference in Pattern Recognition*, pages 439–444, Brisbane, Australia, 1998.

[47] D.M. Gavrila. Pedestrian detection from a moving vehicle. In *Proceedings of the the European Conference on Computer Vision*, volume 2, pages 37–49, Dublin, Ireland, 2000.

[48] D.M. Gavrila. Sensor–based pedestrian protection. *IEEE Intelligent Systems*, 16(6):77–81, 2001.

[49] D.M. Gavrila and J. Giebel and S. Munder. Vision–based pedestrian detection: The PROTECTOR system. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004.

[50] D.M. Gavrila and M. Kunert and U. Lages. A multi–sensor approach for the protection of vulnerable traffic participants – the PROTECTOR project. In *IEEE Instrumentation and Measurement Technology Conference*, pages 2044–2048, 2001.

[51] D.M. Gavrila and J. Giebel. Virtual sample generation for template–based shape matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 676–681, Kauai, USA, 2001.

[52] D.M. Gavrila and V. Philomin. Real–time object detection for smart vehicles. In *Proceedings of the International Conference on Computer Vision*, pages 87–93, Kerkyra, Greece, 1999.

[53] D. Gerónimo, A.D. Sappa, and A. López and D. Ponsa. Pedestrian detection using AdaBoost learning of features and vehicle pitch estimation. In *Proceedings of the International Conference on Visualization, Imagind and Image Processing*, pages 400–405, Palma de Mallorca, Spain, 2006.

[54] D. Gerónimo and J. Serrat. Visión artificial aplicada a vehículos inteligentes. Technical report, Universitat Autònoma de Barcelona, 2004.

[55] J. Giebel, D.M. Gavrila, and C. Schnör. A bayesian framework for multi–cue 3D object tracking. In *Proceedings of the the European Conference on Computer Vision*, pages 241–252, Prague, Czech Republic, 2004.

[56] E. Goubet and J. Katz and F. Porikli. Pedestrian tracking using thermal infrared imaging. In *Proceedings of the SPIE Conference Infrared Technology and Applications*, volume 6206, pages 797–808, Orlando, USA, 2006.

[57] G. Grubb, A. Zelinsky, and L. Nilsson and M. Rilbe. 3D vision sensing for improved pedestrian safety. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004.

[58] B. Heisele and C. Whöler. Motion–based recognition of pedestrians. In *Proceedings of the International Conference in Pattern Recognition*, pages 1325–1330, Brisbane, Australia, 1998.

[59] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the the European Conference on Computer Vision*, pages 343–356, Copenhagen, Denmark, 1996.

[60] J.Sporring, M. Nielsen, and L. Florack and P.Johansen (Eds). *Gaussian Scale–Space Theory.* Kluwer Academic Publishers, 1997.

[61] R. Klette, K. Schluns, A. Koschan, and A. Koschan and K. Schluns. *Computer Vision: Three–Dimensional Data from Images.* Springer–Verlag Singapore Pte. Limited, 1998.

[62] K. Konolige. Small vision systems: Hardware and implementation. In *Proceedings of the International Symposium on Robotics Research*, pages 111–116, Hayama, Japan, 1997.

[63] R. Labayrade and D. Aubert and J.P. Tarel. Real time obstacle detection in stereovision on non flat road geometry through v–disparity representation. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Versailles, France, 2002.

[64] D. Lefée, S. Mousset, and A. Bensrhair and M. Bertozzi. Cooperation of passive vision systems in detection and tracking of pedestrians. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 768–773, Parma, Italy, 2004.

[65] K. Levi and Y. Weiss. Learning object detection from a small number of examples: the importance of good features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 53–60, Washington DC, USA, 2004.

[66] Y. Liang, H. Tyan, and H. Liao and S. Chen. Stabilizing image sequences taken by the camcorder mounted on a moving vehicle. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pages 90–95, Shangai, Chinga, 2003.

[67] Y-M. Liang, H-R. Tyan, S-L. Chang, and H-Y. Mark Liao and S–W. Chen. Video stabilization for a camcorder mounted on a moving vehicle. *IEEE Transactions on Vehicular Technology*, 53(6):1636–1648, 2004.

[68] R. Lienhart and J. Maydt. An extended set of Haar–like features for rapid object detection. In *Proceedings of the IEEE International Conference on Image Processing*, 2002.

[69] P. Lombardi. A survey on pedestrian detection for autonomous driving systems. Technical report, Dipartimento di Informatica e Sistemistica, Università di Pavia, 2001.

[70] A. López. *Multilocal methods for ridge and valley delineation in image analysis.* PhD thesis, Centre de Visió per Computador and Universitat Autònoma de Barcelona, 1999.

[71] D.G. Lowe. Distinctive image features from scale–invariant keypoints. *International Journal on Computer Vision*, 60(2):91–110, 2004.

[72] M. Mählisch, M. Oberländer, O. Löhlein, and D.M. Gavrila and W. Ritter. A multiple detector approach to low–resolution FIR pedestrian recognition. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 325–330, Las Vegas, USA, 2005.

[73] S. Mallat. A theory for multiresolution signal decompositivon: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[74] P. Marchal, M. Dehesa, D.M. Gavrila, M-M. Meinecke, and N. Skellern and R. Viciguerra. SAVE–U final report. Technical report, Information Society Technology Programme (1998–2002) of the European Community, 2005.

[75] O. Masoud and N. Papanikolopoulos. A novel method for tracking and counting pedestrians in real–time using a single camera. *IEEE Transactions on Vehicular Technology*, 50(5):1267–1278, 2001.

[76] M. Meinecke, M.A. Obojski, M.Töns, R. Doerfler, P. Marchal, and L. Letellierand D.M. Gavrila and R. Morris. Approach for protection of vulnerable road users using sensor fusion techniques. In *Proceedings of the International Radar Symposium*, Dresden, Germany, 2003.

[77] M-M. Meinecke, M.A. Obojski, and M. Töns and M. Dehesa. SAVE–U: first experiences with a pre–crash system for enhancing pedestrian safety. In *European Congress and Exhibition on Intelligent Transportation Systems and Services*, Hannover, Germany, 2005.

[78] K. Mitsuhashi. Japanese pedestrian crashes. in *Proceedings of the Eight U.S./Japan Workshop on Advanced Technology in Highway Engineering: Nighttime and Pedestrian Safety.* Technical report, Federal Highway Administration, 1999.

[79] A. Mohan and C. Papageorgiou and T. Poggio. Example–based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.

[80] National Center of Health Statistics. National vital statistics report, 2002.

[81] N.A. Ogale. A survey of techniques for human detection from video. Technical report, Department of Computer Science, University of Maryland, 2005.

[82] M. Oren, C. Papageorgiou, P. Sinha, and E. Osuna and T. Poggio. Pedestrian detection using wavelet templates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 193–199, Puerto Rico, 1997.

[83] C. Papageorgiou and T. Evgeniou and T. Poggio. A trainable pedestrian detection system. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 241–246, Stuttgart, Germany, 1998.

[84] C. Papageorgiou and T. Poggio. Trainable pedestrian detection. In *Proceedings of the International Conference on Image Processing*, volume 4, pages 35–39, Kobe, Japan, 1999.

[85] C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal on Computer Vision*, 38(1):15–33, 2000.

[86] V. Philomin and R. Duraiswami and L.S. Davis. Pedestrian tracking from a moving vehicle. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 350–355, Dearborn, USA, 2000.

[87] D. Ponsa, A. López, F. Lumbreras, and J. Serrat and T. Graf. 3D vehicle sensor based on monocular vision. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, Vienna, Austria, 2005.

[88] D. Ponsa, A. López, J. Serrat, and F. Lumbreras and T. Graf. Multiple vehicle 3D tracking using an unscented Kalman filter. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, Vienna, Austria, 2005.

[89] PReVENT. `http://www.prevent-ip.org`.

[90] P. Rousseeuw and A. Leroy. *Robust regression and outlier detection.* John Wiley & Sons, Inc., 1987.

[91] C. Rudin and I. Daubechies and R.E. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

[92] G. Sala and D. Neunzig. Protector: Accident analysis. deliverable no. d02.1. Technical report, 2000.

[93] A.D. Sappa, D. Gerónimo, and F. Dornaika and A. López. On–board camera extrinsic parameter estimation. *IEE Electronic Letters*, 42(13):745–747, 2006.

[94] A.D. Sappa, D. Gerónimo, and F. Dornaika and A. López. Real time vehicle pose using on–board stereo–vision system. In *International Conference on Image Analysis and Recognition*, Póvoa de Varzim, Portugal, 2006.

[95] SAVE–U: Sensors and System Architecture for Vulnerable Road Users Protection. `http://www.save-u.org`.

[96] R.E. Schapire, Y. Freund, and P. Barlett and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

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

[98] U. Scheunert, H. Cramer, and B. Fardi and G. Wanielik. Multi sensor based tracking of pedestrians: A survey of suitable movement models. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 774–778, Parma, Italy, 2004.

[99] B. Scholkopf and A. Smola. *Learning with Kernel Support Vector Machines, Regularization, Optimization and Beyond.* MIT Press, Cambridge, MA, 2002.

[100] A. Shashua and Y. Gdalyahu and G. Hayun. Pedestrian detection for driving assistance systems: Single–frame classification and system level performance. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004.

[101] M. Soga, T. Kato, M. Ohta, and Y. Ninomiya. Pedestrian detection with stereo vision. In *Proceedings of the IEEE International Conference on Data Engineering*, Tokyo, Japan, 2005.

[102] G. P. Stein, O. Mano, and A. Shashua. A robust method for computing vehicle ego–motion. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 362–368, Dearborn, USA, 2000.

[103] E. Stollnitz and T. DeRose and D. Salesin. Wavelets for computer graphics: A primer. *IEEE Computer Graphics and Applications*, 1994.

[104] F. Suard and A. Rakotomamonjy and A. Broggi. Pedestrian detection using infrared images and histograms of oriented gradients. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 206–212, Tokyo, Japan, 2006.

[105] H. Sun and C. Hua and Y. Luo. Multi–stage classifier based algorithm of pedestrian detection in night with a near infrared camera in a moving car. In *Proceedings of the IEEE International Conference on Image and Graphics*, pages 431–436, Hong Kong, China, 2004.

[106] H. Sunyoto and W. van der Mark and D.M. Gavrila. A comparative study of fast dense stereo vision algorithms. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004.

[107] M. Szarvas, A. Yoshizawa, and M. Yamamoto and J. Ogata. Pedestrian detection with convolutional neural networks. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 224–229, Las Vegas, USA, 2005.

[108] T. Tsuji, H. Hattori, and M. Watanable and N. Nagaoka. Development of night–vision system. *IEEE Transactions on Intelligent Transportation Systems*, 3(3):203–209, 2002.

[109] W. van der Mark and .M. Gavrila. Real–time dense stereo for intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):38–50, 2006.

[110] J. van Schalkwyk. The magnificent ROC., 2001. `http://www.anaesthetist.com/mnm/stats/roc`.

[111] P. Viola and M. Jones and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 734–742, 2003.

[112] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Kauai Marriott, USA, 2001.

[113] L. Vlacic and M. Parent and F. Harashima. *Intelligent Vehicle Technologies.* Butterworth-Heinemann, 2001.

[114] C. Wang, H. Tanahashi, H. Hirayu, and Y. Niwa and K. Yamamoto. Comparison of local plane fitting methods for range data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 663–669, Hawaii, USA, 2001.

[115] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical report, University of North Carolina at Chapel Hill, Department of Computer Science, 2002.

[116] C. Whöler and U. Kreßel and J.K. Anlauf. Pedestrian recognition by classification of image sequences – global approaches vs. local spatio–temporal processing. In *Proceedings of the International Conference in Pattern Recognition*, Barcelona, Spain, 2000.

[117] C. Wöhler and J. K. Anlauf. An adaptable time–delay neural–network algorithm for image sequence analysis. *IEEE Transactions on Neural Networks*, 10(6):1531–1536, 1999.

[118] C. Wöhler and J. K. Anlauf. Real–time object recognition on image sequences with the adaptable time delay neural network algorithm – applications for autonomous vehicles. *Image and Vision Computing*, 19(9–10):593–618, 2001.

[119] F. Xu and X. Liu and K. Fujimura. Pedestrian detection and tracking with night vision. *IEEE Transactions on Intelligent Transportation Systems*, 6(1):63–71, 2005.

[120] L. Zhao and C. Thorpe. Stereo and neural network–based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(3):148–154, 2000.

[121] Q. Zhu, S. Avidan, and M-C. Yeh and K–T. Cheng. Fast human detection using a cascade of histrograms of oriented gradients. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York, USA, 2006.