PEDESTRIAN DETECTION USING LASER AND VISION

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF
ELECTRICAL AND COMPUTER ENGINEERING
OF COIMBRA UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Cristiano Premebida

UNDER SUPERVISION OF
Professor Urbano J. Carreira Nunes
Abstract

A multisensor-based pedestrian detection system is proposed in this thesis with the purpose of detecting pedestrians, in urban scenarios, in the context of protection systems for Advanced Driver Assistance Systems (ADAS) applications. A multi-layer LIDAR and a monocular color camera, mounted on-board an instrumented electrical vehicle (the ISRobotCar), are the primary sensors used in the proposed system. The problem of pedestrian classification and detection is approached by three systems: (1) LIDAR-based system, (2) Vision-based system, and (3) Information fusion system. LIDAR and Vision-based systems are addressed and studied separately, while the Information fusion system integrates techniques, methods and information from the LIDAR and Vision-based systems. Moreover, solutions to estimation of the vehicle's velocity and objects speed are proposed. As concerns pedestrian classification problem, proposed solutions are presented and compared, in terms of classification performance, as function of: classification methods, feature spaces, fusion schemes, and fusion rules. Additionally, ensemble of classifiers, used in the form of rejection-cascades are developed and discussed. In particular, a SVM-cascade is proposed and compared with other rejection-cascades, namely AdaBoost-based cascades. This thesis also presents three architectures for pedestrian classification using LIDAR and monocular camera: centralized, decentralized, and multistage. Regarding the problem of pedestrian detection, several techniques and algorithms were developed, in particular: sample selection methods, cascade of classifiers, LIDAR and camera-based data integration, context-based information fusion, decision making. To support experimental analysis, a multi-sensor dataset, constituted by data from a LIDAR, camera, encoder and DGPS, was used. The aforementioned dataset, named Laser and Image Pedestrian Detection (LIPD) dataset, was collected in an urban environment, at day light conditions, using the ISRobotCar driven at low speed up to 35Km/h. Finally, it is worth of mention the main contributions of this thesis: the LIPD dataset; a multivariable LIDAR-based segmentation method; sensor fusion architectures; sample selection methods based on LDA and SVM; the SVM-cascade classification method; a context-based pedestrian detection solution.
Contents

Acknowledgment iii

Resumo v

Abstract vii

List of Tables xiv

List of Figures xx

List of Acronyms and Glossary xxii

1 Introduction 1

1.1 Motivation and Objectives 6

1.2 Research context and problem summary 7

1.3 Data processing systems for pedestrian detection 8

1.3.1 LIDAR-based system 8

1.3.2 Vision-based system 9

1.3.3 Information fusion system 10

1.4 Contributions 11

1.5 Thesis Outline 13

2 State of Art 15
2.1 LIDAR-based systems for pedestrian detection  
2.2 Vision-based systems for pedestrian detection  
2.3 Pedestrian detection using LIDAR and monocular camera  
2.4 Summary  

3 Background and Dataset  
3.1 LIDAR and camera characteristics  
3.2 LIDAR and camera calibration  
3.3 Pedestrian detection ruled by probability theory  
3.4 Classification methods  
3.5 Risk minimization  
3.6 The LIPD dataset  
3.6.1 Classification dataset  
3.6.2 Detection dataset  

4 LIDAR-based system  
4.1 Preprocessing  
4.2 Segmentation  
4.2.1 Single vs Multilayer data segmentation  
4.2.2 Multivariable segmentation  
4.3 Feature extraction  
4.3.1 Feature analysis and comparison  
4.4 Classification  
4.5 Speed estimation and modeling for pedestrians
5 Vision-based system

5.1 Descriptors for pedestrian classification
  5.1.1 Feature selection

5.2 Pedestrian classification

5.3 Dataset resampling

5.4 Pedestrian detection
  5.4.1 LIDAR-based ROI generating method
  5.4.2 Rejection cascades

6 Information fusion system

6.1 Information fusion for pedestrian classification

6.2 Combining classifiers
  6.2.1 Single classifiers scheme
  6.2.2 Nontrainable fusion rules
  6.2.3 Trainable fusion rules
  6.2.4 Multistage method
  6.2.5 Centralized fusion architecture
  6.2.6 Decentralized fusion architecture
  6.2.7 Experiments

6.3 Sensor fusion for vehicle velocity estimation
  6.3.1 Experiments using the Kalman Filter

7 Pedestrian detection using LIDAR and vision

7.1 Pedestrian detection without contextual information

7.2 Criterion used for performance evaluation
7.3 Classification methods used in a decision module 111

7.4 Context-based information 112

7.4.1 Context-based system using object position in semantic map 113

7.4.2 Context-based system using object speed 115

7.5 Experiments 115

7.5.1 Experiments with single classifiers 116

7.5.2 Experiments with the Haar-AdaBoost cascade 117

7.5.3 Experiments with the S-AdaBoost and the M-AdaBoost using HOG-COV features 117

7.5.4 Experiments with the SVM-cascade and the SRM-cascade 118

7.5.5 Experiments with the context-based system:
incidence regions 120

7.5.6 Experiments with the context-based system:
obstacles speed 121

8 Conclusion and Future Work 125

A Feature selection 129

Bibliography 133
List of Tables

2.1 Outline of some related work on LIDAR-based systems for pedestrian detection/classification in urban scenarios. ........................................ 16

2.2 List of recent surveys on vision-based systems for pedestrian detection/classification in urban scenarios. ........................................ 17

2.3 Research on LIDAR and vision data fusion systems for pedestrian and on-road objects detection/classification in urban scenarios. ....... 18

3.1 Camera and LIDAR specifications ........................................ 22

3.2 Intrinsic camera parameters: $K$ ........................................ 23

3.3 Terminology and definitions for performance metrics .................. 31

3.4 Sensors used in the acquisition of the LIPD dataset .................. 34

3.5 Statistics of the classification set $\mathcal{D}_C$ .......................... 40

3.6 Statistics of detection dataset $\mathcal{D}_D$ .............................. 42

4.1 Classification performance: testing set .............................. 57

4.2 Polynomial fitting up to third-order models ...................... 62

5.1 Vision-based features relevance (in %) .............................. 67

5.2 Classifiers parameters ........................................ 69

5.3 Classification performance results .............................. 69

5.4 Non-resize vs resize of the detection windows .................. 76

6.1 Classification performance using single-classifers .............. 93
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>Classification performance using nontrainable fusion rules</td>
<td>93</td>
</tr>
<tr>
<td>6.3</td>
<td>Classification performance using trainable fusion rules</td>
<td>94</td>
</tr>
<tr>
<td>6.4</td>
<td>Classification performance using the multistage ensemble</td>
<td>95</td>
</tr>
<tr>
<td>6.5</td>
<td>Discrete-time models for single incremental encoder</td>
<td>99</td>
</tr>
<tr>
<td>6.6</td>
<td>Discrete-time models for DGPS</td>
<td>101</td>
</tr>
<tr>
<td>7.1</td>
<td>Single classifiers performance</td>
<td>116</td>
</tr>
<tr>
<td>7.2</td>
<td>Nontrainable fusion rules performance</td>
<td>117</td>
</tr>
<tr>
<td>7.3</td>
<td>Detection performance using Haar-like AdaBoost cascade</td>
<td>117</td>
</tr>
</tbody>
</table>
List of Figures

1.1 The main data processing stages are summarized by three systems: (1) LIDAR, (2) Vision, and (3) Information fusion. The processing stages of each system are pointed out in their boxes. .......................... 3

1.2 Schematic representation of the processing stages of a generic pedestrian detection system using LIDAR and monocular data. .................. 4

1.3 ISRobotCar and onboard sensor setup, enlarged at the bottom-right part, used in the dataset collection. A short specification of the sensors is presented at the top-right side of the figure. ....................... 7

1.4 Measurement flow-diagram depicting the sensors, the interface protocols, and the energy supply. Except the base-GPS, all the sensors illustrated in this figure were mounted on-board the ISRobotCar. ... 12

3.1 Google© map of the Engineering Campus and the GPS trajectory, in black, traveled by the ISRobotCar. ................................. 35

3.2 Laser-based ROI projections in the image frame. The laser-points are illustrated by dots in the laser space and their projections in the image are represented by crosses, where the colors represent a LIDAR layer. The segments, enclosed by black-ellipses, and their corresponding pedestrians are marked as bounding-boxes in the image. ........ 37

3.3 Illustrative images of: cropped pedestrians (upper part of the figure); full frames (middle part); and ROI projections (blue rectangular boxes, illustrated in the lower part of the figure). ............................. 38

3.4 Training dataset distribution regarding the bounding-boxes height and width (in pixels). ............................................................... 39
3.5 Bounding-boxes geometrical center, defined in pixel coordinates, for the training dataset. Pedestrian locations are concentrated around the region with average pixel equal to 230. ................................. 40

3.6 Ground-truth bounding-boxes of the test dataset. At the right, it is depicted the distribution in the vertical axis, with a significant concentration around the mean i.e., 270. At the bottom, the distribution in the horizontal axis is shown, with 43.75% of the ground-truth in the interval [320, 460]. ......................................................... 41

3.7 Representation of the LIPD dataset, in the form of a diagram, showing the classification and detection sets, $D_C$ and $D_D$, and their subsets. Regarding $D_D$, notice the following: (1) the exact number of negative examples on $D_{ima}$ depends on the detection and classification approaches to be used; (2) raw laser data is provided; (3) all the positive examples correspond to labeled cropped images. ......................................................... 42

4.1 Block diagram of the LIDAR-based system. LIDAR data passes through the processing stages towards objects classification. .......................... 43

4.2 Laser coordinates convention, with the Cartesian and polar representation of a given point $p_i$. At the right, the laser-layers and the vertical angular resolution are shown. ................................. 44

4.3 Processing steps in the LIDAR-based preprocessing stage. (a) Range-points in Cartesian coordinates. (b) Points are filtered and grouped per layer, where each color indicates a given layer. (c) Range-points outside the camera FOV are discarded. (d) Projection of the points, per layer, in the image frame. Laser-to-image projections, and the FOV determination, depend on the calibration, described in 3.2. .... 45

4.4 (a) Example of laser-points before segmentation, where the points were colored as function of the laser-layers according the convention given in Fig. 4.2. (b) Set of segments $S_i, i = 1, \cdots, 4$ obtained by the segmentation stage. ......................................................... 46
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>An example that illustrates a segment of range points, which corresponds to a pedestrian in the laser plane, with the corresponding feature set (in the text-box); some attributes and geometric primitives are also depicted.</td>
</tr>
<tr>
<td>4.6</td>
<td>Redundancy computed using the feature set ( f_i, i = 1, \ldots, 18 ), for the lasers Ibeo Alasca-XT (left) and Sick LMS200 (right). The features with indexes ( i = 2, 6, 12, 13, 14, 17 ) are very redundant, particularly for the Sick.</td>
</tr>
<tr>
<td>4.7</td>
<td>From the results on the relevance calculated for both sensory feature sets, the set of features most relevance are: ( f_i, i = 1, \ldots, 6, 14, 15, 17 ).</td>
</tr>
<tr>
<td>4.8</td>
<td>ROCs of the single classifiers using the classification dataset obtained with the Ibeo laser (a), and with the Sick LMS200 (b).</td>
</tr>
<tr>
<td>4.9</td>
<td>Number of features vs classifier performance using the dataset obtained with the Ibeo Alasca-XT and with Sick LMS200.</td>
</tr>
<tr>
<td>4.10</td>
<td>An example of the trajectory walked by a pedestrian, perceived by the LIDAR (gray points). The center of the circles extracted from the laser points, depicted as red crosses, were used as reference points for fitting the polynomials.</td>
</tr>
<tr>
<td>5.1</td>
<td>Covariance matrices are calculated in the regions ( R_1, \ldots, 5 ) of a given input image ( I(x, y) ). A 36-dimensional feature vector ( f_{cov} ) is obtained from each matrix ( COV_{R_i} ), totaling 180 feature elements.</td>
</tr>
<tr>
<td>5.2</td>
<td>ROC curves of the classifiers obtained with the average of 4-fold cross validation on the vision-based classification dataset.</td>
</tr>
<tr>
<td>5.3</td>
<td>Negative samples which possess feature vectors positioned 'far' from the separation margin (a); and samples which correspond to support vectors (b).</td>
</tr>
<tr>
<td>5.4</td>
<td>Representation of a detection window, ( \mathcal{D}W_i ), with size defined by ( (w_i, h_i) ), which is shifted, inside the ROI, at locations given by ( (x_i, y_i) ).</td>
</tr>
</tbody>
</table>
5.5 Functional diagram illustrating the negative-rejection cascade detector: each stage $k$ rejects the samples classified as negative (F: false), while the positives (T: true) pass through all the stages to be finally detected as a pedestrian. .................................................. 77

5.6 Empirical risk ($R_{emp}(\alpha)$), Bound on the expected risk, and Complexity penalty ($\epsilon(n, VCd, \eta)$) for the SVM-cascade. ............................... 78

5.7 Results on the training dataset in terms of $FPR$ (left part) and $TPR$ (right part), as function of the number of component-classifiers, using the SVM-cascade. .................................................. 79

5.8 Results on the training dataset in terms of $TPR$ (left part) and $FPR$ (right part), as function of the number of stages, for the S-AdaBoost (in black) and for the M-AdaBoost (in blue). ......................... 81

6.1 Illustrative example of the bagging process: $D_{bagg}$ is obtained considering the samples inside the margin defined by a base-linear classifier. 89

6.2 Functional diagram of the multistage ensemble. ......................... 90

6.3 Centralized sensor fusion architecture: LIDAR and vision information are combined at the feature level, and the pedestrian classification system can be carried out by single-classifiers or fusion methods using features extracted from both sensors. ................................. 91

6.4 Decentralized sensor fusion architecture: the LIDAR and the vision-based features are processed separately, therefore the classification decision can be achieved per feature space or by combining the outputs from each sensor-based module. ........................................ 91

6.5 ROCs for the single classifiers according to the feature set.............. 92

6.6 ROCs for the nontrainable fusion rules according to the feature set. .. 94

6.7 ROCs for the trainable fusion rules according to the feature set........ 95

6.8 ROCs for the multistage ensemble according to the feature set......... 95

6.9 DGPS inconsistencies. .................................................. 100
6.10 Speed estimation $\dot{\vartheta} = \hat{\dot{\vartheta}}$, using the KF, for encoder measurements. Results in dashed-blue were obtained for the 2nd-order model, and the results for the 3rd-order model are shown in black. .......................... 103

6.11 Estimated speed $\dot{\vartheta} = \sqrt{\dot{\vartheta}_{px}^2 + \dot{\vartheta}_{py}^2}$, without the validation stage, using DGPS measurements. The results for the models with $n = 2$ and $n = 3$ are shown in dashed-blue and black respectively. .......................... 104

6.12 DGPS-based estimated speed $\dot{\vartheta}$, for both models, using the KF$^g$ algorithm with validation region (VR). .......................... 104

6.13 Speed $\dot{\vartheta}$ estimated using DGPS and encoder measurements. When a DGPS measurement does not pass the data association stage, $\dot{\vartheta}$ is estimated according to expression (6.25). .......................... 105

7.1 Pedestrian detection system composed by LIDAR and Vision-based systems. .......................... 108

7.2 Block diagram representation of the decision module. The main stages comprised in this module are: classification in the form of multi-scale sliding-windows, non-maximum suppression filtering, and decision making. .......................... 109

7.3 Laser-based ROIs in the image frame. Entire body pedestrians (class type 1) and partially occluded pedestrians (class type 0) are highlighted in the image by red and blue bounding-boxes respectively. .......................... 111

7.4 Block diagram illustrating the main processing modules comprised in the pedestrian detection system when contextual information is incorporated into the system. .......................... 112

7.5 Semantic map of the environment where the dataset were collected. Examples of incidence zones, used in the context-based system, are given in (b) and (c). .......................... 114

7.6 Estimated probability density function (pdf) for the classes PED (a) and nPED (b). These distributions are used to determine the prior probability of a given object be a PED or a nPED as function of the speed estimated by the LIDAR-based module. .......................... 115
7.7 Results, on the test set, regarding the true positive rate (a), and the false positive rate (b), for the Haar-AdaBoost rejection cascade as function of the number of stages. .............................................. 118

7.8 Results, on the test set, regarding the true positive rate (a), and the false positive rate (b), for the S-AdaBoost (curve in blue), and for the M-AdaBoost (in black). .......................................................... 119

7.9 Results, on the test set, regarding the true positive rate (a), and the false positive rate (b), for the SRM-cascade (blue-square markers) and for the SVM-cascade (black-circle markers). ....................... 119

7.10 Histogram and Normal distribution fitted to the output scores of the SVM in the first stage of the SVM-cascade. The decision is indicated by the vertical line (in red). At $x = 0$ the SVM guarantee 98% of $TPR$, according to the threshold $Thr_{tp}$ chosen for the cascade training (see Section 5.4.2). ......................................................... 121

7.11 Results for the decision module using the MAP decision, where the likelihoods are given by the SVM-cascade and the priors come from the context-based module. The performance results are shown as function of the prior probability of the object be a pedestrian in the incidence zones (represented by $P$ in the texts arrows)................................. 122

7.12 Results for pedestrian detection using the SVM-cascade and priors derived by the distributions of the estimated speed of the objects in the scene. ................................................................. 123
List of Acronyms and Glossary

ADAS......... Advanced Driver Assistance Systems
ANN.......... Artificial Neural Network
BER.......... Balanced Error Rate
COV.......... Region Covariance descriptor
ERM.......... Empirical Risk Minimization
FOV.......... Field of View
GMM.......... Gaussian Mixture Models
HOG.......... Histogram of Oriented Gradients
ITS.......... Intelligent Transportation System
IV........... Intelligent Vehicles
KF........... Kalman Filter
LDA.......... Linear Discriminant Analysis
LIDAR........ Light Detection And Ranging
LIPD.......... Laser and Image Pedestrian Detection Dataset
MAP.......... Maximum A Posteriori
ML........... Maximum Likelihood
MV........... Majority Vote, fusion rule
NBC .......... Naive Bayes Classifier
NN .......... Nearest Neighbour
nPED .......... Non-pedestrian (negative class)
PDF .......... Probability Density Function
PDS .......... Pedestrian Detection System
PED .......... Pedestrian (positive class)
ROC .......... Receiver Operating Characteristic curve
ROI .......... Region Of Interest
SVM .......... Support Vector Machine
VR .......... Validation Region
VRU .......... Vulnerable Road Users
Chapter 1

Introduction

PROTECTION systems for pedestrian safety, in urban environment, is an emerging scientific research area of Advanced Driver Assistance Systems (ADAS) which achieved a notable development in the last decade. It is still a growing research field, evidenced by recent projects [ULTra, 2010], challenges [DARPA, 2007] [EL-ROB, 2010], and recent publications [Enzweiler and Gavrila, 2011] [Geronimo et al., 2010] [Dollar et al., 2009] [Gandhi and Trivedi, 2007]. For instance, in the last years, three significant surveys in pedestrian detection and protection systems, in the context of Intelligent Vehicles (IV) and Intelligent Transportation Systems (ITS), were published in [Gandhi and Trivedi, 2006] [Enzweiler and Gavrila, 2009] [Geronimo et al., 2010]. The main reasons for the interest in this scientific domain are basically:

- society concerns: it is an appealing topic of research due to its direct impact in the society, in terms of traffic casualties and the large economic and societal cost implicated;

- industry involvement: there is a strong interest of the automotive industry, indicated by the continuous enhancements on safety features in the vehicles;

- research interest: international projects consortium, international challenging competitions and awards, and the scientific community in several fields have demonstrated, and still do, large interest in innovation and developments associated to this topic.

Pedestrian protection systems can be divided, in general words, in two fields of
resea rch: passive and active safety systems. The former is characterized by built-in
safety features in vehicles, designed primarily to mitigate possible injuries in pedes-
trians due to an impact e.g., special designed front bumper, deformable hood, specific
air-bags placed nearby the frontal columns of the vehicle, and so on. The later, ac-
tive safety systems, which is of interest here, are based on pedestrian detection using
sensors on-board a vehicle, and/or on the infrastructure, with the role of predict and
anticipate possible risks of collision. In particular, active pedestrian detection systems
using on-board LIDAR, or laserscanner, and monocular camera is emphasized. More
specifically, this thesis is focused on different strategies for LIDAR and image data
processing where the algorithms and methods were developed in two main schemes:
(1) centralized scheme, where the features extracted from the sensors data are com-
bined and then processed in a common module; and (2) decentralized or parallel
scheme, i.e., each sensor has an independent processing module.

Active protection systems\(^1\) have deserved much attention in the recent years be-
cause of many applications in the automotive market e.g., Adaptive Cruise Control
(ACC), Lane Departure Warning (LDW), Anti-lock Braking System (ABS), Collision
Warning Systems (CWS). Onboard sensor-based detection systems demand much re-
search efforts, since ACC, LDW, and CWS have the primary decision step depending,
directly or indirectly, on the outputs from sensors. Thus, putting in these terms, a
sensor-based detection is one of the primary modules of a general ADAS’s protection
system. However, it is important to note that the outputs of such detection module
involve many inter-dependent processing stages (or internal modules), for instance:
data acquisition, measurement alignment, synchronization, preprocessing, filtering,
feature/attributes extraction, data association, tracking, classification and decision
making. Despite to the fact that object detection, and pedestrian detection in par-
ticular, is an interesting research topic, it is by no means an easy problem to deal
with. Nevertheless, and because this is an emerging researching area in ISR-UC, it
is a very stimulating topic of research. In summary, Figure 1.1 illustrates the main
processing systems covered in this thesis: (1) LIDAR-based system; (2) Vision-based
system; and (3) Information fusion system.

\(^1\)Although this work is focused on pedestrian detection, other entities, or obstacles, can also be
considered, such as: vehicles, Vulnerable Road Users (VRU) in general, static and moving objects,
and so on.
Figure 1.1: The main data processing stages are summarized by three systems: (1) LIDAR, (2) Vision, and (3) Information fusion. The processing stages of each system are pointed out in their boxes.

Three main areas of statistical theory, applied to the problem of pedestrian detection in the field of IV/ITS, shall be discussed and revisited: estimation, pattern recognition (classification), and detection. More specifically, these topics are conceptually presented below:

- parameter estimation: is the stochastic framework necessary on mapping the object (detected entity) properties/attributes and the sensors measurements into the parameters (state variables) of a mathematical model, namely, the statistical characterization of the process that governs and describes the uncertainties of the system.

- classification: is a decision process of selecting a class (e.g., positive or negative) based on a classification method (the classifier) trained on a labeled set defined in a given feature space.

- detection: outputs the estimated position and scale, in the image plane, of potential pedestrians present in the sensors field of view (FOV). The decision should be considered under non-deterministic conditions, but usually modeled as an uncertainty following a given or approximated distribution in a statistical framework.
Although for some researchers a detection system encompass a tracking stage, or dynamic state variable estimation, a detection system can be designed without a explicit tracking framework. Hence, tracking will be considered a topic apart of the detection system itself - even having knowledge that categorically a tracking module is part of the whole detection architecture. Thus, the term detection hereafter refers to all the processing stages implicated in estimating the class label, the position, and the size (scale) of an object present in the sensor coverage area. Eventually, the dynamic behavior, estimated by means of a tracking module, is also involved on the detection process. Classification means the process of estimating the object’s class label, e.g., pedestrian(PED) or non-pedestrian(nPED), based on a classifier, or on an ensemble of classifiers, trained in a set of exemplars.

The perception system described here is decomposed into modules, or processing systems, aiming to give flexibility to the architecture and to facilitate the understanding of the framework. Although each module can be treated separately, which is the case of the decentralized scheme, it is important to note that the centralized and the multistage architecture has to be understood as an evolving and inter-dependent decision process which depends direct and/or indirectly of all the modules which take part on it. At the end, the architecture has to be seen as a jointly-framework where the processing stages have variable dependencies, function of the information flow along the processing modules. Figure 1.2 gives an overview of the proposed system designed for pedestrian detection (nevertheless, this schematic diagram can be extended for more general cases). The main processing tasks, involved in each module, are described in the text-boxes just below the blocks. In short:
1. Sensors & Dataset: the sensor specifications and characteristics, as well the dataset, are described in Chapter 3.

2. Preprocessing: specific preprocessing tasks in the LIDAR-based system are addressed in Chapter 4, while the vision-based system is discussed in Chapter 5.

3. Intermediate Processing: most of the processes implicated in this ‘general’ block are performed on the LIDAR-based system, except feature extraction which also takes part in the Vision-based system.

4. Classification & Detection: classification and detection are presented, in more general way, in Chapter 3, while specific implementations of pedestrian classification are detailed in Chapters 4, 5 and 6. As concerns pedestrian detection, experiments are reported in Chapter 7.

To infer the presence, the position and dynamics (speed), and to identify the category/class of an object, the pedestrian detection system relies on information from various sources and gathered at different time and conditions. The ‘sources’ of information and the conditions they follow are:

- sensor data: data from the sensors should be processed conveniently, and the measurements have to be acquired properly;

- features/attributes representation: raw information provided by the measurement units should be conveniently processed and converted into a representative i.e., informative feature space, to avoid redundancy, to decrease the complexity of the system, and to maintain as much as possible relevant (crucial) information which will be used in the classification module;

- prior and context-based information: refers to prior probabilities, used in the proposed context-based system, which model objects speed and contextual information retrieved by a semantic map;

- temporal-spatial evolution of the object: it is assumed that pedestrians, in most of the time, evolve over time changing its position in the sensors FOV and consequently its velocity; moreover, the vehicle’s movement has to be taken into consideration.
• general probabilistic system characterization: the various random (stochastic) factors that affect the system should be (if possible) characterized, namely: errors, disturbances, uncertainties, environment conditions, restrictions, physical constraints, and so on;

1.1 Motivation and Objectives

The motivation of this work is strictly related with developments and research activities back to October 2004: the time I have started a Master. Since then, the core of my work involved LIDAR data processing with the purpose of segmentation, tracking, and classification of objects. Since the beginning of 2007, when I have started my PhD, the work has been focused on pedestrian detection, in outdoor scenarios, using data from LIDAR (Sick LMS200 and Ibeo Alasca XT) combined with monocular image frames. The theme itself, its emerging potential applications, and the challenges inherent to the work explored here constituted the primary elements that feed the engine that kept the author motivation during this period of researching, programming, debugging, testing, etc.

The general goal of this thesis is to develop and to demonstrate the effectiveness of a practical sensor-fusion system for pedestrian detection. The framework that gives theoretical and technical support for such system is actually an ensemble of various techniques, algorithms and data-fusion schemes that are inherently covered by the main theoretical areas briefly discussed so far, which are: detection, estimation, inference (classification), and sensor fusion. The principal sensor apparatus used for the detection system is constituted by an Ibeo Alasca-XT LIDAR (a 4-layer laser scanner) and a VGA-firewire monocular camera mounted on the ISRobotCar (autonomous electric vehicle with a chassis from Yamaha Europe and control systems developed in the Institute for Systems and Robotics of Coimbra University) as shown in Fig. 1.3.

The work described here is, in general aspects, part of some researching projects under the supervision of my advisor and his partners at the Institute of Systems and Robotics (ISR-UC) which aim to study and to develop multi-target detection, tracking, and classification systems for intelligent vehicles, that is, perception systems to be integrated in vehicles and/or mobile robotics. The present study, on the other hand, has the objective to develop solutions to the problem of pedestrian detection using,
1.2. Research context and problem summary

Pedestrian Detection Systems (PDS), or pedestrian protection systems as in [Gandhi and Trivedi, 2007] and [Geronimo et al., 2010], is a specific type of ADAS, and it has the purpose of detecting the presence of a single or a group of pedestrians, stationary or in a moving behavior, in the FOV of onboard sensors, aiming to assist the driver in making decisions, for instance providing warning signals or, in an ultimate case, to take decisions and executing counteractive or evasive actions to mitigate the injuries in a potential collision. Some of the challenges of a PDS are:

1. Pedestrians attributes have a high variability in relation to shape, height, color (clothes), appearance, pose, behavior (stationary or moving), carrying objects, and they can appear in groups of persons or can be partially occluded.

Figure 1.3: ISRobotCar and onboard sensor setup, enlarged at the bottom-right part, used in the dataset collection. A short specification of the sensors is presented at the top-right side of the figure.
2. Urban environment, as any outdoor scenario, presents a wide range of conditions regarding the background, illumination, weather, obstacles, and so on.

3. Intrinsic and extrinsic factors have a vast influence in PDS, namely: sensor noise, scale factor (pedestrian appear at different viewing angles and range), resolution, detection performance, occlusions, vibrations, synchronization, calibration inaccuracy, apparent movement, oscillations, etc.

In order to obtain a robust and flexible solution to tackle these problems in a common framework, a sensor-fusion architecture composed of processing modules, designed for specific processing tasks, is proposed. This architecture and its intrinsic modules are described in the next section.

1.3 Data processing systems for pedestrian detection

The LIDAR, Vision and Information fusion systems, summarized in Fig. 1.1, contain specific processing stages which are briefly described in the sequel.

1.3.1 LIDAR-based system

This part of the system is emphasized and more explored, compared to the vision-based system, due to the research trajectory of the author in this field, that is back to 2005 [Premebida and Nunes, 2005], and the recent impact and increasing interest of the scientific community in using laser-based systems in tasks related to mobile robotics and autonomous vehicles [Premebida et al., 2009b] [Zhao et al., 2009] [Gidel et al., 2010]. Although a relevant work was made using range data from a Sick LMS200 laserscanner [Premebida and Nunes, 2006], the majority of the experimental analysis and the provided dataset are based on information from an Ibeo Alasca XT (4-layers automotive LIDAR), thus the subject of this section is on the stages and processes that are necessary to obtain a reliable perception system using range information - characterized by laser scans i.e., a sequence of range and bearing points. The LIDAR-based system is composed of the following processing modules:

**Preprocessing:** incorporates a set of base tasks aiming to reduce the complexity
1.3. DATA PROCESSING SYSTEMS FOR PEDESTRIAN DETECTION

of subsequent stages. The processes associated with this module are: sensor parameters adjustments, laser calibration, coordinate transformation, data acquisition, data synchronization, filtering, and FOV delimitation. Details regarding this stage is given in Section 4.1.

**Segmentation**: it is the primary stage to detect the entities of interest, where each entity/object constitutes a hypothesis of being a PED or a nPED. The segmentation process, addressed in Section 4.2, is performed in the LIDAR space, where the detected objects are characterized by a group/cluster of laser-points, here named *segment*, and can be performed by means of specific methods as presented in [Premebida and Nunes, 2005] or general data clustering techniques [Jain et al., 1999].

**Feature extraction**: geometrical/range features are extracted from laser-segments to compose a 18-dimensional feature vector which is used for pedestrian classification. The laser-based features are described in Section 4.3.

**Classification**: the classification stage, which can be formed by a single classifier or an ensemble of classifiers, is employed with the goal of obtaining good generalization capacity in test datasets. The classifiers and techniques used for pedestrian classification using laser-based features are discussed in Section 4.4.

**Tracking and modeling**: after the segments are filtered out and the objects of interest are detected, it is useful to keep the stochastic state of the object under tracking since the object behavior evolves in time and in space. The state and measurement models, used for pedestrians dynamic behavior estimation, are addressed in Section 4.5.

**Detection**: laser-segments are transformed into image coordinates as regions of interest (ROIs). Inside each ROI, a image-based classification method is used in the form of a multiscale sliding-window which is shifted in position and size for searching pedestrian evidence. Thus, the LIDAR-based system acts as primary object detection, and the decision making depends on the Vision-based system (see Section 5.4.1).

1.3.2 Vision-based system

The incorporation of a vision-based processing system for object classification is an intuitive solution that would greatly improve the system's classification performance
[Gandhi and Trivedi, 2007] [Geronimo et al., 2010]. But, compared with laser-based solutions, it demands significant time-processing, the feature space dimensionality is usually high, and the camera is much more sensitive to the environment changes. Regardless these drawbacks, vision-based pedestrian detection is a mature research domain with very promising solutions and of a greatly potential [Gandhi and Trivedi, 2006], [Dollar et al., 2009], [Enzweiler and Gavrila, 2009]. The vision-based system is used concurrently and separately of the LIDAR-based system [Premebida et al., 2009c]. The vision system has the following processing modules/stages:

**Preprocessing and calibration**: since the camera works with Bayer format, the images are converted to RGB-format and then to black-white scale for further processing. In off-line mode, camera intrinsic and extrinsic parameters are estimated.

**Feature extraction**: Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005], Region Covariance (COV) [Tuzel et al., 2006] and Haar-like features, based on intensity gradients of the image, are used to extract a set of features for pedestrian detection and classification. A description of these image-based features is given in Section 5.1. Depending on the adopted classification scheme, a feature selection method is also used (see 5.1.1).

**Classification**: the same for the LIDAR-based case, a set of classifiers over the image based feature-space are used to discriminate between PED and nPED. Section 5.2 presents the classification methods used in the Vision-based system. Moreover, experiments are reported regarding detection windows scales before applying a classification method, see Section 5.4.1.

**Sample selection**: two dataset resampling methods, using LDA and SVM, are proposed in Section 5.3. Sample selection is necessary to reduce the training dataset cardinality and to decrease the complexity of the training process.

**Detection**: image-based pedestrian detection techniques, algorithms and methods, in particular rejection-cascades, are addressed in Section 5.4.

### 1.3.3 Information fusion system

Two information fusion approaches are considered. Firstly, a centralized approach in which the data from the LIDAR and from the camera are combined in a mutual
feature space and then all the subsequent tasks are handled in a common processing module. Secondly, a decentralized fusion approach is used in such way that data from the sensors are processed separately and combined in a higher stage, at the likelihood level, by means of fusion methods.

Independently of the fusion approach to be devised, there are some requirements that have to be retained:

1. Calibration and synchronization between the LIDAR and the camera are mandatory. Some processing tasks were developed for the information fusion system, which are: camera calibration, laser and camera extrinsic calibration, multisensor data acquisition and synchronization (time-stamp index), context-based information fusion.

2. The fusion strategies are likelihood-based, hence, all classification techniques use precise or approximated-pdf parametrization models.

3. The experimental evaluation is based on a field-dataset (the LIPD dataset), where the ground-truth was generated under user-supervision using the images as reference.

1.4 Contributions

Existing methods for segmentation, feature selection, tracking, classification, detection, among others, were conveniently adapted and modified to address a solution to the problem of pedestrian detection in outdoor environment. Concurrently, novel methodologies and algorithmic extensions were developed to allow the application of these methods and theories to a realistic situation - characterized by the field dataset. The main contributions of this thesis are the strategies, methods and developments, used to combine information from a LIDAR and a monocular camera, mounted on-board a vehicle, for pedestrian detection in a dataset collected in an urban environment. More specifically, the contributions are:

1. **LIPD Dataset**: a field multisensor dataset collected in the UC-Engineering Campus, containing data from a 4-layer laserscanner, a camera, an encoder,
Figure 1.4: Measurement flow-diagram depicting the sensors, the interface protocols, and the energy supply. Except the base-GPS, all the sensors illustrated in this figure were mounted on-board the ISRobotCar.

and DGPS. A schematic representation of the sensors and their interfaces are shown in Fig. 1.4.

2. **LIDAR-based data processing:** many algorithms and methods have been developed in this module, in particular: (1) a multivariable segmentation method; (2) a coherent set of laser-based features for pedestrian classification; (3) a segment-to-segment association approach; (4) modeling and classification of segmented objects.

3. **Sensor fusion schemes:** three architectures, for LIDAR and image information fusion, were tested and compared, namely: centralized, decentralized, and multistage.

4. **Pedestrian detection system:** experimental evaluation, of four strategies, have been performed: (1) single classifiers; (2) centralized scheme; (3) decentralized scheme; (4) cascade of classifiers. Moreover a context-based system is proposed to improve the pedestrian detection.

Part of the work presented in this thesis have been published, or submitted, in the following articles, conferences and workshops:


---

Data from an IMU and from the pack of batteries, which supplies the vehicle, were also recorded. However these data were not used in this thesis.
Booosting-SVM and SRM-SVM cascade classifiers in laser and vision-based pedestrian detection". *IEEE/ITSC Conference.*


### 1.5 Thesis Outline

Chapter 2 covers recent state-of-art research on pedestrian detection using LIDAR, monocular vision, and the combination of both sensors, in the context of IV/ITS applications. The theoretical basis used throughout this work and the LIPD dataset are described in Chapter 3. Chapter 4 specifically focuses on the LIDAR data processing...
system, and Chapter 5 presents the vision-based system. The combining strategies and schemes for information fusion are presented in Chapter 6. The various methods and algorithms proposed here for pedestrian detection and classification, using LIDAR and monocular camera information, are compared and evaluated. The experimental results, performed on the LIPD dataset, are presented and discussed in Chapter 7. Finally, Chapter 8 concludes this thesis.
Chapter 2

State of Art

This chapter gives a survey on pedestrian detection using in-vehicle sensors for ITS, IV and active safety applications. In particular, the research works covered in this chapter consider monocular camera(s) (color or black white) and 2D Sick-LMS or 4-layers Ibeo automotive laserscanners as primary sensors for pedestrian detection in urban scenarios.

2.1 LIDAR-based systems for pedestrian detection

Pedestrian detection systems based on data from 2D LIDAR, in the field of intelligent vehicles for urban scenarios, have applications varying from detection, tracking, classification, collision warning, and high-level decision systems. The particular purpose of this section is to review LIDAR-based systems for pedestrian detection and classification in urban environment; in Table 2.1 a summary of the related work is provided.

2.2 Vision-based systems for pedestrian detection

There are many scientific studies on pedestrian and/or public (group of pedestrians) detection using monocular cameras, with applications in the field of robotics, advanced human interfaces, automotive safety, intelligent vehicles, among others. In
Table 2.1: Outline of some related work on LIDAR-based systems for pedestrian detection/classification in urban scenarios.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Technology and system spec.</th>
<th>Comments and characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Dietmayer et al., 2001]</td>
<td>Ibeo single-layer Laser scanner prototype, with angular resolution of 0.25° and range up to 100m.</td>
<td>It is one of the seminal works on object detection and classification using laser scanner. Although, the referred work proposed a system to detect, classify and to track a set of objects (vehicles, trucks, bicycles, small objects, and so on), pedestrian is among the classes of interest. Basic segmentation and classification methods were presented, and a free-moving model under Gaussian and linear conditions were used to model the objects. Linear discrete KF was employed as stochastic filter. This work, published in 2001, provided much of the basis for further developments in this researching topic.</td>
</tr>
<tr>
<td>[Fuerstenberg et al., 2002]</td>
<td>Ibeo LD ML automotive laser scanner, four layer and range up to 50m. It is the precursor of the Alasca XT and Lux sensor products.</td>
<td>The proposed system has four models: sensor, road, ego-vehicle and object models, where the later is focused on pedestrian detection and tracking. Typical issues are highlighted in this work: data association uncertainties, segmentation mistakes, time-processing and memory limitations, and various realistic aspects of such application. Approximately ten papers have been published by the Ibeo researching group between 2001 and 2004 in object detection, tracking and classification in the context of urban scenarios [Streller and Dietmayer, 2004].</td>
</tr>
<tr>
<td>[Mendes et al., 2004]</td>
<td>Sick LMS200 single layer laser, mounted in the Yamaha ISR-UC electrical vehicle</td>
<td>Sick LMS200 single layer laser sensor of the Alasca XT and Lux sensor products.</td>
</tr>
<tr>
<td>[Premebida et al., 2006]</td>
<td>Two datasets were built using data from a Sick LMS200 and an Ibeo Alasca XT LIDAR for pedestrian classification performance analysis.</td>
<td>This work focuses in pedestrian classification (positive) using only LIDAR features against non-pedestrians urban objects: vehicles, trees, posts, walls, sidewalks, road, etc. Two datasets with 18-dimensional feature space are used to compare 5 classification methods.</td>
</tr>
<tr>
<td>[Zhao et al., 2009]</td>
<td>Two test-bed vehicles are used, one equipped with a LMS201 and the other with a LD-OEM Sick laser</td>
<td>A system has been proposed for ego-vehicle localization and object detection in urban environment. For all laser-segments, the object speed is estimated and, for pedestrian classification, a set of features are used to model the pedestrian-likelihood. MAP is used as classifier, where the prior was obtained using supervised training process [Fayad and Cherfaoui, 2007] is another work that also uses Sick laser for object, and pedestrian, detection in urban scenarios.</td>
</tr>
<tr>
<td>[Gidel et al., 2010]</td>
<td>Ibeo Alasca XT, 4-layer LIDAR, mounted in the center of the bumper area of a commercial vehicle.</td>
<td>Pedestrian detection is performed in each laser-layer using segmentation and line-extraction algorithms; each layer is then fused and pedestrians are filtered-out by means of a non-parametric kernel-pdf based algorithm, i.e., pedestrians are detected based on the ML estimator which is modeled using a Parzen (non-parametric) kernel. Temporal tracking is carried out using a decentralized approach based on PF. Experiments on an urban environment are presented and a good detection rate is obtained when pedestrians are up to 20m.</td>
</tr>
<tr>
<td>[Oliveira and Nunes, 2010]</td>
<td>A Sick LMS200 laser scanner, with range limited to 8m, mounted in the ISR RobotCar vehicle.</td>
<td>LIDAR-based object detection is accomplished by means of not only a featureless approach, but also inferring context-aware relations of object parts. A coarse-to-fine segmentation based on skeleton random graph is proposed; after segmentation, each segment is labeled, and scored by a Procrustes analysis. After defining the sub-segments of each object, a contextual analysis is in charge of assessing levels of intra-object or inter-object relationship, ultimately integrated into a Markov logic network.</td>
</tr>
</tbody>
</table>

In particular, this section provides a summary review of the most recent works that survey vision-based approaches used for pedestrian detection for automotive and intelligent vehicles applications i.e., in the IV and ITS community, which are summarized
2.3. PEDESTRIAN DETECTION USING LIDAR AND MONOCULAR CAMERA

Table 2.2: List of recent surveys on vision-based systems for pedestrian detection/classification in urban scenarios.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Comments and characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Gavrila, 2001]</td>
<td>This is probably the first article that surveys state-of-art research works on pedestrian detection focused on IV and ITS applications. Although most of the related works use vision-based systems, especially monocular and stereo cameras, some preliminary works using radar and laser scanners are also referenced. Hence, this article provides a relevant review of the emerging pedestrian detection solutions at the beginning of this century.</td>
</tr>
<tr>
<td>[Gandhi and Trivedi, 2006] [Gandhi and Trivedi, 2007]</td>
<td>These papers, especially the latter, describe the recent research on pedestrian safety, a scientific field with the majority of solutions using cameras. The second article, in particular, presents a comprehensive review of the sensory technology, approaches and methods, and challenges surrounding the active safety system on pedestrian detection and collision avoidance. In summary, it is an indispensable reading.</td>
</tr>
<tr>
<td>[Dollar et al., 2009] [Geronimo et al., 2010] [Enzweller and Gavrila, 2009]</td>
<td>A recent review of works dealing with pedestrian protection are surveyed, discussing the efforts, achievements and challenges in this domain. Moreover, some contributions are presented: for instance, vision-based datasets are available as benchmarking in [Dollar et al., 2009] and [Enzweller and Gavrila, 2009]. Furthermore, these papers, together or even separately, cover basically all the recent efforts and research programs dealing with this problem in the context of IV and ITS applications, emphasizing the camera-based solutions. Currently, [Enzweller and Gavrila, 2009] possibly represents the state of art on monocular camera-based system for pedestrian detection.</td>
</tr>
</tbody>
</table>

Although the great effort on pedestrian detection using vision devoted by the IV and ITS community in the past 10 years, for instance Gavrila’s surveys [Gavrila, 2001] and [Enzweller and Gavrila, 2009], much of the issues and challenges in the past still remain to be solved for practical and effective applications. In short, the main open problems in this domain are: to reduce the significant computational effort and the usual huge algorithmic complexity; to obtain a high detection rate with a reasonable low false alarm; to have a robust solution under weather and environmental variations; to handle with the pedestrian detection in situations of partial occlusions; to have a reliable system which accomplish safety standards.

2.3 Pedestrian detection using LIDAR and monocular camera

This section covers methods that combine LIDAR and monocular camera data, in a complementary or redundant way, to detect pedestrians in urban-like scenarios. While in the two previous sections pedestrian detection solutions were considered separately for each sensor, here the sensor data are considered jointly in such way that the final fusion system would achieve better detection performance than the
single based sensors systems or, at least, the solution should be less complex. Table 2.3 surveys some significant related works on pedestrian detection using LIDAR and monocular visible-spectrum cameras.

Table 2.3: Research on LIDAR and vision data fusion systems for pedestrian and on-road objects detection/classification in urban scenarios.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Vision system</th>
<th>LIDAR system</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Mahlbach et al., 2000]</td>
<td>Monocular color camera. An AdaBoost, using Haar-like features, processes the images delimited by the ROIs.</td>
<td>Multi-layer LIDAR. The objects detected by the LIDAR define the ROI in the image plane.</td>
<td>The paper is focused on the non-proposed method designated “cross-calibration”. The idea behind this method is to facilitate the correspondence between the LIDAR space and the image plane projection.</td>
</tr>
<tr>
<td>[Eames et al., 2000]</td>
<td>Monocular grey-scale camera. A Convolutional Neural Networks classifier is used.</td>
<td>Multi-layer LIDAR. Objects detected in the LIDAR-space are projected to the image plane (ROI) using perspective mapping (intrinsic-extrinsic parameters are obtained).</td>
<td>A reduced flat-world is used to model the road. Some comparisons are presented considering some variations of the system: vision, vision and LIDAR-based ROI, flat model, and non-restricted road model.</td>
</tr>
<tr>
<td>[Cheng et al., 2007]</td>
<td>Two monocular color cameras. One camera for lane detection and other for vehicle detection using HOG features.</td>
<td>Single-layer LIDAR and Radar. Using ERF, local tracking techniques are used in LIDAR and Radar reference system and fused to form a global tracking approach.</td>
<td>The fusion strategy using LIDAR and radar information, for on-road object detection, constitutes the focus of this paper, with emphasis on a local and global tracking approach. The vision-based obstacle detection system uses range information available from global tracks, in the form of ROI, as a decision making system.</td>
</tr>
<tr>
<td>[Huang et al., 2007]</td>
<td>Monocular color camera. A multi-layer-SVM classifier is used to verify the hypothesis candidates inside the ROIs.</td>
<td>Single-layer LIDAR. The entities detected by the LIDAR generate hypothesis candidates, which are projected in the image plane (ROI) by means of perspective mapping.</td>
<td>The image plane is subdivided in five areas, where different trained SVMs are employed to classify the vehicles. A comparison study between single-SVM and 5-SVM approaches is presented.</td>
</tr>
<tr>
<td>[Daviskell et al., 2007]</td>
<td>Monocular color camera. Conditional Random Fields (CRFs) trained with Virtual Evidence Boosting (VEB).</td>
<td>Single-layer LIDAR. Geometrical information is processed from the LIDAR data to estimate/classify the objects as vehicles or non-vehicles.</td>
<td>To deal with the problem of object scale variations in the images, the range information provided by the LIDAR is used during the CRFs classification. The classification method was evaluated and compared with several features: geometrical (from laser data), visual (color and texture) and combination of both. The CRF and a LogitBoost classifier were also compared.</td>
</tr>
<tr>
<td>[Spinello and Siegwart, 2008]</td>
<td>HOG-SVM classifier based on monocular color images.</td>
<td>A multi-layer LIDAR (Ibeo Alasca XT) is employed to detect on-road objects whose positions are projected into the image plane.</td>
<td>The object’s position is detected by the LIDAR and the vision-based system classifies the detected objects as pedestrians or non-pedestrians. A Bayesien decomposed expression is used as reasoning fusion rule.</td>
</tr>
<tr>
<td>[Pangar et al., 2008]</td>
<td>An AdaBoost classifier, trained with Haar-like features, is used to classify pedestrians.</td>
<td>A Ibeo Alasca XT LIDAR is employed for object segmentation, tracking and detection.</td>
<td>The speed, estimated during the tracking process, and the vision score-based likelihood is fused in a Bayesian framework using an autoregressive (AR) formalism to model the observations.</td>
</tr>
<tr>
<td>[Boegli et al., 2009]</td>
<td>NIR camera with 752x480 pixels and Haar-like features based AdaBoost classifier is used.</td>
<td>Standard single laser scanner (LMS 291) is employed for ROI generation in the image plane.</td>
<td>In the proposed system the LIDAR provides a list of ROIs in which a pedestrian may appear, while the camera is employed to detect and classify potential pedestrians in the ROIs. It is a system designed for pedestrian detection in specific situations, designated as critical areas, in urban scenarios.</td>
</tr>
<tr>
<td>[Oliveira et al., 2010]</td>
<td>Monocular camera. HOG and LRF features are classified by a SVM and a MLP, and combined by means of fuzzy integral.</td>
<td>Side LMS-300. 2D range pixels are clustered in meaningful segments using a coarse-fine approach, and then labeled using a template matching procedure, which is based on Procrustes analysis.</td>
<td>Feature fusion and vision has been accomplished by two approaches: (i) independent integration of sensors-driven features and classifiers, and (ii) a ROI is found by laser segmentation and an image classifier is used to name the projected ROI. The fusion approach is based on ‘semantic’ information.</td>
</tr>
</tbody>
</table>

LIDAR and camera fusion can be performed using centralized or decentralized configuration as discussed in [Premebida et al., 2009a]. In the first, a LIDAR is used as primary detection sensor generating ROIs which are projected onto the image. The projected ROIs can be used as simply potential candidates for pedestrian detection without any additional information, or can be accompanied by a probability score or ‘confidence’ value. Conversely, in the decentralized approach, data from each sensor
are processed independently and the final decision is obtained by means of fusion rules or combining methods [Premebida et al., 2009c]; besides, multistage schemes (or cascade ensembles) can be arranged to combined information from LIDAR and camera in the same structure. For instance, in [Ludwig et al., 2011], a LIDAR-dependent processing stage was used to decrease the complexity and the time processing of a cascade trained with image-based classifiers.

2.4 Summary

Pedestrian detection, in the context of ITS and/or IV, is a researching area that arose about a decade ago. More recently, pedestrian detection gained wider attention due to their applicability in safety systems for the automotive industry with most of the techniques and methods coming from areas such as robotics, computer vision for autonomous navigation, and machine learning. Although more than 10 years of development, there are many topics that were not fully exploited in this field; hence, it is a researching area with many open problems to be solved. The related works covered in this chapter were restricted to two technologies, LIDAR and monocular visible-spectrum camera devices; consequently, three possibilities were considered for pedestrian detection systems: LIDAR-based systems; monocular image-based systems; and data fusion (LIDAR and vision). use LIDAR for detection and vision for classification. Although it is possible to obtain a pedestrian classification and detection solution based solely on monocular camera, the most recent works suggest that such systems should have a LIDAR for detection (position and dynamics estimation), and a camera for classification.

It is notorious that image-based (vision) systems appear at the top of the proposed solutions so far, and probably it will be for the next on-going and future solutions. However, recent approaches tend to integrate more information, from other sensor technologies, to pursue a definitely, or at least acceptable, solution to attend realistic applications [Gandhi and Trivedi, 2007] [Broggi et al., 2009] [Geronimo et al., 2010].
Chapter 3

Background and Dataset

The next sections in this chapter serve as a prelude to the pedestrian detection problem, aiming to provide sufficient information to convey the main concepts and algorithms necessary to understand the aforementioned problem and, concurrently, to provide the basis of the proposed solutions. The purpose of this chapter is to introduce the main concepts, techniques and the methodology that constitute the fundamental blocks used throughout this thesis; furthermore, in this chapter the LIPD dataset is introduced in detail. Once the general system architecture was explained with the aid of a functional block diagram representation, depicted in Figure 1.2, followed by a short description of the main modules in Section 1.3, this chapter will cover in a more formal way the following topics: (1) characteristics of the sensors, (2) sensor calibration, (3) object detection framework, (4) object classification concepts. Moreover, the LIPD dataset is detailed in the last section. In summary, the main goals of this chapter are: (1) to convey the knowledge basis (prerequisites) necessary for the development and usage of algorithms that operate in a PDS; (2) to give the fundamental characteristics of the sensors used in the vehicle platform; and (3) to introduce relevant aspects of the LIPD dataset that, throughout this work, gives support to most of the empirical analysis.
3.1 LIDAR and camera characteristics

A LIDAR, or laserscanner, is an active optical laser-based range and bearing measurement sensor. The laser used in this work is an Alasca XT, manufactured by Ibeo; it is a four-layer laser rangefinder that uses the time-of-flight measurement principle. The angular FOV, angular resolution, and frequencies of data delivery vary according to parameters set up by the user. A monocular camera, using a standard C-mount lens (1.4/6.5mm), has been employed to capture color-based image frames from the scenes. It is a FireWire camera, model Guppy, manufactured by Allied, equipped with a CCD sensor, capturing VGA images in Bayer standard (a post-processing routine was used to transform the images to RGB format). Since the camera has CS-mount standard, a 5mm ring was necessary to use the lens properly. The LIDAR and the camera were mounted in a “rigid” platform on the frontal part of the vehicle, with the camera just above the laser, as depicted in Fig. 1.3. The data stream from the LIDAR is sent to the host PC by means of an Ethernet-based Ibeo processing unit\(^1\). The acquired scans consist of raw range-data that are treated as clouds of points. The images are acquired using openCV-based libraries in a sequential way having the Ibeo API thread priority over the process. The images were transformed to RGB-standard for off-line processing purpose, i.e., for feature extraction and pedestrian detection. The characteristics of the sensors are summarized in Table 3.1.

\begin{table}[h]
\centering
\begin{tabular}{|l|l||l|l|}
\hline
\multicolumn{2}{|c|}{Camera} & \multicolumn{2}{|c|}{LIDAR} \\
\hline
Parameter & Description or value & Parameter & Description or value \\
\hline
Manufacturer & Allied & Manufacturer & Ibeo Alasca XT \\
Model & Guppy & Layers & 4 parallel layers \\
Field of View & 66x40 (horiz. x vert.) & used FOV & 120° \\
Sensor & CCD & Range res. & 4 cm \\
Resolution & 640 x 480 & used Horiz. Res. & 0.125° \\
Used fps & 30 & Used freq. & 12.5Hz \\
Color & RGB Bayer format & Vert. res. & (-1.6°,-0.8°,0.8°,1.6°) \\
Focal length & 6.5 mm & Range & 0.3 - 200m \\
Lens & Pentax C-mount & Cover & water-proof chamber \\
\hline
\end{tabular}
\end{table}

\(^1\)A data acquisition algorithm, based on an Ibeo Linux-API, was developed to work with an Arcnet-PCMCIA adapter before the acquisition of such Ibeo processing unit.
3.2 LIDAR and camera calibration

The methodology adopted to calibrate the laser-camera setup consisted in three fundamental steps: collecting a set (more than fifteen) of laser and image synchronized frames; obtaining the camera intrinsic and extrinsic parameters; and then estimating the laser-camera coordinate transformation matrix. The extrinsic and intrinsic parameters of the camera were calculated using [Bouguet, 2007], whose intrinsic values are summarized in Table 3.2. Once these parameters were known, the referred laser-camera pose was estimated using [Zhang and Pless, 2004] calibration method. These parameters were assumed to be constant during the experimental dataset collection, as considered in most of the works in the PDS context [Douillard et al., 2007], [Spinello and Siegwart, 2008], [Gidel et al., 2010].

### Table 3.2: Intrinsic camera parameters: $K$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal length</td>
<td>$f_c : [623.731 ; 624.595]$</td>
</tr>
<tr>
<td>Principal point</td>
<td>$c_c : [360.876 ; 256.895]$</td>
</tr>
<tr>
<td>Distortion coefficients</td>
<td>$k_c : [-0.304 ; 0.106 ; -0.00032 ; -0.000027 ; 0.0]$</td>
</tr>
<tr>
<td>Uncertainties</td>
<td>$f_{c\text{error}} : [2.819 ; 2.836]$</td>
</tr>
<tr>
<td></td>
<td>$c_{c\text{error}} : [2.065 ; 1.859]$</td>
</tr>
<tr>
<td></td>
<td>$k_{c\text{error}} : [0.0071 ; 0.028 ; 0.00063 ; 0.00046 ; 0.0]$</td>
</tr>
</tbody>
</table>

Points in the camera reference system $P^C = [P_C^X, P_C^Y, P_C^Z]$ can be transformed into the laser coordinate system $P^L = [P_L^X, P_L^Y, P_L^Z]$ using the transformation $P^L = R_C^L P^C + T_C^L$, where $R_C^L$ is the 3x3 orthonormal rotation matrix representing the camera’s orientation relative to the laser and $T_C^L$ is the 3-dimensional vector representing the relative position. The method described in [Zhang and Pless, 2004] was used to estimate $R_C^L$ and $T_C^L$, given in (3.3). The transformation between a point in the laser coordinate system $P^L$ to a point in the camera reference system $P^C$ is obtained by $P^C = (P^L - T_C^L)/R_C^L$. The 3D point $P^C$ is normalized and the distortion coefficients, given in Table 3.2, are applied in order to obtain $X_n$ [Heikkila and Silven, 1997]. Finally, the pixel coordinates in the image plane is calculated as follows.

Denoting a point in the image plane by $P^I = [u, v]$, where $u$ and $v$ are pixel
coordinates, and considering a pinhole model, the coordinates of $P^I$ are calculated:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X_n(1) \\ X_n(2) \\ 1 \end{bmatrix}$$

(3.1)

with the camera matrix $K$ given by

$$K = \begin{bmatrix} fc(1) & \alpha_c fc(2) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix}$$

(3.2)

where $\alpha_c$ is the skew coefficient.

With the LIDAR data it is only possible to obtain the horizontal limits of the object position in the image. If it is assumed that the vehicle moves on a “flat” surface, and knowing the distance from the laser to the ground, it is easy to calculate the bottom limit of the ROI. The top limit of the ROI was estimated using the distance to the object and considering 2.5m as the maximum height of a pedestrian. The following matrix, necessary to make a rigid correspondence between the laserscanner and the camera reference system, was obtained:

$$R_L^C T_L^C = \begin{bmatrix} 0.999 & -0.014 & -0.009 & 11.92 \\ 0.014 & 0.999 & 0.026 & -161.26 \\ 0.009 & -0.027 & 0.999 & 0.78 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(3.3)

where the translational vector components are in mm.

One can ask why does not simply use IMU data to infer the vehicle inclination and then to avoid the constant flat surface assumption? The main reason is that, even if the vehicle inclination is known, information about the road and the scene ahead the vehicle should also be known. A feasible solution could be made on the employment of environment information, retrieved by a GPS-based semantic map of the roads, where the vehicle has been driven.
3.3 Pedestrian detection ruled by probability theory

In general terms, an event can be defined as a collection (set) of outcomes, or realizations, of a random process. In the particular case treated in this work, the set of outcomes is represented by all possible objects present in urban environments (e.g., pedestrians (adults, children, groups of persons), vehicles, bikes, posts, lamps, trees, hydrants, walls, and barriers,) that are perceived (sensed) by the monocular camera and/or by the LIDAR: that is the "complete" space of detection that delimits the problem. Restricting the problem to the particular case of dichotomy, the event of interest $A$ is defined by a set with two elements: positives (pedestrians) and negatives (all the remaining possible elements that pertain to the random process). Thus, an event is said to have occurred if the outcome, a measurement or realization, is one of the elements pertaining to $A$. Let $\Omega$ be the sure event in the complete set of realizations, i.e., any object or entity possible of detection by the sensors, then the probability of the realization of $A$ is a number (measurable variable) that satisfies the three axioms of probability [Martinez and Martinez, 2002]. To simplify the notations, it preferable to define instead of $A$ two mutually exclusive events: PED = \{"pedestrian"\} and nPED = \{"non-pedestrian: i.e., any other urban detectable object"\}. The hypothesis of the outcome be a PED is denoted by $H_1$, and the hypothesis of being nPED is denoted by $H_0$. Using a more simplified notation, the two events (or classes) of interest are also denoted by $\omega_0$ and $\omega_1$ therefore, the set $\Omega$ of events has two elements: $\Omega = \{\omega_i : i \in \{0, 1\}\}, \omega_0 = \text{nPED}, \omega_1 = \text{PED}$. The probabilistic framework used throughout this thesis is based on the Bayesian statistic. In particular, the following interpretations and concepts of probabilities are considered:

Prior probability (or a priori) $P(\omega_i)$: it is the probability of occurrence of an event (PED or nPED) without any measurement realization; that is the measurement unconditional probability. The prior reflects the (possibly subjective) initial degree of belief regarding the occurrence of an event.

Likelihood $p(z|\omega_i)$: is the pdf of the measurements $z$ conditioned on a given event $\omega_i$. It is directly proportional to the conditional probability that a measurement has occurred given the event. The probability density function (pdf) dictates the values that can be assumed by a continuous random variable (real-valued) according to the outcome of the random process, whereas pmf is used for discrete-valued random variables.
**Posterior probability** (a posteriori) $P(\omega_i|z)$: it is the conditional probability that an event is true given the measurement. In the universe of the Bayesian theory, these variables are related in the classic Bayes' formula:

$$P(\omega_i|z) = \frac{p(z|\omega_i)P(\omega_i)}{p(z)}$$

(3.4)

where, $p(z) = \sum_{i=1}^{2} p(z|\omega_i)P(\omega_i)$.

### Pedestrian detection as a decision problem

Consider the pedestrian detection as a hypothesis problem between two mutually exclusive hypotheses: the hypothesis of non-pedestrian $H_0 : \omega_0$, and the alternate hypothesis (the object is a pedestrian) $H_1 : \omega_1$, where $\omega_i$ is the variable that defines the class (object identity, category), and $\omega_0,1$ are the values taken by $\omega$, that is, the class ‘label’ or object assignment class.

The type I error probability, the false positive or false alarm error, is $P_{eI}$ while the type II error probability, false negative (FN) or missing error, is $P_{eII}$. The power of the test is defined as follows:

$$\pi \triangleq P\{H_1|\omega_1\} = 1 - P_{eII}$$

(3.5)

and it measures the test’s capability to discern $H_1$ when $\omega_1$ is true instance i.e., the detection probability. The decision as to which hypothesis to accept is made based on a set of observations $z$, although it is theoretically possible to take a decision based on the a priori probability, that is, without current observation. The observations enter in the system as likelihoods, conditioned on $\omega_0$ and $\omega_1$, that are considered to be known. According to the Neyman-Pearson Lemma (cited in [Duda and Hart, 1973]), the optimal decision, in the sense of minimizing the probability of type II error, or maximizing (3.5), subject to a given (maximum) probability of type I error, is based on the likelihood ratio

$$\Lambda(H_1, H_0) = \frac{p(z|H_1)}{p(z|H_0)} \geq H_1 T_\Lambda$$

(3.6)

the threshold $T_\Lambda$ is such that

$$P\{\Lambda(H_1, H_0) > T_\Lambda|H_0\} = P_{eI}$$

(3.7)
that is, "accept H1" if \( \Lambda \) exceeds \( T_\Lambda \) and "accept H0" if \( \Lambda \) is below \( T_\Lambda \). Or, in another form, choose H1 if \( p(z|H1) > T_\Lambda \), and accept H0 if \( p(z|H0) > T_\Lambda \), knowing that \( p(z|H0) + p(z|H1) = 1 \). With a structure similar to (3.6), is the maximum \textit{a posteriori} decision rule, which leads to identical results of (3.6) if the prior is uniform.

**General decision problem**

Other forms of decision rules are also used, based on different decision criterion: standard fusion rules (e.g., average, maximum, minimum), normalized product rule, majority vote, and more complex methods like ensemble of classifiers (these methods are discussed in Chapter 6). In a more general way, valid for single and for ensemble of classifiers, with output \( \in \mathbb{R}^1 \), the final decision follows the form of the majority vote: \( \hat{\omega} = \text{sign}(\sum_{i=1}^{NC}(\hat{\omega}_i)) \in \{-1,0,1\} \), where \( \hat{\omega}_i \) is the output of a given \( i^{th} \) classifier, and \( NC \) is the number of classifiers in an ensemble. Thus, if \( \hat{\omega} \leq 0 \) the class \( \omega_0 \) is chosen, otherwise it is said \( \hat{\omega} = \omega_1 \).

### 3.4 Classification methods

A classifier is ultimately a decision function \( F(.) \) that maps a \( N \)-dimensional feature vector \( z \), defined in a feature space, onto the set of possible classes \( \Omega = \{\omega_0, \omega_1\} \), i.e., \( F(.) : \mathbb{R}^N \rightarrow \Omega \). The classification methods used throughout the development of this work are: Linear Discriminant Analysis (LDA), Naive Bayes classifier (NBC), Gaussian Mixture Model-based classifier (GMM), Support Vector Machine (SVM), and Artificial Feed-forward Neural Network (ANN). These classifiers, described in the sequel, were employed as decision functions to separate the feature space in classes, which is the usual case, and also as trainable rules for classifiers fusion, which is detailed in Chapter 6.

**Linear Discriminant Analysis: LDA**

Let us consider \( w \) a vector of adjustable gains and \( z_c \) the feature vector, belonging to a given class \( (\omega_c, c = 1, 2) \), with mean and covariance given by \( \mu_c \) and \( \Sigma_c \) respectively. The linear combination \( w \cdot z_c \) has mean \( w \cdot \mu_c \) and covariance \( w^T \Sigma_c w \). The ratio,
$J(w)$, of the variance between the classes, $\sigma^2_b$, by the variance within the classes, $\sigma^2_w$, is a suitable measure of separation between these two classes:

$$J(w) = \frac{\sigma^2_b}{\sigma^2_w} = \frac{(w \cdot (\mu_2 - \mu_1))^2}{w^T (\Sigma_1 + \Sigma_2) w} \quad (3.8)$$

To obtain the maximum separation between classes, one has to find the vector $w$ which solves the optimization problem

$$\max_w J(w) \quad (3.9)$$

whose solution is given by

$$w = (\Sigma_1 + \Sigma_2)^{-1}(\mu_2 - \mu_1) \quad (3.10)$$

Finally, to find the plane that best separates the classes, the expression $w^T \mu_1 + b = -(w^T \mu_2 + b)$ has to be solved for the bias $b$.

**Naive Bayes: NBC**

Based on the consideration that each feature $z_i, i = 1, \cdots, N$, of a feature vector $z_c$ with $N$ elements, is statistically independent of each other, the probability density function (pdf) that characterizes the object class $c$ is modeled as the product of each feature-model pdf. In the Naive Bayes classifier used in this work, the pdf of each class is modeled by a unidimensional Gaussian with parameters $\theta_i = [\mu_i, \sigma_i]$, expressed as:

$$p(z_i|\omega_c, \theta_i) = \frac{1}{\sigma_i \sqrt{(2\pi)}} \exp\left[-\frac{(z_i - \mu_i)^2}{2\sigma_i^2}\right] \quad (3.11)$$

Thus, the combined likelihood is expressed by:

$$p(z_c|\omega_c) \propto \prod_i p(z_i|\omega_c, \theta_i) \quad (3.12)$$
3.4. CLASSIFICATION METHODS

Gaussian Mixture Model classifier: GMM

The GMMC tries to estimate the Likelihood of the feature-vector $z_c$, given the object class $\omega_c$, as a mixture of $M$ Gaussians, defined by a set of parameters $\Theta_c = (\rho_c; \mu_c; \Sigma_c)$, where $\rho_c$ is the weight-vector, such that $\sum_{m=1}^{M} \rho_c(m) = 1$, $\mu_c$ is the mean-vector, and $\Sigma_c$ is the covariance matrix of a given class. The Likelihood is modeled as:

$$p(z_c|\omega_c, \Theta_c) \propto \exp\left[-\frac{1}{2}(z_c - \mu_c)^T (\Sigma_c)^{-1} (z_c - \mu_c)\right] \sqrt{(2\pi)^M}$$  \hspace{1cm} (3.13)

To maintain the likelihoods consistent in the interval $[0, 1]$, the expressions (3.12) and (3.13) have to be normalized. For the case where $c = 1$ i.e., the predicted class is $\hat{\omega} = \omega_1$, the normalized Likelihood is given by:

$$p(z|\hat{\omega} = \omega_1) = \frac{p(z|\omega_1, \Theta_c)}{p(z|\omega_1, \Theta_c) + p(z|\omega_0, \Theta_c)}$$  \hspace{1cm} (3.14)

Support Vector Machine: SVM

The SVM classifier uses the SRM principle, which is based on the statistical learning theory developed by Vapnik [Vapnik, 1998]. In the binary classification problems, which is the case discussed here, the objective of the SVM classifier is to find the optimal separating hyperplane, defined in the feature space $\mathbb{R}^N$, subject to a maximum margin. Considering a discriminant function of the form

$$g(z) = \xi^T \phi(z) + \xi_0$$  \hspace{1cm} (3.15)

with the binary decision rule

$$\begin{align*}
  if \quad g(z) > 0 & \Rightarrow z \in \omega_1(y_i = +1) \\
  if \quad g(z) < 0 & \Rightarrow z \in \omega_0(y_i = -1)
\end{align*}$$  \hspace{1cm} (3.16)

The SVM training procedure determines the maximum margin solution through

---

2 Due to context-based considerations in this work, the number of Gaussian was restricted to a maximum of 5 components.
the maximization of a Lagrangian $L_D$; in dual form it becomes [Webb, 2002]

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \phi^T(z_i) \phi(z_j)$$ \hspace{1cm} (3.17)

where $y_i = \pm 1, i = 1, \cdots, n$, are class indicator values and $\alpha_i$ are Lagrange multipliers satisfying $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$, for a parameter $C$ (of regularization). In plain words, the solution for $\xi$ is

$$\xi = \sum_{i \in SV} \alpha_i y_i \phi(z_i)$$ \hspace{1cm} (3.18)

where $SV$ corresponds to the set of support vectors represented by an integer index. Finally, the decision function is performed according to

$$g(z) = sign(\sum_{i \in SV} \alpha_i y_i \phi^T(z_i) \phi(\tilde{z}) + \xi_0)$$ \hspace{1cm} (3.19)

where $\tilde{z}$ is an observed sample i.e., a given test feature vector. The solution for the bias $\xi_0$, as well as more details about the SVM training can be found in [Webb, 2002] [Duda. et al., 2001] [Theodoridis and Koutoumbas, 2003].

Artificial Neural Network: ANN

In this thesis a multi-layer perceptron (MLP) neural network is used. The MATLAB function newff was used to create the network architecture. All the experiments were conducted using a MLP network with: one hidden layer, up to four neurons, logsig activation function for the hidden layers, number of validation equal to six, and gradient descent (trainigndm) as backpropagation training function. Detailed explanations regarding feed-forward neural networks for pattern recognition is given in [Bishop, 1995] and [Duda. et al., 2001].

Classification performance metrics

The classification performance is evaluated in terms of the error, which will be assessed using accuracy (Acc), Balanced Error Rate (BER), and Area Under ROC
3.4. **CLASSIFICATION METHODS**

(AUC); moreover, to support graphical analysis, the Receiver Operating Characteristic curves (ROC) will be used. The terminology and definitions shown in Table 3.3 were considered in this work.

<table>
<thead>
<tr>
<th>Table 3.3: Terminology and definitions for performance metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>True positive (TP)</strong> if $\hat{\omega}_i = \omega_1$ when the true instance is $\omega_1$</td>
</tr>
<tr>
<td><strong>True negative (TN)</strong> if $\hat{\omega}_i = \omega_0$ when the true instance is $\omega_0$</td>
</tr>
<tr>
<td><strong>False positive (FP)</strong> if $\hat{\omega}_i = \omega_1$ when the true instance is $\omega_0$</td>
</tr>
<tr>
<td><strong>False negative (FN)</strong> if $\hat{\omega}_i = \omega_0$ when the true instance is $\omega_1$</td>
</tr>
<tr>
<td><strong>False pos. rate (FPR)</strong> $FPR = FP/nn$, where $nn$ is number of negatives</td>
</tr>
<tr>
<td><strong>False neg. rate (FNR)</strong> $FNR = FN/np$, where $np$ is the number of positives</td>
</tr>
<tr>
<td><strong>True pos. rate (TPR)</strong> $TPR = TP/np$</td>
</tr>
<tr>
<td><strong>True neg. rate (TNR)</strong> $TNR = TN/nn$</td>
</tr>
<tr>
<td><strong>Error rate</strong> The error rate is calculated as: $(FP+FN)/(np+nn)$</td>
</tr>
<tr>
<td><strong>Accuracy (Acc)</strong> $Acc = 1 - Error rate = (TP + TN)/(np+nn)$</td>
</tr>
<tr>
<td><strong>Balanced Error Rate</strong> $BER = 0.5(FPR+FNR) = 1 - (TPR+TNR)/2$</td>
</tr>
<tr>
<td><strong>Area under ROC (AUC)</strong> $\int_{i=0}^{1} ROC(i)di \approx \sum_i (FPR(i+1)-FPR(i))(TPR(i+1)+TPR(i))$</td>
</tr>
</tbody>
</table>

Although Acc and Error rate are usual measures of classification performance, they do not reflect a consistent performance of classifiers in imbalanced dataset. For example, supposing a dataset with 990 samples of *negatives* and 10 samples representing the class *positives*, a classification method can achieve an accuracy of 99% by simply classifying all as *negatives*, nevertheless its capability of class discrimination is very low; this justify the usage of other metrics such as BER and AUC.

**K-fold cross validation**

Cross validation is a very common technique for selecting the classifier’s parameters, and also for estimating the generalization capability of classification methods. More specifically, in K-fold cross validation a given set is subdivided in K subsets, where K-1 subsets are used as training set and the remaining subset is used as validation/testing set. The cross validation process is repeated K times, hence each K subset is used only one time as testing set. The average on the classification performance metrics, calculated on the K testing subsets, are used to select the classifiers’ parameters. Thus, representing the input set as $X : |X| = n$, K-fold cross validation starts by separating $X$ in K subset: $X = \{X_i : i = 1, \cdots , K/|X_i| = \frac{n}{K}\}$. For $i = 1$ to $K$, $X_i$ is used for testing and the remaining subsets $\{X_j : j \neq i\}$ are used
for training. The performance result, e.g. using \( \text{Acc} \), obtained for the testing subsets (\( \text{Acc}_i, i = 1, \cdots, K \)) are averaged to produce the final performance evaluation: \( \text{Acc} = \frac{\sum_i \text{Acc}_i}{K} \).

### 3.5 Risk minimization

Let us define \( \Omega = \{\omega_1, \cdots, \omega_k\} \) the finite set of classes of interest (in particular \( k = 2 \)), and \( \hat{\omega}_i, i \in \{1, 2\} \) denoting the predicted class by a given classifier \( \mathcal{F}(\cdot) \). In this work, the loss function \( L(\hat{\omega}_i | \omega_k) \) is assumed uniform, i.e., if an object is misclassified a unit cost is considered, and zero cost when the object is correctly classified, hence

\[
L(\hat{\omega}_i | \omega_k) = 1 - \delta(i, k) = \begin{cases} 
0 & \text{if } i = k \\
1 & \text{elsewhere}
\end{cases}
\] (3.20)

where \( \delta(i, k) \) is the Kronecker delta function.

#### Bayesian risk minimization

Considering the adopted loss function (3.20), the conditional risk is defined by

\[
R(\hat{\omega}_i | z) = \sum_{k=1,k \neq i} P(\omega_k | z) = 1 - P(\hat{\omega}_i | z)
\] (3.21)

The minimization of this risk is equivalent to maximize the posterior probability \( P(\hat{\omega}_i | z) \) [Duda. et al., 2001]. Therefore, the Bayes decision function, or the MAP classifier, that minimizes (3.21) is of the form:

\[
\hat{\omega}_{\text{MAP}}(z) = \arg \max_{\omega \in \Omega} P(\hat{\omega}_i | z)
\] (3.22)

Once prior knowledge is very difficult to estimate (or even impossible), in some circumstances MAP methods are impracticable. Therefore, a method that does not depend on prior probability is preferable. This holds true for methods based solely in the likelihood \( P(z | \hat{\omega}_i) \).


**ERM principle**

The statistical learning theory [Vapnik, 1998], [Burges, 1998] offers a general approach that, although being based on the general and statistical principle of minimizing the expected loss, can be conducted in the general Empirical Risk Minimization (ERM) principle [Vapnik, 1998]. Considering $\mathcal{X}$ and $\mathcal{Y}$ as the input and output spaces, respectively, and restricting to the case of binary classification ($\mathcal{Y} = \{-1, 1\}$), it is assumed that the pairs $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ are random variables distributed according to an unknown distribution $P(\mathbf{z})$. The goal is to construct a classifier $\mathcal{F}(\alpha) : \mathcal{X} \to \mathcal{Y}$ which predicts $\mathbf{y}$ from $\mathbf{x}$, given an observed (labeled) sequence of $n$ i.i.d. pairs $\mathbf{z} = (\mathbf{x}_i, \mathbf{y}_i)$ sampled according to $P(\mathbf{z})$.

The goal of a learning problem, in general terms, is to minimize the risk functional $R(\alpha) = \int L(\mathbf{z}, \alpha) dP(\mathbf{z})$, where $\alpha$ characterizes the parameters of a given learning machine $\mathcal{F}(\alpha)$. Since $P(\mathbf{z})$ is unknown, the expected risk functional $R(\alpha)$ is replaced by the empirical risk

$$R_{\text{emp}}(\alpha) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\mathcal{F}(\mathbf{x}_i) \neq \mathbf{y}_i},$$

(3.23)

given the training set $\mathbf{z} = (\mathbf{x}_i, \mathbf{y}_i)$, where $i = 1, \cdots, n$. Minimizing (3.23) is called the empirical risk minimization induction principle (ERM principle) [Vapnik, 1998].

In non deterministic problems, which is the case of most realistic problems, $R_{\text{emp}}(\alpha)$ is used to measure the agreement of a classifier with the data. $\text{Acc}$, which is a measure proportional to $R_{\text{emp}}(\alpha)$, will be used, jointly with other performance metrics, as criterion to select the best classifier.

**Structural risk minimization (SRM)**

The key problem in pattern classification is to solve the trade-off between fit and complexity, in other words, to keep the model as simple as possible (simplicity and complexity refers to the same problem) and, at the same time, obtain a good generalization capacity (i.e., avoid overfitting). There is no universal definition for complexity, or simplicity, which depends strongly on the problem/domain under study. In the lack of an unique definition of complexity, the following statement can be adopted to summarize the problem: Generalization = Data + Knowledge [Duda, et al., 2001]. This preamle is the intuitive basis for the problem formulation stated by [Vapnik,
1998]: \( R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(\cdot) \); or, in other words: \textit{Expected risk} \leq \textit{Empirical risk} + \textit{Deviation (Complexity penalty)}.

In resume, the key purpose of the SRM principle, in a very direct-term, is to minimize the expected risk \( R(\alpha) \) [Bousquet et al., 2004]:

\[
R(\alpha) \leq R_{\text{emp}}(\alpha) + \epsilon(n, VCd, \eta)
\]

which is bounded by \( R_{\text{emp}}(\alpha) \) (3.23) and by the complexity penalty

\[
\epsilon(n, VCd, \eta) = \sqrt{\frac{VCd_i(\ln(2n/VCd_i) + 1) + \ln(1/4\eta)}{n}},
\]

where \( VCd_i \) is the Vapnik-Chernovenkis dimension, \( n \) is the cardinality of the training dataset, and \( \eta \) is a probabilistic factor to assure probability convergence conditions [Bousquet et al., 2004].

### 3.6 The LIPD dataset

The Laser and Image Pedestrian Detection (LIPD) dataset contains, besides monocular images and LIDAR scans, data from two proprioceptive sensors, an IMU and an incremental encoder, in conjunction with data from the vehicle batteries and data from a DGPS. The dataset was recorded from the sensor acquisition system mounted in the ISR-UC instrument Yamaha vehicle driving through areas of the Engineering Campus of the University of Coimbra and in an urban area from neighborhoods (see Fig. 3.1). Table 3.4 outlines the sensors and its manufacturers, the data communication interface protocols, and the acquisition rate used to record the dataset in a host PC powered by a DC/AC generator connected to a 12Vdc battery.

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Manufacturer</th>
<th>Interface</th>
<th>Acquisition rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIDAR (Alasca-XT)</td>
<td>Ibeo</td>
<td>Ethernet</td>
<td>12.5Hz</td>
</tr>
<tr>
<td>Camera (Guppy)</td>
<td>Allied</td>
<td>FireWire</td>
<td>12.5fps (set@30fps)</td>
</tr>
<tr>
<td>IMU</td>
<td>XSens</td>
<td>USB</td>
<td>12.5Hz (set@120Hz)</td>
</tr>
<tr>
<td>DGPS</td>
<td>TopCon</td>
<td>USB</td>
<td>5Hz</td>
</tr>
<tr>
<td>Encoder and batteries</td>
<td>-</td>
<td>USB</td>
<td>10Hz</td>
</tr>
</tbody>
</table>
Due to the fact that the dataset was obtained in outdoor conditions, and once the sensor apparatus has been exposed to weather and environmental conditions, not unexpectedly, some ‘difficulties’ have occurred namely: light exposure variations, vibrations, oscillations, noise, dust and particles on the air, among others. Perhaps one of the main problems during the data recording was the occurrence of some spots in the images due to dust on the lens.

Once the specific interest lies in comparing detection and classification performances of the proposed methods and algorithms on pedestrian detection using LIDAR and image data-fusion schemes, the criteria adopted to compose the LIPD dataset were:

- The sensor data were recorded as binary files in a single storage unit and sharing the same, and unique, time-stamp of the form $hh\_mm\_ss\_ms$, that is, $hour\_minute\_second\_miliseconds$. The camera was calibrated before and after the dataset recording to avoid inconsistencies.

- The entire LIPD dataset, designated $\mathcal{D}_{LIPD}$, comprises two sets, $\mathcal{D}_C$ and $\mathcal{D}_D$, hence $\mathcal{D}_C \cup \mathcal{D}_D = \mathcal{D}_{LIPD}$. The former is used for pedestrian classification purposes, and the later for pedestrian detection and context-based analysis. Multi-sensor data is contained in $\mathcal{D}_D$, while $\mathcal{D}_C$ is composed of cropped images.
and laser segments.

In summary, $\mathcal{D}_C$ is composed exclusively of laser-segments (defined in Section 4.2) and ROIs in the image frames, representing positives and negatives, aiming to evaluate exclusively the classification algorithms. On the other hand, $\mathcal{D}_D$ comprises raw laser-scans and full images frames, necessary to evaluate the detection system under realistic application-specific requirements. Both $\mathcal{D}_C$ and $\mathcal{D}_D$ are divided in two parts\(^3\): a training set $\mathcal{D}_\text{Dtrain}$, used to train the classifier parameters and also to perform cross-validation, bagging and feature selection; and the testing set $\mathcal{D}_\text{Dtest}$ used to evaluate the performance of the techniques and methods learned using $\mathcal{D}_\text{Dtrain}$. Both $\mathcal{D}_\text{Dtrain}$ and $\mathcal{D}_\text{Dtest}$ were partitioned in two sets: laser-based ($\mathcal{D}_\text{laser}$) and image-based ($\mathcal{D}_\text{ima}$) sets. The cardinality is defined by the number of samples, and dimensionality by the number of features.

For sake of clearness, the following notation should be observed in the sequel (for both $\mathcal{D}_C$ and $\mathcal{D}_D$): $\mathcal{D}_\text{train} = \mathcal{D}_\text{laser}\text{train} \cup \mathcal{D}_\text{ima}\text{train}$, $\mathcal{D}_\text{test} = \mathcal{D}_\text{laser}\text{test} \cup \mathcal{D}_\text{ima}\text{test}$, $\mathcal{D}_\text{laser} = \mathcal{D}_\text{laser}\text{train} \cup \mathcal{D}_\text{laser}\text{test}$ and $\mathcal{D}_\text{ima} = \mathcal{D}_\text{ima}\text{train} \cup \mathcal{D}_\text{ima}\text{test}$.

\textbf{Definition 3.6.1.} Positive example: is defined by a entire body pedestrian (marked by black bounding-boxes in Fig. 3.2) present in both the camera and laser field of view (FOV). Any other entity present in the FOV of both sensors is considered a negative example (nPED), while an occluded pedestrian denotes a partial occluded PED.

\textbf{Definition 3.6.2.} ROI: is a region defined by the projection of a given laser-based segment in the image plane. ROIs are defined considering the extremes of a segment rather than individual laser-points - as illustrated in Fig. 3.2. Due to the Alasca XT laser sensing principle, the vertical component in the underlying data is strongly limited, therefore the top and bottom part of an object can not be estimated directly from the laser measurements. Under the assumption of flat surface, knowing the distance of the laser setup from the ground and considering $2.5m$ the maximum height of a pedestrian, the ROIs coordinates are calculated.

\textbf{Remark} Dataset notation: subscripts and superscripts will be used throughout this text to differentiate specific characteristics of a given set and they will be omitted

\(^3\)In the sequel the subscripts refer to the detection set $\mathcal{D}_D$. The classification dataset follows similar notations, just substituting the subscripts by $C$.\(^3\)
3.6. THE LIPD DATASET

Figure 3.2: Laser-based ROI projections in the image frame. The laser-points are illustrated by dots in the laser space and their projections in the image are represented by crosses, where the colors represent a LIDAR layer. The segments, enclosed by black-ellipses, and their corresponding pedestrians are marked as bounding-boxes in the image.

when its identification is evident in the context they are used, unless otherwise stated, and when there is no ambiguity.

The manual labeling process, inherent to any supervised dataset, was carried out using the image frames as primary reference for pedestrian and non-pedestrians annotation. The labeled segments, extracted from raw data laser-scans, were validated using the corresponding image frame (for ground truth confirmation). All the samples of interest were labeled under user supervision, avoiding some problems invariably presented on realistic situations, such as: data association mistakes, over-segmentation, measurement missings, calibration imprecision, road irregularities, tracking inconsistencies, vehicle vibrations, and so on. However, it is not possible to guarantee 100% of correspondence, due to a sort of reasons, such as: it is a human-based task and it is prone to mistakes; the calibration between the sensors is not perfectly precise; the time synchronization is not perfect and neither with a precise constant interval.

3.6.1 Classification dataset

The elements of the classification dataset $\mathcal{D}_C$ are feature vectors, i.e., the set is defined in terms of vector elements. The laser-based set is defined as $\mathcal{D}_{\text{laser}} = \{ \mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n \}$:
CHAPTER 3. BACKGROUND AND DATASET

Figure 3.3: Illustrative images of: cropped pedestrians (upper part of the figure); full frames (middle part); and ROI projections (blue rectangular boxes, illustrated in the lower part of the figure).

\[ n = (n_{\text{train}} + n_{\text{test}}) \], where \( n \) is the total number of samples and \( f_k \) is the \( k \)-th laser-based feature vector. Similar definition is valid for \( D^{ima} \) (substituting \( f \) by the image-based vectors \( f' \)). The cardinality depends on the number of samples e.g., \( |D_{\text{train}}| = n_{\text{train}} \), and the dimensionality varies with the number of features: \( |f| \), or \( |f'| \), or the union of both. Figure 3.4 shows, for the training set, the bounding-boxes size (width and height) distribution in pixel.

The training and testing sets, \( D_{\text{train}} \) and \( D_{\text{test}} \) respectively, were captured in two sessions in the Winter season of 2008, in the ISR-UC Campus \(^4\), under the following sensors configuration:

1. Laser-based data: the Ibeo LIDAR was mounted approximately 52 cm above the ground, with FOV restricted to 120°, horizontal angular resolution of 0.25°, vertical resolution of \([-1.6^\circ, -0.8^\circ, 0.8^\circ, 1.6^\circ]\), and measurement range up to 35 m;

2. Image-based data: the Allied Guppy camera, with a 6.5mm lens, was mounted above the laser, with horizontal angle of view of \( \approx 42^\circ \), as shown in Fig. 1.3.

\(^4\)http://www.isr.uc.pt/~cpremebida/PoloII-Google-map.pdf
3.6. THE LIPD DATASET

Figure 3.4: Training dataset distribution regarding the bounding-boxes height and width (in pixels).

Positives correspond to pedestrians in static or moving states, and the negatives consist of posts, tree-trunks, hydrants, light-posts, walls, fences, brushes, foliage, cars, etc. The dataset and the corresponding ground truth, generated under user supervision, are available on the Web\(^5\). Although they were collected around the same area, $D_{\text{train}}$ and $D_{\text{test}}$ differ on the dates and the day-time period they have been recorded. Another relevant aspect is that on the test set, some samples were acquired at dusk, where the illumination condition changed drastically. Some images of the dataset are shown in Figure 3.3. Table 3.5 summarizes the dataset characteristics.

It is important to mention that ROI elements, used to extract the image-features $f^v$ which compose $D_{\text{ima}}$, were extracted directly from the laser-segment projections in the image plane without user intervention or any post-processing; it means that all the ROIs were obtained directly from the labeled laser-segment projections and, as consequence, are prone to error due to calibration imprecision, road irregularities, vehicle vibrations, and so on. Nevertheless, it was decided to include the ROIs with no user intervention or any correction, resulting in a closer realistic image-based dataset.

\(^5\)http://www.isr.uc.pt/~cpremebida/dataset
Table 3.5: Statistics of the classification set $\mathcal{D}_C$

<table>
<thead>
<tr>
<th>Designation</th>
<th>Total</th>
<th>$np$</th>
<th>$nn$</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{D}<em>{C\text{train}}$ $n</em>{\text{train}}=1100$</td>
<td>550</td>
<td>550</td>
<td></td>
<td>Sunny day, Winter season, collected between 15:30 to 16:30</td>
</tr>
<tr>
<td>$\mathcal{D}<em>{C\text{test}}$ $n</em>{\text{test}}=1400$</td>
<td>400</td>
<td>1000</td>
<td></td>
<td>Sunny day, Winter season, collected between 12:00 to 17:30</td>
</tr>
</tbody>
</table>

Figure 3.5: Bounding-boxes geometrical center, defined in pixel coordinates, for the training dataset. Pedestrian locations are concentrated around the region with average pixel equal to 230.

### 3.6.2 Detection dataset

Approximately 80 Gbytes of image frames and raw laser-scans, collected in different days of the Autumn of 2010, at day-light conditions, and in sessions of 4-5 hours, were recorded for the dataset composition. From that amount of data, a selection process was carried out to reduce the cardinality of the dataset at tractable value and, at the same time, to keep a representative dataset in terms of the universe that defines the problem of pedestrian detection in outdoor scenarios. Finally, 14367 images were used to compose the training set and 12050 to compose the testing set of the $\mathcal{D}_D$ set.
3.6. THE LIPD DATASET

The training part of the dataset contains 5237 manually labeled positives (image’s cutouts of pedestrian in up-right entire body), and 6328 full-frame images without any pedestrian evidence. Thus, the elements of the training dataset are the aforementioned 5327 positive bounding-boxes and a free-number of negative bounding-boxes which can be extracted from the negatives frames. The geometrical center of the positive bounding-boxes, in pixels, are shown in Fig. 3.5. The testing set contains 4823 full-frame images, which correspond to the frames with DGPS information extracted from the entire set (with 12050 samples). To be consistent with recent benchmarking datasets, detailed annotations regarding the pedestrians appearance (in terms of occlusion) were done, namely: occluded/partial pedestrians (class-0) and entire body pedestrians (class-1). A summary of the dataset characteristics is given in Table 3.6. The position and size, in pixels, of the labeled bounding-boxes are shown in Fig. 3.6.

In summary, Fig. 3.7 shows a representation of the sets and ‘subsets’, and their elements, that compose the LIPD dataset. It is important to note that $\mathcal{D}_C$ is composed of a precise number of feature-vector elements, i.e., 1100 elements in $\mathcal{D}_{\text{train}}$ and 1400
elements in $\mathcal{D}_{test}$. On the other hand, the composition of the $\mathcal{D}_{D}$ set is subtle in the sense that the corresponding number of negative elements depends on the approaches used for sample selection, filtering, detection methodology, and so on.

![Figure 3.7: Representation of the LIPD dataset, in the form of a diagram, showing the classification and detection sets, $\mathcal{D}_{C}$ and $\mathcal{D}_{D}$, and their subsets. Regarding $\mathcal{D}_{D}$, notice the following: (1) the exact number of negative examples on $\mathcal{D}_{ima}$ depends on the detection and classification approaches to be used; (2) raw laser data is provided; (3) all the positive examples correspond to labeled cropped images.](image-url)
Chapter 4

LIDAR-based system

This chapter deals with the LIDAR data processing system, which are composed of the following stages: (1) preprocessing; (2) segmentation; (3) feature extraction; and (4) classification. Figure 4.1 shows, in the form of a block-diagram, the inputs and outputs through the processing stages. Furthermore, the last section of this chapter is devoted to the stochastic problem for pedestrian speed estimation using measurements from the Alasca-XT lasercanner.

4.1 Preprocessing

The preprocessing stage is in charge of coordinate transformation, filtering, a pre-segmentation performed in each laser-layer, and camera-based FOV determination. Basically, this processing stage outputs the laser points of interest for further segmentation. The raw-data acquired by the laser is defined in a $\mathbb{R}^3$ measurement space. In polar coordinates, a given range-point is denoted by $p_i = (r_i, \alpha_i, \phi_i)$, as
CHAPTER 4. LIDAR-BASED SYSTEM

Figure 4.2: Laser coordinates convention, with the Cartesian and polar representation of a given point $p_i$. At the right, the laser-layers and the vertical angular resolution are shown.

shown in Fig. 4.2, where $\phi_i \in \{-1.6^\circ, -0.8^\circ, 0.8^\circ, 1.6^\circ\}$ defines the ‘vertical’ layers. Expressing a scan as a sequence of $n_r$ measurement range-points in the form $Scan = \{p_1, p_2, \cdots p_{n_r}\}$, a given point $p_i \in Scan$ is fully represented as:

$$p_i = (x_i, y_i, z_i, r_i | x_i = r_i \cos(\alpha_i), y_i = r_i \sin(\alpha_i), z_i = r_i \sin(\phi_i)$$ \hspace{1cm} (4.1)

where $r_i = (x_i^2 + y_i^2)^{1/2}$, as illustrated in Fig. 4.2.

Due to the Alasca XT laser measurement principle, $n_r$ is not always constant, showing small differences among the layers. After analyzing thousands of laser scans, collected with the same acquisition frequency of 12.5Hz and under constant LIDAR configuration parameters, the number of points per layer, denoted $n_l$, has been delimited to $n_l = 1300 \forall l \in \{1, 2, 3, 4\}$. The range points provided by the LIDAR come in polar coordinates. Throughout this thesis, by default, the range points are defined in Cartesian coordinates, therefore ‘coordinate transformation’, in Fig. 4.3(a), refers to the conversion process to represent the set of points in Cartesian coordinates according to the convention illustrated in left part of Fig. 4.2.

To complete the preprocessing stage, other processes are performed, which are: (1) pre-segmentation and filtering: the process of discarding spurious/’isolated’ points (outliers) and points above a maximum distance of interest $r_{max} = 35m$. Also, in this processing step the range points from $Scan$ are grouped per layer; (2) FOV determination: the process of removing points that lie out the camera FOV; thus, this stage is used when the LIDAR is combined with image frames from the camera. Figure 4.3 illustrates the ‘internal’ processing steps towards pedestrian classification. The
4.2. SEGMENTATION

Figure 4.3: Processing steps in the LIDAR-based preprocessing stage. (a) Range-points in Cartesian coordinates. (b) Points are filtered and grouped per layer, where each color indicates a given layer. (c) Range-points outside the camera FOV are discarded. (d) Projection of the points, per layer, in the image frame. Laser-to-image projections, and the FOV determination, depend on the calibration, described in 3.2.

Evolution of the laser data through the preprocessing stages is also shown. Moreover, projection of laser-points in the image is given in Fig. 4.3(d). The color of the laser-points follows the standard shown in the right part of Fig. 4.2.

**Definition 4.1.1.** Let \( l \) be the positive integer that represents the laser-layer subscript thus, for the Alasca XT \( l \in (1, 2, 3, 4) \). A given layer is defined as the nonempty set \( \{ \text{Scan}_l \} \) that consists of points measured by the \( l^{th} \) layer. Thus: \( \{ \text{Scan}_l \} = \{ p_1, p_2, \cdots, p_{n_l} \} | i \leq n_l; \) where \( n_l \) is the number of points in \( \text{Scan}_l \).

In conclusion, the preprocessing module outputs the sets \( \text{Scan}_l \), where the range-points has a maximum range limited by \( r_{\text{max}}: \max_i(r_i) \leq r_{\text{max}}, \forall r_i \in \text{Scan}_l \). Notice that, after pre-segmentation and filtering, the number of points per layer \( n_p_l \) will be \( < 1300 \), and will be \( \ll 1300 \) when a FOV is imposed. Hereafter the subscript of the elements of \( p_i \) will be dropped, and also the braces \{\} on the sets. In case of ambiguity it will re-used.

4.2 Segmentation

Segmentation constitutes a very important stage that can be performed by means of specific methods as presented in [Premebida and Nunes, 2005]. The segmentation purpose is to extract groups of points, from \( \text{Scan}_l \), sharing similar spatial properties.
It is a decisive processing phase since failures during this stage will strongly affect all the subsequent modules, with a not easy retrofit correction solution. A tradeoff exists in the segmentation process, which consists on deciding between merging or splitting the spatial distributed laser-points. In describing the segmentation process, the definition for segment is:

**Definition 4.2.1.** A segment is composed by a set of laser-points sharing similar spatial properties. A given segment is defined by the set $S_i : i \leq ns$ of laser-points that respects a given clustering condition. Therefore, $S_i$ is the set of points conditioned on $\alpha$: the parameters of a given (chosen) segmentation method.

An example of segments $S_{1,..,4}$ extracted from a group of points in Scan$_i$ is shown in Fig. 4.4. In practical (real) solutions, the number of segments $ns$ per scan is limited, thus the subscript $i$ in Definition 4.2.1 is up to a few dozens (in this work it is limited to $ns = 20$). The set of segments $\{S\} : S_i \subseteq S$, obtained by the segmentation stage, is treated as inputs for the feature extraction stage and, further, as objects of interest in the classification stage. When time is under consideration, which is particularly important in the tracking stage, a segment will be explicitly represented by $S_i(k)$, otherwise the time-index ($k$) will be omitted.

**Remark** Due to classification purpose, mentioned in the sequel, a segment $S_i$ is explicitly defined by a group of range-points related to one, unambiguously, object of interest. Although in realistic situations it does not occur in a deterministic way.
4.2. SEGMENTATION

4.2.1 Single vs Multilayer data segmentation

Multilayer data segmentation follows, in general terms, a similar processing strategy used for single layer lasers (e.g., Sick LMS200). However, multi-layer lasers bring additional difficulties to the process, namely: (1) the over-segmentation occurrence is much more frequent; (2) the laser points of the same object present a considerable variance specially when the "shape" of such object varies in function of the height; (3) the horizontal angular resolution is not constant; (4) segment to segment association (in this case, multi-segment merging) is a mandatory processing step. The solution for these problems is not unique, and few research has been published with satisfactory and detailed solution so far: making the problem even worse.

A solution to deal with multi-layer data segmentation has been proposed, which comprises a set of processing steps, culminating in the (final) set of segments. The first step is performing segmentation per layer, then an iterative process is used to merge the segments detected in all the layers into a final set \( S \). This process can be seen as a function \( S = f_\alpha(S_{1,l}, \ldots, S_{ns_l,l}) \) which takes as input the \( ns_l \) segments extracted from the layers, and outputs the final set of segments \( S \) based on the parameters \( \alpha \).

Thus, given \( ns_l \) segments extracted from \( \text{Scan}_i \) in the all layers \( i.e., \) for \( l = (1, 2, 3, 4) \), the merging process consists of the following steps:

1) fit a rectangle, defined in Cartesian coordinates, in each segment \( S_{j,l} \);
2) calculate the area of intersection \( A_{j,l} \) among all the rectangles;
3) finally, merge the segments with \( A_{j,l} > 0.25^1 \).

In resume, the final set of segments \( S \) is the output of the function \( f_\alpha : S_{j,l} \rightarrow S_i \), where \( j = 1, \ldots, ns_l \) and \( i = 1, \ldots, ns \). Here, \( \alpha \) is characterized by the intersection area. Notice that other attributes, such as spatial similarities or features, can be used to reinforce the decision merging process. In [Kwak et al., 2010] a supervised SVM-based approach is presented where range-based attributes and image-based features are combined for object segmentation. This approach requires, as expected, that both laser and camera are calibrated.

In general terms, segmentation is defined as the process of separating foreground objects from the background in the sensor measurement space. The key step is to detect the break-point, characterized by a discontinuity between two consecutive laser

\(^1\text{This value was chosen based on experimental trials.}\)
Algorithm 1 Distance-based segmentation methods

**Input:** Scan: set of range points, Thr: specific threshold for a chosen method

**Output:** S: set of segments;

1: np: number of points in Scan;
2: i ← 0: number of segments;
3: for n = 1; n < (np − 1); n + 1 do
4: calculate the Euclidean distance $D_{eucl}$ between $p_n, p_{n+1}$;
5: if $D_{eucl} > Thr_i$ then
6: A break-point is detected;
7: i ← i + 1;
8: $S ← S_i$ form the segment;
9: else
10: continue;
11: end if
12: end for

points, which represents, possibly, an object boundary. The next subsections present several methods, used for 2D laser data segmentation, grouped in three categories, 1) distance-based methods, 2) stochastic distance-based methods, and 3) multivariable.

A number of different segmentation methods have been implemented and compared. Most of the individual algorithms are detailed in [Premebida and Nunes, 2005], and consequently will be not detailed here, except the proposed multivariable method (Section 4.2.2). The general concepts behind the methods, and a proposed taxonomy, are shortly described in the sequel.

**Distance-based segmentation methods**

This is the most used approach for detecting break-points in 2D data due the simplicity and considerable efficiency of the methods pertaining to this category. Differences in range and bearing, or $x$ and $y$ in the Cartesian case, of adjacent laser measurements are used to declare the existence of a discontinuity based on a threshold value (or discontinuity measurement). The methods included in this category, summarized in Algorithm 1, share a common aspect: the Euclidean distance between two laser-points $D_{eucl}(p_i, p_{i+1})$ are calculated and, if $D_{eucl}$ is greater than a threshold $Thr_i$ ($i$ relates to a given method), a breakpoint is considered to be occurred. The clustering methods differ in the expression used to compute $Thr_i$. 
Algorithm 2 Stochastic distance-based segmentation methods

Input: $\text{Scan}_l = \{p_1, \cdots, p_{\text{np}_l}\}$: set of range points, $\mathfrak{M}$: stochastic model of the filter (e.g., KF)

Output: $S$: set of segments per layer $l$

1. $\text{np}_l$: number of points in $\text{Scan}_l$;
2. $i \leftarrow 0$: number of segments;
3. Initialization of the filter model;
4. for $n = 1; n < (\text{np}_l - 1); n + 1$ do
5. State and covariance prediction;
6. Measurement prediction: $\hat{p}_n$;
7. Generate the validation region ($VR$) conditioned on a $\chi$-test threshold$^2$;
8. Compute the innovation $\nu = \hat{p}_n - p_n$;
9. if $\nu \in VR$ then
10. Filter gain calculation;
11. Update the state vector and the covariance matrix;
12. else
13. A break-point is detected;
14. $i \leftarrow i + 1$;
15. $S \leftarrow S_i$ form the segment;
16. Initialize the model $\mathfrak{M}$;
17. end if
18. end for

Stochastic distance-based segmentation methods

This type of segmentation method uses a stochastic filter, namely KF or EKF, in such way that a validation region, built around the estimated measurement, is considered to decide if a new laser-point should be validated (continues in the process) or discarded (a breakpoint is detected). All the methods of this category were studied and implemented under Gaussian assumptions on the validation regions, i.e., following a Chi-square distribution. Algorithm 2 summarizes the functional steps involved in these methods.

Multivariable segmentation method

In this method a set of attributes, calculated from two consecutive laser-points, for instance $P_i = [p_i, p_{i+1}]$, are used to form a multivariable vector which is compared with the next vector, calculated from the pair ahead $P_{i+1} = [p_{i+1}, p_{i+2}]$. Hence, a breakpoint is detect if the cosine distance between $P_i$ and $P_{i+1}$ is greater than a threshold. This method is detailed in the sequel.
4.2.2 Multivariable segmentation

The method proposed in this section is formulated as a decision making process where a set of attributes, a multivariable array of features, extracted from pairs of laser points, are used to support in deciding if a subsequent point should be merged (considered as part of the cluster) or split (a break-point). This decision is based on a similarity multidimensional metric e.g., Cosine similarity, Mahalanobis distance, Euclidean distance. Thus, the evidence of a breakpoint existing between two laser points is based on a threshold defined as function of the metric to be used; the threshold values were estimated using a set of segments, from a validation dataset, corresponding to labeled objects: pedestrians, vehicles, and others. Based on the validation dataset, build for this purpose, the Cosine distance has been chosen to be used as the decision metric. For each consecutive pair of laser points $P_i = (p_i, p_{i+1})$, the following set of attributes are calculated:

$$f_1 = \sqrt{\Delta P_X^2 + \Delta P_Y^2}$$ where $\Delta P_X$ and $\Delta P_Y$ are the difference between $p_i$ and $p_{i+1}$ in $x$ and $y$ Cartesian directions.

$$f_2 = \bar{p}_i$$ it is the average value of $p_i$ and $p_{i+1}$, that is, $f_2 = (p_i + p_{i+1})/2$;

$$f_3 = \bar{p}_i \cdot \Delta P_X$$: is the scalar multiplication between $\bar{p}_i$ and $\Delta P_X$;

$$f_4 = \bar{p}_i \cdot \Delta P_Y$$: is the scalar multiplication between $\bar{p}_i$ and $\Delta P_Y$;

$$f_5 = \text{standard deviation of } \bar{p}_i$$;

$$f_6 = \text{second order moment of } \bar{p}_i$$;

This segmentation method is a sequential process (in terms of spatial ordering), performed for two pairs of points, for instance $P_i$ and $P_{i+1}$. The vector of attributes $s_i = (f_1, \ldots , f_6)$, which corresponds to the pair $(p_i,p_{i+1})$, is compared with $s_{i+1}$, which is calculated using $(p_{i+1},p_{i+2})$, and the process follows two steps: 1) the Cosine distance $CosD_i$ (4.2) between $s_i$ and $s_{i+1}$ is calculated and, 2) if $CosD_i > Thr$, then $p_{i+2}$ is considered a break-point. Otherwise $p_{i+2}$ is considered part of the segment which contains $p_i$ and $p_{i+1}$.

$$CosD_i = \frac{s_i \cdot s_{i+1}}{||s_i|| ||s_{i+1}||} = \frac{\sum_{j=1}^{6} s_i(j) \times s_{i+1}(j)}{\sqrt{\sum_{j=1}^{6} (s_i(j))^2} \times \sqrt{\sum_{j=1}^{6} (s_{i+1}(j))^2}}$$ (4.2)
Comments: although the segmentation methods described above can work well in most of the cases, they face difficulties in more challenging situations in urban scenarios that occur frequently enough to be not considered. A number of situations that are particularly difficult to be assessed for segmentation methods are:

- **partial occlusions**: It is a problematic occurrence which actually does not affect the segmentation process itself but, the process of segment-to-segment association is indeed influenced since objects partially occluded tend to create more segments than the number of objects;

- **cluttered situations**: That is the drama of any segmentation method, specially when the LIDAR scans zones with vegetations, or group of people (close each other), to cite two examples. As expected, it turns much more complex when the vehicle and/or the objects are moving;

- **translucent/transparent surfaces**: Due to the sensing operation principle of the LIDAR, which is subjected to the infrared laser beams frequency and intensity, and the detection electronics of the sensor, some objects have weak reflectivity and the range measurements are sparse, incomplete and with large deviation. Distance turns the problem worst. Parts of the vehicles, such as glass-built materials, are typical examples of such occurrence;

Another important aspect that brings considerable difficulty to the segmentation process is the LIDAR motion (actually the vehicle movement). Although the vehicle, and the LIDAR, dynamics can be estimated with some rigor, the objects detected by the LIDAR can also be in movement and its shape (the objects contour perceived by the laser) may change abruptly causing an additional difficulty for the segmentation process. In conclusion, the goal of the segmentation methods described herein is to determine segments in a laser scan where, ideally, each segment should be associated with exactly one physical object of interest. From each detected segment a feature vector is calculated and further used, in a classifier, to predict the class (pedestrian or not) which corresponds the segment.
4.3 Feature extraction

Features extracted from LIDAR data and its utilization for pedestrian or people detection is a subject that was investigated by [Streller and Dietmayer, 2004], [Douillard et al., 2007] and [Arras et al., 2007]; although the latter one was concentrated on indoor environments, many of the features used here are based on Arras’s work. The feature vector is calculated using 2D information, in polar and/or Cartesian space, hence for the case of a multi-layer LIDAR the vertical information has to be projected on a common 2D plane, which means that all these features can be used in single-layer lasers. Figure 4.5 illustrates an example of a segment and the related feature set.

Denoting \( x \) as the set of points, belonging to a given segment \( S_i \), in \( x-y \) coordinates, that is, \( x = \{ x_i, y_i \} \) (see equation (4.1)). In the sequel, a 18-dimensional laser-based feature vector \( f_i = \{ f_1, \cdots, f_{18} \} \) is detailed, with many of the features been contribution of the author.

\( f_1 = np \cdot r_{\text{min}} \): is the product of the number of range-points (\( np \)) with the minimum range distance (\( r_{\text{min}} \));

\( f_2 = np \): the number of points in \( S_i \);

\( f_3 = \sqrt{\Delta X^2 + \Delta Y^2} \) (Normalized Cartesian dimension): this feature corresponds to

![Figure 4.5: An example that illustrates a segment of range points, which corresponds to a pedestrian in the laser plane, with the corresponding feature set (in the text-box); some attributes and geometric primitives are also depicted.](image)
the root mean square of the segment width ($\Delta X$) and length ($\Delta Y$) dimensions;

\[ f_4 = \sqrt{\frac{1}{np-1} \sum_{n=1}^{np} \| x_n - \bar{x} \|} \] (Internal standard deviation): denotes the standard deviation of the range-points ($x_n$) with respect to the segment centroid $\bar{x}$;

\[ f_5 = \text{Radius} \leftarrow (\text{fitted circle's radius}) \] denotes the radius of a circle extracted from the segment points. The Guivant’s method [Guivant et al., 2002] has been used in fitting the circle and to extract the corresponding radius;

\[ f_6 = \frac{1}{np} \sum_{n=1}^{np} \| x_n - \hat{x} \| \] Mean average deviation from the median $\hat{x}$;

\[ f_7 = IAV \] (Inscribed Angle Variance): proposed by [Xavier et al., 2005], corresponds to the mean of the internal angles along the extremes points and the in-between points that constitute the segment;

\[ f_8 = \text{std}(f_7) \] Standard deviation of the inscribed angles calculated previously;

\[ f_9 = \frac{1}{np} \sum_{n=1}^{np} (x_n - \hat{x}_{l,n})^2 \] (Linearity): this feature measures the straightness of the segment and corresponds to the residual sum of squares to a line $\hat{x}_{l,n}$ fitted into the segment in the least squares sense;

\[ f_{10} = \frac{1}{np} \sum_{n=1}^{np} (x_n - \hat{x}_{c,n})^2 \] (Circularity): this feature measures the circularity of a segment. Like for the $f_9$ feature, it denotes the summation of the squared residuals to a fitted circle $\hat{x}_{c,n}$;

\[ f_{11} = \sum_{n=1}^{np} \frac{(x_n - \mu_x)^{ko}}{np} \] ($2^{th}$ central moment): it is the $2^{th}$ moment taken about the mean $\mu_x$, and $ko$ is order of the moment, i.e. $ko = 2$;

\[ f_{12} \] ($3^{th}$ central moment): $f_{11}$ with $ko = 3$;

\[ f_{13} \] ($4^{th}$ central moment): $f_{11}$ with $ko = 4$;

\[ f_{14} = \frac{1}{np} \sum_{n=1}^{np} || x_n - x_{n-1} || \] (Segment length): this feature is defined as the summation over the norm of the Euclidean distance between adjacent points;

\[ f_{15} = \text{std}(f_{14}) \] Standard deviation of the segment length;

\[ f_{16} = \sqrt{\frac{\sigma_x^2 + \sigma_y^2}{2}} \] (Standard Deviational Ellipse): $\sigma_x^2$ and $\sigma_y^2$ are the variances in $x$ and $y$ Cartesian directions respectively. While the standard distance deviation is a good single measure of the dispersion around the mean center, it does not show the potential skewed nature of the points (anisotropy). The standard deviation ellipse gives dispersion in the two dimensions ($x$ and $y$);

\[ f_{17} = \sum_{n=1}^{np} \frac{(\text{Deuc}_n)^2}{np-2} \] (Unbiased distance deviation): $\text{Deuc}_n$ is the Euclidean distance between each point $x_n$ and the mean center $\mu_x$. The subtraction by 2 in the denominator provides an unbiased estimate of standard distances, since $\text{Deuc}_n$ has two constants;

\[ f_{18} = \frac{\sum_{n=1}^{np} \text{Deuc}_n - (\sum_{n=1}^{np} \text{Deuc}_n)^2/np}{np-1} \] (Euclidean distance dispersion): it is a measure of dispersion between the summation of the squared points and the square of the
Fig. 4.6: Redundancy computed using the feature set $f_i, i = 1, \cdots, 18$, for the lasers Ibeo Alasca-XT (left) and Sick LMS200 (right). The features with indexes $i = 2, 6, 12, 13, 14, 17$ are very redundant, particularly for the Sick.

4.3.1 Feature analysis and comparison

An important aspect when designing a classifier is to decide the feature set to be used and, if possible, the composition of features to be applied during the classification process. To prevent redundancy and to take advantage of the diversity among the features, the method named mRMR (minimum-redundancy maximum-relevancy) [Peng et al., 2005] have been adopted as feature selection. Figure 4.6 illustrates the redundancy among the features as function of the pixel intensity in gray-scale map. The greater the redundancy between a pair of features, the pixel color tends to white. The features with maximum relevance using the Ibeo laser dataset are (refer to Fig. 4.7): $f_i=[6, 17, 14, 3, 4, 5, 1, 2, 15, 11, 9, 10, 16, 12, 13, 18, 8, 7]$, and those using the Sick laser: $f_i=[1, 5, 17, 6, 14, 2, 3, 4, 15, 9, 8, 18, 10, 7, 11, 12, 13, 16]$. On the other hand, the set of features with maximum relevance and minimal redundancy are: $f_i=[6, 14, 15, 17, 1, 3, 2, 9, 4, 8, 5, 11, 10, 7, 12, 16, 18, 13]$, and $f_i=[1, 15, 14, 8, 2, 9, 5, 11, 6, 12, 4, 13, 17, 16, 18, 10, 7, 3]$, for the Ibeo and the Sick LIDAR respectively.
4.4 Classification

The LIDAR-based classification stage consists of a decision-making function which decides the class (PER or nPED) a given detected object belongs to. The aforementioned decision function can assumed many forms, for instance: (1) a single classifier; (2) an ensemble of classifiers; or (3) a cascade. General aspects concerning the classification methods, feature selection, sample selection, to mention some topics, will be omitted here. Reciprocally, these issues are addressed in Chapters 5 (Vision module) and 6 (Fusion methods). In particular, the classification strategies used in the LIDAR-based module are:

1. Single classifiers: five classifiers have been used, namely LDA, NBC, GMM, SVM, and ANN.

2. Multi-classifiers fusion: two fusion strategies were employed to combine classifiers: (1) fusion rules (average, minimum, maximum, product, majority vote), and (2) trainable-fusion rules.

3. Cascade: two cascades have been implemented: an Adaboost and a cascade of SVM.

All these classification techniques share some common aspects, namely: the input feature-vector is the same (described in Section 4.3); the decision making process
depends directly of the current-time likelihood (or classifier confidence), however when
tracking is available the classifier decision can make use of prior information; all the
classification methods are trained in a supervised way, using the datasets and the
principles addressed in the Chap. 3.

To assess the best classifier (that is, having the best generalization ability) to be
used in the pedestrian detection system, a set of experiments, on the classification
dataset (detailed in Chap. 3), have been carried out. To support the experimental
analysis, the results were compared in terms of ROC, AUC, and Acc (see Section 3.4
for details). The results presented in this chapter correspond to experiments with
single classifiers, while the remaining strategies (fusion of classifiers and cascade)
are presented in the Chapter 6 - Fusion module. The ROC curves are shown in
Fig. 4.8, and the classification performance metrics (AUC, Acc) and the number
of features ($n\text{fe}a$) are given in Table 4.1, where $n\text{fe}a$ is the number of features
used in the classifiers. Notice that, due to the NBC and GMM principle, these
classification methods demand a reduced number of features; the cause behind this
behavior is once $n\text{fe}a$ increases the chance to obtain redundant features also increases
and consequently the classifier likelihoods tend to zero.

As a demonstration of the impact on the classification performance when $n\text{fe}a$
varies, consider the bar-graphs in Fig. 4.9(a) and Fig. 4.9(b), where the AUC and
Table 4.1: Classification performance: testing set

<table>
<thead>
<tr>
<th></th>
<th>Ibeo dataset</th>
<th></th>
<th>Sick dataset</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AUC</td>
<td>Acc</td>
<td>nfea</td>
<td></td>
</tr>
<tr>
<td>LDA</td>
<td>0.915</td>
<td>0.865</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>NBC</td>
<td>0.949</td>
<td>0.857</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>GMM</td>
<td>0.953</td>
<td>0.880</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.966</td>
<td>0.933</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>0.941</td>
<td>0.941</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.9: Number of features vs classifier performance using the dataset obtained with the Ibeo Alasca-XT and with Sick LMS200.

The accuracy (Acc) were used as classification performance metrics. The total number of features, restricted to 10, have been selected using the mRMR method. The best results for each classification method are presented in Table 4.1.

Summary

The classification performance were evaluated in terms of Acc, AUC and ROC curves. Each single classifier were trained and tested as function of the number of features ($nfea = 1, \cdots, 18$). Based on in Table 4.1, it is possible to conclude, with some approximations, that LDA, SVM and ANN are inclined toward the best results if $nfea$ tends to the maximum value. On the other hand, GMM and NBC have good performances when $nfea$ is, in average, less than the half of the number of features. The ROC curves (shown in Fig. 4.8) reinforce the conclusion that the SVM and ANN are the best classification methods to be used, however both methods demand much more features than NBC and GMM. Finally, notice that the classification performance consists on evaluating the classifiers output obtained on feature vectors extracted
from labeled segment, what is a simplification regarding the detection case where
laser scans enter into the system as raw-data streams, which are subjected to a sort
of situations and difficulties, such as: missing readings, cluttered, inconsistencies on
the data association, segmentation mistakes.

4.5 Speed estimation and modeling for pedestrians

In this chapter state vector estimation and kinematic models of the dynamic behavior
of pedestrians are considered. Pedestrian dynamics is modeled under linear time-
varying stochastic assumptions, although the state observations enter in the system
as non-linear equations. Moreover the system is modeled directly as discrete-time, and
the process is considered to be driven by white noise. The position of a pedestrian,
defined by the position of a segment in the LIDAR Cartesian system, is considered
to evolve in time constrained to three motion models: constant velocity, constant
acceleration, and constant jerk model. These models are tested and compared using
a database with sequences of pedestrians in different poses and dynamic behavior.

The discrete-time process noise $v(k)$ is a scalar-valued zero-mean white sequence
that is, non-correlated:

$$E[v(k)v(k + 1)] = \sigma_v^2 \delta_{k,k+1}$$  \hspace{1cm} (4.3)

and enters into the dynamic equation, proportionally to the noise gain $\Gamma$, as follows:

$$x(k + 1) = Fx(k) + \Gamma v(k)$$  \hspace{1cm} (4.4)

The state vector $x$ is represented by a column vector composed of the object state
variables: position, velocity and acceleration. The measurement equation is

$$z(k) = Hx(k) + \omega(k)$$  \hspace{1cm} (4.5)

where $H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ (the LIDAR-based system provides object position), and $\omega(k)$
is the sequence of zero-mean white Gaussian measurement noise with covariance

$$E[\omega(k)\omega(k)'] = R(k)$$  \hspace{1cm} (4.6)
The models assumptions, considering the sample-time interval \( h \), are: (1) First-order model: the object undergoes a constant velocity during each \( h \); (2) Second-order model: the object keeps a constant acceleration during each \( h \); (3) Third-order model: the object keeps a non-constant (jerk) acceleration during each \( h \). Moreover, 

\[
\tilde{v}(t) = v(k), t \in [(k)h, (k+1)h]
\]

that is, the noise processes are uncorrelated from period to period (piecewise indication).

**Discrete-time linear kinematics models**

The kinematic state models are defined in a decoupled manner, that is, the motion along each coordinate is assumed decoupled (independent) from the other coordinates. More specifically, each coordinate \((x, y)\) of the 2D-Cartesian space is governed by its own equation, although with the same model. The noises entering into \( x \) and \( y \) coordinates are assumed to be mutually independent with possibly different variances. The state equation has the same form for all the models, as expressed in (4.4). The differences are in the transition matrix \( F \) and the noise gain \( \Gamma \).

**Constant velocity model**: the process is excited by the noise \( v(k) \) modeled as the noisy velocity at the \( k \)th sampling (considered constant during the period \( h \)), hence the position suffers an increment given by \( v(k)h \). In this model the transition and the noise matrix are:

\[
F = 1, \Gamma = h
\]  
(4.7)

The covariance of the process noise is: 

\[
Q = E[\Gamma v(k)v(k)\Gamma'] = \Gamma^2\sigma_v^2 = h^2\sigma_v^2
\]

The velocity (noise) uncertainty is of the order of the maximum velocity magnitude \( v_{\text{max}} \). Thus, \( 0.5v_{\text{max}} \leq \sigma_v \leq v_{\text{max}} \) (this consideration follows the recommendations given in [Bar-Shalom and Fortmann, 1988]).

**Constant acceleration model**: if \( v(k) \) is the noisy acceleration, assumed piecewise constant during \( h \), thus the increment in the velocity during this period is \( v(k)h \), and the effect in the object position is \( v(k)h^2/2 \). This model have matrices given by:

\[
F = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \Gamma = \begin{bmatrix} h^2/2 \\ h \end{bmatrix}
\]  
(4.8)
The covariance of this second-order process noise $Q$ is [Bar-Shalom and Li, 1995]:

$$Q = E[\Gamma v(k)v(k)\Gamma'] = \Gamma \sigma_v^2 \Gamma' = \begin{bmatrix} 1/4 h^4 & 1/2 h^3 \\ 1/2 h^3 & h^2 \end{bmatrix} \sigma_v^2$$

For this model $\sigma_v$ is of the order of the maximum acceleration magnitude $a_{max}$. Thus, $0.5a_{max} \leq \sigma_v \leq a_{max}$

**Wiener process acceleration:** in this model, $v(k)$ is the acceleration increment during $h$ and it is assumed to be a zero-mean white sequence - the acceleration is a discrete-time Wiener process [Bar-Shalom and Li, 1995]:

$$F = \begin{bmatrix} 1 & h & h^2/2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}, \Gamma = \begin{bmatrix} h^2/2 \\ h \\ 1 \end{bmatrix} \quad (4.10)$$

where $Q$ is

$$Q = \begin{bmatrix} 1/4 h^4 & 1/2 h^3 & 1/2 h^2 \\ 1/2 h^3 & h^2 & h \\ 1/2 h^2 & h & 1 \end{bmatrix} \sigma_v^2$$

\quad (4.11)

For this model, $\sigma_v$ should be of the order of the magnitude of the maximum acceleration increment over a sampling period, $\Delta a_{max}$. A practical range is $0.5\Delta a_{max} \leq \sigma_v \leq \Delta a_{max}$.

These models can be understood, in more general terms, in the sense that the corresponding (first, second, and third-order) derivatives of the position are actually not zero (as in a theoretically noiseless model), but a zero-mean random process entering into the system in the form of random input noise.

**Measurement model based on segmented objects**

For the tracking process, the vertical points were projected in a common plane: $\phi_i = 0$, and the Cartesian form was adopted, hence each laser-point takes the form $p_i = (x_i, y_i) : x_i = r_i \cos \alpha_i, y_i = r_i \sin \alpha_i$. Let $S_i(k)$ be a segment, at a given time instant $k$, the measurement model representation for the object under tracking associated to $i^{th}$ segment is determined by the centroid $[xc_i, yc_i]$ of a circle fitted around $S_i$. More
specifically, a circle is extracted from the points that define \( S_i \) and the circle-center is used as characteristic/reference point during the tracking process.

The predicted position of the segment is used as the center of a circle fitted on \( S_i \). Thus, the position and its derivatives, velocity and acceleration, are defined at the center of the circle (the reference point) extracted from the laser measurements contained in a segment \( S_i \). The order of the stochastic models presented in the previous section, can be seen as coefficients of polynomials, whose coefficients yield estimates of the position and its derivatives. A inevitable question arises: what is the appropriate order of the model to be used to predict the dynamics of a pedestrian? In order to answer this question, a set of experiments were performed using sequences of measurements collected with the laser platform mounted in ISRobotCar. These sequences, which are time varying \((k)\), correspond to pedestrians crossing the road just in front the vehicle: which is, probably, the most common situation in urban scenario.

The evolution, in time, of the position of a pedestrian is modeled as a polynomial given by [Bar-Shalom and Li, 1995]:

\[
x(k) = \sum_{j=0}^{o} a_j \frac{k^j}{j!} \tag{4.12}
\]

with the parameters \( a_j \), up to third-order \( o = 3 \), being the polynomial coefficients to be estimated. The coefficient \( a_j \) is the \( j \)th derivative of the position at the reference time \( k = 0 \). The estimate of the parameter vector \( a \), according to [Bar-Shalom and Li, 2001], is

\[
\hat{a}_j(n) = \left[ \sum_{k=1}^{n} h(k)h(k)\right]^{-1} \sum_{k=1}^{n} h(k)z(k) \tag{4.13}
\]

where \( z(k) \) is the measurement at instant \( k \), \( n \) is the number of measurements in the data, and \( h(k) = [1 \ t_k \cdots t_k^o] \). To decide which is the appropriate order of the model to fit pedestrian dynamics, the average of the squares of the residuals, in the LS sense, has been used:

\[
Err(k) = [z(k)H\hat{x}(k)](R(k))^{-1}[z(k)H\hat{x}(k)] \tag{4.14}
\]

where \( H = [1 \ 0 \ 0] \) is the measurement matrix, and \( \hat{x}(k) \) is the estimated state vector calculated replacing \( a_j \), in (4.12), by \( \hat{a}_j \) (4.13).
This topic is especially important because, in practice, the models - for instance, for the motion of pedestrians - are not known \textit{a priori} and have to be inferred from the measured data. The data used in the experiments have 25 sequences of laser readings corresponding to 25 pedestrians crossing the road in front the vehicle. In average, each sequence has 36.7 readings. The average quadratic errors, for each polynomial order, are shown in Table 4.2, from which it can be concluded that the third order polynomial is the best solution to fit pedestrian dynamics as perceived by the LIDAR. This polynomial refers to a Wiener acceleration process whose model is characterized by the matrices (4.10) and (4.11). A trial of the trajectory walked by a pedestrian, represented in the LIDAR space, is depicted in Fig. 4.10, as well as the polynomial curves fitted to the data.

\begin{table}[h]
\centering
\caption{Polynomial fitting up to third-order models}
\begin{tabular}{llll}
\hline
 & Average error (4.14) (in cm) & Linear & Quadratic & Cubic \\
\hline
X-axis & 124.59 & 40.58 & 14.07 & \\
Y-axis & 204.76 & 33.46 & 16.09 & \\
\hline
\end{tabular}
\end{table}
Chapter 5

Vision-based system

Studies and experiments in pedestrian classification and detection using a monocular camera are described in this chapter, which is divided in four sections: (1) descriptors for pedestrian classification; (2) pedestrian classification; (3) dataset resampling; and (4) pedestrian detection. The image-based descriptors used in the course of this thesis are HOG and COV, explained in Section 5.1, which comprises an array of 256 feature-components. Moreover, the Haar-like features are described in Section 5.1. The mRMR [Peng et al., 2005] method is applied to obtain an array of feature-components ranked as function of the maximum relevance, in the sense of information theory, and minimal redundancy. Section 5.2 is devoted to the problem of pedestrian classification, emphasizing the experiments using component-classifiers. A selective dataset sampling strategy using support vectors, presented in Section 5.3, is proposed to reduce the cardinality of the training dataset and to keep a set of representative examples for further training stages. Finally, in the last part of this chapter, rejection cascade methods are used to tackle the problem of pedestrian detection in image frames.

5.1 Descriptors for pedestrian classification

Image-based descriptors and features for pedestrian or people detection is a huge area of research and has had a continuous evolution in the recent years, evidenced by the surveys [Gavrila, 2001], [Enzweiler and Gavrila, 2009], [Geronimo et al., 2010], [Dollar
In this section the HOG and COV image-based descriptors and the Haar-like features are described.

**HOG descriptor**

Histogram of Oriented Gradients (HOG) refers to the set of features, proposed by [Dalal and Triggs, 2005], extracted from a dense and overlapping grid of well-normalized local histograms of image gradient orientations over image windows [Dalal and Triggs, 2005]. The Static HOG variant [Dalal, 2006], hereafter simply HOG, is of interest here since the detection is accomplished frame by frame. HOG features have been extensively exploited in the recent literature on pedestrian detection [Geronimo et al., 2010], [Enzweiler and Gavrila, 2009], and [Dollar et al., 2009]; although there are examples of HOG applications in other domains, such as animals and vehicles classification. Given an input image, the original HOG descriptors [Dalal and Triggs, 2005] are extracted according to four main processing steps: 1) Gamma and color normalization; 2) gradient computation; 3) weighted voting into spatial and orientation cells; 4) contrast normalization over overlapping spatial blocks. In [Dalal, 2006] it is presented details about the HOG implementation and classification performance effect, using a linear SVM classifier, regarding the various HOG parameters, such as: gamma correction, gradient filtering masks, size (in pixels) of the histogram blocks, normalization, size of the detection window and spatial stride.

Let $\partial I_x$ and $\partial I_y$ be the first-order gradient of the image $I(x, y)$ of a given detection window $DW_i$, thus $\arctan(\partial I_x, \partial I_y)$ and $\sqrt{\partial I^2_x + \partial I^2_y}$ define respectively the oriented gradient and the contrast intensity of $I(x, y)$. HOG-based features are computed from a $g \times g$ grid, which defines the number of cells in the detection windows, with $\eta \times \eta$ pixel cells, each cell containing $\lambda$ bins of the oriented histograms. The oriented histograms were divided in 9 parts ($\lambda = 9$), with angular difference equal to $2\pi/\lambda = 40^\circ$.

The HOG extractor used in this thesis\(^1\) is a rectangular cell HOG calculated on gray scale images according to the following steps: 1) a $L2$-norm gradient is calculated from the image of a detection window $I(x, y)$; 2) a $3 \times 3$ grid ($g = 3$) was applied over $I(x, y)$, resulting in a set of 9 square cells, each with $\eta^2$ pixels; 3) from each cell, a 9-bin normalized HOG is extracted; 4) the cells extend over 50% of their own area.

\(^1\)HOG and COV functions were implemented in Matlab by O.Ludwig [Ludwig et al., 2009].
5.1. DESCRIPTORS FOR PEDESTRIAN CLASSIFICATION

in order to reinforce the local information over the neighbors cells; 5) these 9 cells, each one with 9-bin HOG, are concatenated and a 81-dimensional HOG-based feature vector \( f_{hog} \) is obtained.

COV descriptor

The seminal work on region covariance descriptors for detection and classification is devoted to [Tuzel et al., 2006], where they proposed a region descriptor (here denoted COV) which uses the covariance of \( d \)-features extracted from the region of interest in the image. To compose the \( d \)-dimensional feature vector, in [Tuzel et al., 2006] it is used, among others, the 3-dimensional color vector. In our works [Ludwig et al., 2009], [Premebida et al., 2009c], and in this thesis, the color information is not used, thus a 8-dimensional feature vector \( f_{cov} \) is calculated. Given the image of a sliding-window detector \( I(x,y) \), where \( (x,y) \) represents pixel locations, the COV-features are comprised in the vector

\[
f_{cov} = [x,y, \partial I_x, \partial I_y, (\partial I_x^2 + \partial I_y^2)^{0.5}, \partial^2 I_x, \partial^2 I_y, \arctan(\partial I_y/\partial I_x)]
\]

(5.1)

where \( \partial I_x = |\frac{\partial I(x,y)}{\partial x}| \) and \( \partial I_y = |\frac{\partial I(x,y)}{\partial y}| \) are the norm of first order derivatives w.r.t. \( x \) and \( y \) respectively, \( \partial I_x^2 = |\frac{\partial^2 I(x,y)}{\partial x^2}| \) and \( \partial I_y^2 = |\frac{\partial^2 I(x,y)}{\partial y^2}| \) are the norm of second derivatives, and \( \arctan(\partial I_y/\partial I_x) \) is a term which account for orientation. The image derivatives were calculated through the masks \([-1,0,1]\) and \([-1,2,1]\).

Like in [Tuzel et al., 2006], five covariance matrices \( COV_{R_i} | i = 1, \cdots , 5 \) are computed from the image \( I(x,y) \) as illustrated in Fig. 5.1. From each region \( R_i \), a covariance matrix is calculated according to

\[
COV_{R_i} = \frac{1}{n-1} \sum_{k=1}^{n} (f_{cov_k} - \mu)(f_{cov_k} - \mu)^T
\]

(5.2)

where \( n \) is the number of pixels in a given region \( R_i \), and \( \mu \) is the mean-feature vector of the pixels. Due to symmetry, covariance matrices has \((d^2 + d)/2\) different values, where \( d = 8 \) (the feature vector dimensionality). Thus, from each region \( R_i \), a 36-dimensional feature vector is calculated, totalizing 180 feature elements.
One of the main disadvantages of using the covariance matrices is that its computation does not lie on Euclidean space. As pointed out by the authors [Tuzel et al., 2006], the COV-based feature space is not closed under multiplication with negative scales. Some, to not say most, of the common machine learning methods work on Euclidean spaces and therefore they are not 100% suitable for work with COV features; even though, our research group obtained relevant results using COV with usual classifiers, such as: SVM, LDA, and ANN.

![Figure 5.1: Covariance matrices are calculated in the regions $R_1, \ldots, 5$ of a given input image $I(x,y)$. A 36-dimensional feature vector $f_{\text{cov}}$ is obtained from each matrix $COV_{R_i}$, totaling 180 feature elements.](image)

Haar-like features

The basic [Viola and Jones, 2001] and the extended Haar-like features [Lienhart and Maydt, 2002] are based on the Papageorgiou’s object detection framework [Papageorgiou and Poggio, 2000] which describes an object class in terms of an overcomplete dictionary of local, oriented, multiscale intensity differences between adjacent regions, efficiently computable as a Haar wavelet transform [Papageorgiou and Poggio, 2000]. Basically, a set of 2D (image space) Discrete Wavelet Transform (DWT) are the basis for obtaining the set of, so named, Haar-features. The very popular Viola and Jones work [Viola and Jones, 2001] introduced the concept of Integral Image representation which allows the Haar-like features computation to be accomplished in very quick CPU time.

The feature set used in this work refers to the extended Haar-like features, c.f. [Lienhart and Maydt, 2002], which is available, in conjunction with the Viola’s cascade framework, in the OpenCV Library [Intel, 2011].
5.1. DESCRIPTORS FOR PEDESTRIAN CLASSIFICATION

5.1.1 Feature selection

As in the laser-based system, the mRMR [Peng et al., 2005] method is also used here for image-based feature selection. Even though some classification methods tend to obtain higher classification performance with large number of features, there are controversial issues in the use of many features because it increases the chance of overfitting. In some cases, a feature selection method is almost mandatory, specially when using classifiers which require independence among the features, such as Naive Bayes and Gaussian Mixture Models. Those classifiers are prone to inconsistencies (e.g., singularities on the covariance matrix, or likelihood value tending to zero [Paalanen et al., 2005]) as the number of features increases; moreover, the condition of independence among the features is a strong prerequisite in some classifiers, namely for the Naive model.

The rationale for using feature selection in this work are three folds: (1) avoid inconsistencies in classifiers which require independence among the features; (2) decrease the classifiers complexity to avoid overfitting and also to decrease computational time; and (3) as a tool to control the complexity, which is proportional to the number of features, in cascade methods.

The mRMR adopts, as measure of relevance \( \text{Rel} \), the normalized conditional mutual information \( \text{Rel} = I(f_i; y|f_S) \) of the variable \( f_i \) to the output \( y \), conditioned to a selected set \( f_S \) (see Appendix A). Table 5.1 summarizes the results of experiments, using the classification dataset (see Table 3.5 in Section 3.6.1), in terms of the most relevant features to explain the target output as function of fractions of 25% of the feature set. It can be concluded that the feature set associated with the COV descriptor brings more relevance to the output than the HOG-based features. However, the COV set has more than twice as many features.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOG</td>
<td>32.8%</td>
<td>23.4%</td>
<td>26.6%</td>
</tr>
<tr>
<td>COV</td>
<td>67.2%</td>
<td>76.6%</td>
<td>73.4%</td>
</tr>
</tbody>
</table>

Table 5.1: Vision-based features relevance (in %)
5.2 Pedestrian classification

Pedestrian classification, and more recently pedestrian detection, are areas in which considerable progress has been made towards practical and realistic solutions. Although the former is a mature field of research, the later still face a considerable gap between prototypes and realistic solutions. This section brings into focus pedestrian classification, with comparison and study of several classification techniques, and converges to detection (in the next section), since the later depends on classification methods.

Image-based pedestrian detection is a subject that had deserved much attention of the scientific community and also the automotive industry along the past decade; for instance, recently Volvo had deployed a safety system which claims to detect standing or moving pedestrians at daylight conditions [ULTra, 2010]. This is basically a collision warning system that fuses radar and image, where the radar is primary used to detect objects in front of the car and the camera-system determines the class (namely car or pedestrian) of the objects.

Let \( f \) be an image-based feature vector, extracted from a cropped image or from a detection window, a classifier can be seen as the function:

\[
L = F(f, \phi) \tag{5.3}
\]

where \( L \) is a confidence value, or the class conditional probability\(^2\) in some classifier models, and \( \phi \) is the classifier's parameter vector. The critical steps behind pedestrian classification are: feature-vector definition, classifier selection, and training model\(^3\). Except in the Haar-AdaBoost cascade discussed in Section 7.5.2, all the classification methods use a feature vector obtained from HOG and COV descriptors. As detailed in previous sections, \( f_{hog} \) has 81 elements and \( f_{cov} \) has 180, thus \( f \) is a 261-dimensional feature vector. However, if the feature selection is used, which may be necessary for some classification methods, then \(|f| < 261\). The methodology adopted for the selection of the most appropriate classifier, necessary for any machine learning problem, has the objective of obtaining the classification method with the best generalization property. The trade-off between overfitting and generalization is behind the paradigm

\(^2\) It is valid if a consistent probabilistic framework is used to model the classifier.
\(^3\) Actually, an appropriate training model selection is one of the most important problems in machine learning and pattern recognition [Duda. et al., 2001] [Kuncheva, 2004] [Vapnik, 1998].
of classifier selection problem. In this section, cross-validation is the technique used to estimate the performance and to select the parameters of the classifier models. In Chapter 6, structural risk minimization will be used for training SVM-based cascade classifiers.

Five classification methods have been used for pedestrian classification; these methods are designated by the acronyms *LDA, NBC, GMM, SVM, ANN* (the readers are referred to chapter 3 for a brief description of these classifiers). The parameters, denoted by $\phi_i$, of the classifiers were chosen based on K-fold cross-validation supported by a sort of performance metrics: BER, Acc, ROC, AUC. Table 5.2 summarizes the classifier’s parameters chosen after the cross-validation process.

Table 5.2: Classifiers parameters

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>LDA</em></td>
<td>complete feature set</td>
</tr>
<tr>
<td><em>NBC</em></td>
<td>75% of the feature set</td>
</tr>
<tr>
<td><em>GMM</em></td>
<td>2 mixture components, 75% of the feature set</td>
</tr>
<tr>
<td><em>SVM</em></td>
<td>linear kernel, margin parameters $= 0.01$ (pos) and $0.05$ (neg)</td>
</tr>
<tr>
<td><em>ANN</em></td>
<td>Feedforward, 3 neurons, Gradient Descendent training</td>
</tr>
</tbody>
</table>

Table 5.3: Classification performance results

<table>
<thead>
<tr>
<th>$F$</th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>$TP_{10%}$</th>
<th>$AUC_{10%}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>LDA</em></td>
<td>0.902</td>
<td>0.850</td>
<td>0.164</td>
<td>0.728</td>
<td>0.032</td>
</tr>
<tr>
<td><em>NBC</em></td>
<td>0.956</td>
<td>0.875</td>
<td>0.113</td>
<td>0.870</td>
<td>0.067</td>
</tr>
<tr>
<td><em>GMM</em></td>
<td>0.950</td>
<td><strong>0.886</strong></td>
<td>0.139</td>
<td>0.841</td>
<td>0.059</td>
</tr>
<tr>
<td><em>SVM</em></td>
<td>0.928</td>
<td>0.841</td>
<td>0.139</td>
<td>0.772</td>
<td>0.046</td>
</tr>
<tr>
<td><em>ANN</em></td>
<td><strong>0.971</strong></td>
<td>0.841</td>
<td>0.127</td>
<td><strong>0.950</strong></td>
<td><strong>0.074</strong></td>
</tr>
</tbody>
</table>

Summary

The classification performance of the vision-based single classifiers were evaluated and compared in terms of AUC, Acc, BER, $TP_{10\%}$, $AUC_{10\%}$, and also using ROC curves. The classifiers trained and tested with the full set of features are *LDA, SVM, and ANN*, while *NBC* and *GMM* were employed with 75% of the most relevant features. Table 5.3 shows an interesting aspect regarding Acc and BER. Acc indicates *GMM* as the best classifier, although the *NBC* has the minimum BER thus, since the
CHAPTER 5. VISION-BASED SYSTEM

Figure 5.2: ROC curves of the classifiers obtained with the average of 4-fold cross validation on the vision-based classification dataset.

experiments were carried out with imbalanced set, BER is more appropriate. All the remaining metrics (AUC, TP_{10\%}, AUC_{10\%}), evidenced in the ROC curves (Fig. 5.2), indicate the ANN as the classifier with the best performance.

5.3 Dataset resampling

Before discussing image-based pedestrian detection, it is important to punctuate a specific and important characteristic of realistic (with large cardinality) datasets used for pedestrian detection in urban scenarios: the large difference between the number of positives and negatives examples. To give a clear insight about this, let us consider the size of a pedestrian detector as a rectangular window (region), in the image plane, with height $h_i$ and width $w_i$, in pixels, where $h_i = 2w_i$. With $s_{i=1,\ldots,5} = (1.0, 1.1, 1.3, 1.5, 2.0)$ denoting the scales, the pedestrian detector windows are shifted through the image with size and locations varying as function of $s_i, w_i$, and $h_i$ in a multiscale sliding window way [Enzweiler and Gavrila, 2009]. For the case
of an image with 640x480 pixels, and considering \( w = (27, 30, 35, 40, 54) \), it generates, depending the approximations done to keep the detector locations with integer pixel values, at least 56550 detection windows per frame. In a dataset with 10000 frames, and for an average rate of 0.5 pedestrian per frame (which is a reasonable consideration for urban scenarios), it gives at least \( 5.6 \times 10^8 \) negative sliding-windows against 5000 positives. Keeping a relaxed ratio in mind of \( 1 : 1.0 \times 10^5 \), a training dataset with 5000 positives windows should have, respecting the previous proportion, the large amount of \( 5 \times 10^8 \) negatives. To avoid bias problems and infeasible computational requirements in such large imbalanced datasets, a under-sampling algorithm is desirable [Kang and Cho, 2006]. To preserve the information which is relevant to compute the classifier separating hyperplane and, at the same time, to reduce the training dataset cardinality and to decrease the complexity of the training process, two under-sampling methods were implemented: (1) a LDA-based method is proposed as a practical and efficient solution (due its low computational requirements and linear characteristic); and (2) a SVM-based data selection (under-sampling), which is inspired in the parallel SVM architecture introduced in [Graf et al., 2004]. Both methods can be used to obtain a balanced or soft-imbalanced dataset.

The notations used in the sequel are:

- \( X_P = \{X^{(i)}_P : i \in I_P = \{1, \cdots, np\}\} \) is the set of positive training examples.
- \( X_N = \{X^{(i)}_N : i \in I_N = \{1, \cdots, nn\}\} \) is the set of negative training examples.
- \( X_{P,N} = X_P \cup X_N \) is the complete training set composed by \( X_P \) and \( X_N \).
- \( X^{S}_N = \{X^{(i)}_N : i \in S_n \subset I_N\} \) is the subset of selected negative training set.
- \( X_{N-\Omega} = \{X^{(i)}_N : i \in I_N \setminus \Omega\} \) is the subset of negative examples with those in \( \Omega \) removed, where \( \Omega \subset X_N \).
- \( SV_{neg} = \{SV \subset X_N\} \) is the subset of negative instances which correspond to the support vectors of a given linear SVM.

The number of negative samples \( nn \) in the LIPD detection dataset heavily outnumber the positives \( np \), hence \( np \ll nn \). For this reason, the resampling algorithms applied in this work selects, from the negative training set \( X_N \), a subset of instances \( X^S_N \) with \( |S_n| \ll |I_N| \). In short terms, the LDA-based method selects each instance \( X^S_{N(i)} \) which is close to the decision hyperplane defined by a LDA classifier trained with a subset of examples. On the other hand, the SVM-based data resampling algorithm selects, from the negative training set \( X_N \), a set of instances which corresponds
to support vectors \((SV_{neg})\).

**LDA-based down-sampling method**

The complete training set \(X_{P,N}\) has high cardinality, mainly due to the number of examples in \(X_{N}\), which turns infeasible to train a single \(LDA\) classifier with all the training examples at once. To solve this problem, an iterative process is proposed where, at each \(i^{th}\) step, the negative set is replaced by a smaller set \(\Omega_{i}\) therefore, the process is decomposed in treatable sub-problems: for each step a single \(LDA\) classifier is trained using \(\{X_{P}, \Omega_{i}\}\), where \(\Omega_{i}\) corresponds to a subset of \(X_{N}\). Then, the trained \(LDA\) is used to obtain the confidence value \(w.r.t.\) each example of \(\Omega_{i}\). The examples wrongly classified as positives (i.e., the false positives) are used to compose the selected set \(X_{N}^{S}\). The process repeats until all the negative examples have been used.

In summary, the final selected set \(X_{N}^{S}\) corresponds to the false positives obtained by a set of \(LDA\) classifiers recursively trained and tested in sub-sets of \(X_{N}\) (see Algorithm 3). In the limit, if the memory capacity and the computer facilities permits, the set \(X_{N}-\Omega\) tends to be replaced by \(X_{N}\) and the number of iterations \(n\) tends to one.

---

**Algorithm 3** LDA-based down-sampling method for negative examples selection

**Input:** \(X_{P,N} = \{X_{P}, X_{N}\}\): training set;  
**Output:** \(X_{N}^{S}\): set of selected negatives;  
1. \(\{FP\}_{i}\): set of false positives; \(n\): number of iterations;  
2. \(\Omega_{i}\): subset of \(X_{N}\);  
   // Initialization:  
3. \(X_{N}^{S} \leftarrow \{\}\): empty set;  
4. Decompose \(X_{N}\) in \(n\) subsets \(\Omega_{i}\), where \(\Omega_{i} \subset X_{N}\);  
   // Selecting samples:  
5. for \(i = 1; i < n; i+1\) do  
6. \(\text{Train a } LDA \text{ classifier using } \{X_{P}, \Omega_{i}\}\);  
7. \(\text{Select false positives, from } \Omega_{i}, \text{ to generate } \{FP\}_{i}\);  
8. end for  
   // Composing \(X_{N}^{S}\):  
9. \(X_{N}^{S} \leftarrow \{FP\}_{i}, i = 1, \ldots, n\).
5.4. PEDESTRIAN DETECTION

SVM-based down-sampling method

Given the initial training set $X_N$, with $nn$ negative training examples, this undersampling algorithm selects $ns$ instances which correspond to the support vectors of $X_N$, where $|SV_{neg}| = ns$. The first step of the resampling algorithm is to split $X_N$ in $n$ subsets $\Omega_i \subset X_N, i = 1, \ldots, n$; further, for each subset $\Omega_i$, a SVM classifier is used to extract the support vectors which will be used to compose $SV_{neg}$. Thus, each $i^{th}$-SVM is trained with a subset comprising $np$ positives and $nn$ negatives. The final step is to aggregate all the negative support vectors obtained from the $n$ SVMs. Lastly, this method outputs the selected negative training set which is composed by the negative support vectors: $X_N^S \leftarrow SV_{neg}$ (see Algorithm 4). Figure 5.3 illustrates, in the row (a), some negative samples whose training feature vectors do not correspond to support vectors and, in the row (b), some samples which correspond to $SV_{neg}$.

![Figure 5.3](image)

Figure 5.3: Negative samples which possess feature vectors positioned ‘far’ from the separation margin (a); and samples which correspond to support vectors (b).

5.4 Pedestrian detection

The pedestrian detection task involves a number of spatio-temporal processing techniques aiming to obtain, in the image frame, the estimated position and the size (scale) of potential pedestrians. Instead of scanning the entire frame, image-based pedestrian detection problem is constrained to ROIs i.e., the image search space is significantly reduced compared to the full frame. The ROI generation, or hypothesis generation, method employed in this work uses laser-segments, obtained from the LIDAR-based system, which are transformed into image plane as a set of ROIs. Inside
Algorithm 4 SVM-based down-sampling method for negative examples selection

Input: \(X_{P,N} = \{X_P, X_N\}\): training set;
Output: \(X^S_N\): set of selected negative samples;

1. \(\{SV_{neg}\}_i\): set of negative support vectors; \(n\): number of iterations;
2. \(\Omega_i\): subset of \(X_N\);
   // Initialization:
3. \(X^S_N \leftarrow \{\}\): empty set;
4. Decompose \(X_N\) in \(n\) subsets \(\Omega_i\), where \(\Omega_i \subset X_N\);
   // Selecting samples:
5. for \(i = 1; i < n; i+1\) do
6.   Train a linear-SVM classifier using \(\{X_P, \Omega_i\}\);
7.   Select support vectors, from \(\Omega_i\), to generate \(\{SV_{neg}\}_i\);
8. end for
   // Composing \(X^S_N\):
9. \(X^S_N \leftarrow \{SV_{neg}\}_i, i = 1, \cdots, n\).

Each region, designated by \(ROI(i)\), a detection window \(DW_i\) is used in a multiscale sliding approach: \(DW_i\) is shifted through \(ROI(i)\) by varying the location \((x_i, y_i)\) and the size \((w_i, h_i)\) as function of the spatial stride step and the scale factor (see Fig. 5.4).

Figure 5.4: Representation of a detection window, \(DW_i\), with size defined by \((w_i, h_i)\), which is shifted, inside the ROI, at locations given by \((x_i, y_i)\).

5.4.1 LIDAR-based ROI generating method

A key problem in monocular image-based pedestrian detection, in the field of ADAS applications [Enzweiler and Gavrila, 2009], [Gandhi and Trivedi, 2007], is the huge
5.4. PEDESTRIAN DETECTION

amount of negatives (potential false alarms) in contrast with the number of positives, what demands a vast processing time consumption and a high confidence detector. To avoid using a multiscale sliding windows approach in the full image, a LIDAR-based ROI generation method is proposed. The idea is to use the LIDAR-based system, discussed in Chapter 4, acting as a primary object detection system, where each detected object, which is represented by a segment, constitutes a hypothesis of being a pedestrian or a non-pedestrian. The LIDAR-based system outputs a set of segments that are transformed into image coordinates as ROIs.

The number of detection windows, used to scan the ROIs in searching for pedestrian evidence, is limited and it is defined by the size of the ROI (as illustrated in Fig. 5.4). This approach decreases the computational processing time, restricting the zones of interest in the image by a dozen of ROIs at most, and keeping the false positives at low values. For instance, the number of detectors generated by this LIDAR-based approach is in average thousands times lower than the usual full-scanning image approach.

Additionally, some experiments have been performed using the image-based focus of attention method proposed in [Walther and Koch, 2006] as an attempt to restrict the system to salient regions, namely, to avoid regions in the image typically covered by the sky and the road and, at the same time, to compare this method with the proposed laser-based detector. That method can in fact reduce drastically the number of searching regions in the image but, as counterpart, a significant number of positives (pedestrians) tends to be missed. Moreover, the scale can not be directly estimated. To summarize, the LIDAR-based ROI generating method has two major advantages: (1) decreasing the time-processing and complexity; (2) enabling range (scale) information.

Detection window’s scale

Two approaches were considered regarding the detector’s scale: (1) the image $I(x,y)$, associated to the detector, is resized before applying the classification method, and (2) the counterpart case, $I(x,y)$ suffers no resizing. Two experiments were carried out to chose between the aforementioned approaches. These experiments involve the training and the testing using three classification methods ($LDA$, linear-$SVM$, and $ANN$) in a dataset composed of HOG and COV feature vectors extracted from a set
of images with size varying from 54x27 to 108x54 pixels according to the scale $s$ (see Section 5.3). In the first experiment, the images associated to the detection windows are resized, using the `imresize` Matlab function (`imresize([h,w],[54,27],'bilinear')`), before the feature vector calculation. While, for the second case, the images of the dataset suffered no resizing before applying HOG and COV descriptors.

The results of these experiments, in a dataset with 5146 positive and 10292 negative examples, are given in Table 5.4. It can be concluded that the performance obtained with non-resized images is slightly better than resized ones, for all the classifiers. Moreover, extra processing time is avoided in non-resized images. Thus, on the remaining experiments on pedestrian detection, the images associated to the detection windows do not suffer resizing.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Acc (non-resized)</th>
<th>BER (non-resized)</th>
<th>Acc (imresize)</th>
<th>BER (imresize)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.968</td>
<td>0.047</td>
<td>0.966</td>
<td>0.053</td>
</tr>
<tr>
<td>SVM</td>
<td>0.958</td>
<td>0.076</td>
<td>0.952</td>
<td>0.091</td>
</tr>
<tr>
<td>ANN</td>
<td>0.980</td>
<td>0.020</td>
<td>0.978</td>
<td>0.025</td>
</tr>
</tbody>
</table>

5.4.2 Rejection cascades

A cascade ensemble, or multistage classification, can be understood as a degenerated decision tree [Duda. et al., 2001], where the nodes in such tree represent a component classifier in a cascade, and the links/branches are considered mutually distinct and exhaustive [Duda. et al., 2001]. The cascades considered in this work perform negative rejection, as in [Viola and Jones, 2001]. Thus, at each stage the instances classified as negatives are eliminated and, conversely, the positives follow through the cascade structure; this is illustrated in Fig. 5.5. The number of component classifiers defines the number of stages $N_c$ in a cascade, where the final decision regarding the instance’s class depends on the decision of all the preceding stages.
Figure 5.5: Functional diagram illustrating the negative-rejection cascade detector: each stage $k$ rejects the samples classified as negative (F: false), while the positives (T: true) pass through all the stages to be finally detected as a pedestrian.

**SVM-cascade**

The proposed SVM-cascade\(^4\) is a rejection cascade trained using a boosting process where the number of features, in a given stage, increases w.r.t. to the preceding stage; thus, the complexity of the cascade and its classification capacity increase as more stages are added to the structure. The SVM-cascade is a negative-rejection cascade of linear-SVMs which follows the training process presented in Algorithm 5. Each stage of the cascade is trained to achieve a given true positive rate ($TPR$), by adjusting the decision hyperplane of the $SVM$ in the current stage, while rejecting the negatives correctly classified. This process is performed varying the bias of the component $SVM$ until $TPR$ is achieved. The subsequent stage of the cascade receives all positives and the false positives from the previous stage, consequently, the training set becomes more and more difficult to predict the correct classes. To improve the classification capacity, the number of features is incremented progressively as the number of stages increases; that is, the number of features and consequently the complexity of a current stage is increased by adding more features w.r.t. the previous stage. The feature vector was previously ordered as function of the maximum relevancy and minimal redundancy (obtained using the mRMR method), and the number of features $nf$ was selected to be constant and equal to 10; thus the first stage has 10 features, the second 20, and so on. The training method is summarized in Algorithm 5.

The SVM-cascade minimizes the empirical risk (3.23) in a particular way where

\(^4\) Called Boosting-SVM in [Ludwig et al., 2011].
Algorithm 5 SVM-Cascade training process

**Input:** $D = \{X_P, X_N\}$, $np$, $nn$: training set, number of positives, and negatives;

**Output:** $\{W\}$: set of parameters of the cascade;

1. $TPR$: true positive rate;
2. $Thr_{tp}$: adopted threshold for $TPR$ (usually $Thr_{tp} > 0.98$);
3. select and order the features with maximum relevance and minimal redundancy in the array $S$, where $|S| = nfea$;
4. nf step: increment on the number of features (in our case, $nf = 10$);
5. $\Omega \leftarrow \{\}$: set of false positives;
6. $i \leftarrow 1$: cascade stage;
7. $D_i \leftarrow \{X_P, X_{N-\Omega}\}$;
8. for $n = 1; n < nfea; n+nf$ do
9. $D_i^n$: training set with $n$ features, where $S_i[i = 1, \ldots, n]$ is the set of features in the $i$th cascade stage;
10. train a linear SVM, using $D_i^n$, in order to obtain the SVM parameters $W_i = (w_i, b_i)$;
11. calculate the true positive rate $TPR$;
12. while $TPR < Thr_{tp}$ do
13. $b_i \leftarrow b_i - 0.05$: decrease the threshold in order to increase $TPR$;
14. recalculate $TPR$ using the current SVM bias $b_i$;
15. end while
16. end for
17. using the cascade with $i$ stages, collect false positive occurrences $\Omega$, where $\Omega \in X_N$, to compose the training set for the next stage: $D_{i+1} = \{X_P, X_{N-\Omega}\}$;
18. $i \leftarrow i + 1$;

the error due to the false positives is decreased much more accentually than the error corresponding to false negatives. This behavior is of great importance in high imbalanced datasets. As the number of stages $Nc$ and the number of features $nf$ increase, $R_{emp}(\alpha)$ decreases, leading to an inevitable question: what is the stopping criteria for $Nc$ and $nf$? An interesting solution for this problem is the ERM principle, introduced in Chapter 3, which can be summarized in three steps. For each $SVM$ model in a given stage $i$: (1) calculate its VC-dimension $VCd_i$, (2) compute
5.4. PEDESTRIAN DETECTION

As the number of features per stage is known, the solution for the number of stages $N_c$ in the cascade is obtained when $R(\alpha_i)$ achieves its minimum. Assuming a probabilistic factor $\eta = 0.1$ [Vapnik, 1998], the key problem resumes in obtaining $V C_d$. For the SVM-cascade, once the component classifier is a linear SVM, $V C_d \leq n f_i + 1$. Thus, knowing $\eta$, $n_i$ (the number of examples in $i$th stage), and $V C_d$, the upper bound is computed (3.25) and $R(\alpha_i)$ is obtained (see 3.5). The curves in Fig. 5.6 result from an experiment using the SVM-cascade. The minimum in the empirical risk should be obtained for $N_c \gg 25$ but, potentially, the higher the value of $N_c$ the cascade tends to overfit and, consequently, the generalization will be poor. On the other hand, the minimal structural risk is achieved at $N_c=25$; although the empirical risk tends to decrease for a high number of stages (much greater than 25), theoretically the cascade is less susceptible to overfitting when

$$N_c = \arg \min_{V C_d, \alpha} R_\alpha$$  \hspace{1cm} (5.4)

The true positive and false positive rates, on the training set, for the SVM-cascade are obtained by the product of the corresponding rates of its stages, i.e., $TPR = \prod_{i=1}^{N_c} TPR(i)$, and $FPR = \prod_{i=1}^{N_c} FPR(i)$. The results are given in Fig. 5.7. As expected, the results, on the training set, indicate a reduction on the true positives less accentuated than the false positives.

![Figure 5.7: Results on the training dataset in terms of $FPR$ (left part) and $TPR$ (right part), as function of the number of component-classifiers, using the SVM-cascade.](image-url)
AdaBoost-based cascade

The Adaptive Boosting (AdaBoost) algorithm, which was introduced by [Freund and Schapire, 1995], is one of the most used algorithms in rejection cascades, evidenced by numerous of applications and publications [Viola and Jones, 2001], [Freund and Schapire, 1995], [Friedman et al., 2000], [Sun, 2007], [Monteiro et al., 2006a], [Enzweiler and Gavrila, 2009]. In this section, the Gentle AdaBoost algorithm of the GML AdaBoost Matlab Toolbox [Vezhnevets, 2008] is used in the negative-rejection cascade, where the weak learner is a decision tree with one split i.e., a stump weak learner. The AdaBoost is trained, using HOG and COV feature vectors, according to Algorithm 6. Differently from the cascade framework introduced by P.Viola and M.Jones [Viola and Jones, 2001], which is available in OpenCV [Intel, 2011], where a desirable true positive and false positive rates are chosen, the GML AdaBoost implementation tries to minimize the empirical error directly thus, some precautions have to be taken in imbalanced datasets.

Algorithm 6 AdaBoost training algorithm

\textbf{Input:} $D = (x_1, y_1), \ldots, (x_m, y_m)$: is training set, where $x_i \in \mathcal{X}, y_i \in \mathcal{Y} = \{-1, 1\}$. $\mathcal{X}$ is the HOG-COV feature space.

\textbf{Output:} $\{\alpha, h\}$: set of base learners;

1: Initial distribution: $D^1(k) \leftarrow 1/m$;
2: for $k = 1; k < N; k + 1$ do
3: \hspace{1em} Train base learner $h_k \rightarrow \mathcal{Y}$, using $D^k$;
4: \hspace{1em} Update parameter $\alpha_k$:
5: \hspace{2em} $\alpha_k = 0.5 \log \left( \frac{\sum_{i:h_k(x_i)=y_i} D^k(i)}{\sum_{i:h_k(x_i) \neq y_i} D^k(i)} \right) = 0.5 \log \left( \frac{1-\epsilon_k}{\epsilon_k} \right)$
6: \hspace{1em} $\epsilon_k$: is the classification error using $h_k$;
7: \hspace{1em} Update and normalize the sample weights distribution:
8: \hspace{2em} $D^{k+1}(i) = \frac{D^k \exp(-\alpha_k h_k(x_i)y_i)}{Z_k}$;
9: \hspace{1em} $Z_k$: is the normalization factor.
10: base model: $\alpha_k, h_k$;
11: end for
12: final classifier output: $\text{sign} \left( \sum_{k=1}^{N} \alpha_k h_k(x) \right)$;

AdaBoost for imbalanced dataset: M-AdaBoost

Designating the weak learner by $h_k$ and the weighting parameter by $\alpha_k$, the AdaBoost algorithm iteratively updates (for each round $k$) the weights over the training set distribution $D(k)$ [Freund and Schapire, 1996], where the weighting parameter increases
5.4. PEDESTRIAN DETECTION

Figure 5.8: Results on the training dataset in terms of $TPR$ (left part) and $FPR$ (right part), as function of the number of stages, for the S-AdaBoost (in black) and for the M-AdaBoost (in blue).

for misclassified examples and decreases for correctly classified examples. The problem in the AdaBoost is the weighting strategy, which treats the training examples equally distributed, thus the weight parameters, independently of the class (positives or negatives), are increased or decreased uniformly over the training set distribution. When dealing with strong imbalanced datasets, this ‘traditional’ strategy is not satisfactory. The proposed idea is to control the distribution $D_k(i)$ as function of the class label. A similar idea was published by [Sun, 2007], with the same objective of this work: "the learning objective in dealing with the imbalance class problem is to improve the identification performance on the small class" [Sun, 2007].

The adopted strategy to control the sample distribution $D_k(i)$ is the multiplication by a coefficient $C_i$ to obtain a new non-uniform distribution $C_i D_k(i)$, where the value of $C$ depends on the class label $y$; thus, the explicit expression is $C = \{ C_1(y_1), \ldots, C_n(y_n) \}$. The framework used to train the M-AdaBoost is basically the same as shown in Algorithm 6, except that $D_k(i)$ is substituted by $C_i D_k(i)$. For the minority class (the positives, $y(i) = 1$) $C_i = \frac{n_{p_k}}{n_{p_k} + n_{n_k}}$, and for the negatives ($y(i) = -1$) $C_i = \frac{n_{n_k}}{n_{n_k} + n_{p_k}}$, where $n_{p_k}$ is the current number of positives and $n_{n_k}$ refers to the number of negative examples. Figure 5.8 shows the results, using the training set, in terms of $FPR$ and $TPR$, as function of the number of weak-classifiers, for the ‘standard’ AdaBoost (hereafter called S-AdaBoost) and for the M-AdaBoost.
Chapter 6

Information fusion system

This chapter provides qualitative and quantitative studies of the strategies adopted in sensor fusion for pedestrian classification and for ego-motion estimation. The problem of pedestrian classification using data from LIDAR and from monocular camera is approached by means of two strategies, centralized and decentralized, according to the fusion method being used, which can be: nontrainable fusion rules, trainable fusion rules, multistage scheme. The estimation of the vehicle ‘ego-motion’, which is necessary for object tracking, is done using a KF-based approach, where the observation process receives information from an encoder and a DGPS.

6.1 Information fusion for pedestrian classification

Camera and LIDAR data fusion has the objective of taking advantage, as maximum as possible, of the strengths of both sensors (in a complementary or redundant manner) and at the same time minimizing the weaknesses, in order to improve the overall system performance. Monocular cameras, operating in the visible spectrum, are the most common technology used for pedestrian classification and detection. The technology is mature, the cost is affordable (in most of the cases), the number of related works using imaging sensors is vast, the amount of information captured by a camera is richness, and the acquisition-time is short. All these advantages, among others, make the cameras the most widely used sensors for pedestrian safety systems. On the other hand, a system having a LIDAR perception module brings some advantages:
good range accuracy, robustness to circumstantial variability in the environment (e.g., illumination, temperature, humidity), the segmentation process, or background subtraction, is facilitated, and the LIDAR provides direct distance and rough geometrical information, e.g., shape and dimension, of the perceived objects.

Nevertheless, each of these technologies have drawbacks. Monocular visible spectrum cameras are strongly affected by glaring sources of light and by changes in the weather conditions. Moreover image processing and image analysis, for pedestrian classification and detection, is complex. Disadvantages of non-3D\(^1\) LIDAR, such as the Ibeo Alasca-XT used in this work, are: ‘vertical’ information is very restrictive, it is prone to miss partial occluded pedestrians, it demands more electrical consumption, the cost is sometimes prohibited, and some weather conditions affect the sensor (e.g., fog and rain). Thus, it is a common sense that every sensor technology has its advantages and limitations.

The architectures used for multisensor data fusion is frequently distinguished among three alternatives [Hall and Llinas, 1997]: (1) Direct/raw data fusion: sensor data are directly combined since the sensors are commensurate, i.e., they are measuring the same physical phenomena; (2) Intermediate/feature level: a set of representative attributes (features) are extracted from sensor data; the feature vectors can be combined in a single feature representation (centralized scheme) or processed separately (decentralized architecture); (3) High-level fusion: each sensor data is processed, throughout a decentralized scheme, to achieve high-level inferences that are subsequently combined. Combining LIDAR and image information, with the purpose to overcome the limitations of a each sensor and to enhance the performance of the overall system, are conceptually represented by two strategies:

1. Centralized architecture: LIDAR and camera information are combined at the feature level, thus a common/central classification framework is used to process the laser and the image-based features as a single, and jointly, feature vector.

2. Decentralized architecture: information from each sensor is processed separately, hence the output of each sensor-based module, in the form of a likelihood or a confidence-score, is combined in the decision level.

In terms of spatio-temporal integration, the laser scans and the image frames are

\(^1\)An example of a 3D LIDAR is the Velodyne HDL-64E sensor.
6.2. COMBINING CLASSIFIERS

combined according to the following:

1. Spatial fusion: once both sensors are calibrated, the laser-segments define ROI in the image plane. Consequently, the same object can be processed in both sensor space.

2. Temporal fusion: as described in Chap. 3, the data acquired by the laser and by the camera are synchronized according to a common time stamp, hence there is a direct scan-by-frame correspondence in the dataset.

6.2 Combining classifiers

According to Kuncheva [Kuncheva, 2004], there are two main strategies in combining classifiers: fusion and selection. In classifier fusion the whole feature space is supposed to be known by all component classifiers. Conversely, in classifier selection each component classifier is supposed to learn a specific part of the feature set. Moreover, a multiple classifier system can be designed according to four approaches [Kuncheva, 2004]: (1) Data level: using different data subsets; (2) Feature level: using different feature subsets; (3) Classifier level: using different component classifiers; (4) Combination level: designing different combiners.

Multiple classifiers fusion and selection combination strategies are used according to the architecture and the approaches mentioned previously. Before discussing the fusion architectures and the classifier combination methods, it is convenient to present the single\footnote{The terms single, component, or base classifier will be used (sometimes) interchangeably.} classifiers and the way they are used on combining classifiers.

6.2.1 Single classifiers scheme

The set of single classifiers used in the context of multiple classifier ensemble is the same as introduced in Chap. 3, namely: LDA, NBC, GMM, SVM, and ANN. These classifiers are employed in the centralized and in the decentralized architectures, although the way they are used differs according to one of the following approaches:
1. Data level: single classifiers are trained in specific parts of the dataset. The proposed multistage ensemble is an example of this approach.

2. Feature level: single classifiers are used in different feature sets, namely LIDAR (c.f., Chap. 4), vision (HOG/COV) (c.f., Chap. 5), and combined feature set (LIDAR and vision features).

3. Classifier level: different (heterogeneous case) or similar (homogeneous case) single classifiers can be used, in a fusion ensemble, to support the classification decision. The experiments reported here are devoted to the heterogeneous case.

4. Combination level: the outputs of single classifiers are combined through fusion methods such as: majority vote, median, maximum, trainable fusion rules.

### 6.2.2 Nontrainable fusion rules

A fusion rule is used to combine the outputs, which usually follows a probabilistic distribution, of a set of single classifiers in order to provide a final classification decision. Denoting $\mathcal{L}_i$ the confidence score\(^3\) yielded by classifier $\Theta_i$, ($i = 1, \cdots, nc$), where $nc$ is the number of component classifiers, four fusion rules are considered: average $\mathcal{F}_{\text{Average}}(\Theta_i)$, maximum $\mathcal{F}_{\text{Max}}(\Theta_i)$, majority vote $\mathcal{F}_{\text{Mode}}(\Theta_i)$, and Naive-product $\mathcal{F}_{\text{Nprod}}(\Theta_i)$. The average rule simply calculate the simple mean of the component classifier outputs:

$$\mathcal{F}_{\text{Average}}(\Theta_i) = \frac{1}{nc} \sum_{i=1}^{nc} \mathcal{L}_i \quad (6.1)$$

The maximum rule outputs the maximum value over the classifier responses:

$$\mathcal{F}_{\text{Max}}(\Theta_i) = \max_i \{\mathcal{L}_i\} \quad (6.2)$$

while the majority vote takes the *simple majority* decision among all classifiers decisions. Considering that a classifier decision assumes the values -1 or 1, that is

---

\(^3\)e.g., the value of the likelihood function that models the classifier output.
\[ \text{sign}(L_i) \in \{-1, 1\}, \text{ therefore:} \]

\[ F_{\text{Mvote}}(\Theta_i) = \text{sign}(\sum_i \text{sign}(L_i)) \quad (6.3) \]

Inspired in the recursive Bayesian updating approach, the joint probability of the class being a pedestrian (\(q_1\)), is computed as

\[ P(q_1|\Theta_1, \ldots, \Theta_{nc}) = \frac{P(q_1)P(\Theta_1|q_1)P(\Theta_2|q_1 \ldots \Theta_{nc})}{P(\Theta_1, \ldots, \Theta_{nc})} \quad (6.4) \]

Assuming classifiers’ independence, equation (6.4) becomes

\[ P(q_1|\Theta_1, \ldots, \Theta_{nc}) = \frac{P(q_1)\prod_{i=1}^{nc} P(\Theta_i|q_1)}{P(\Theta_1, \ldots, \Theta_{nc})} \quad (6.5) \]

Equations (6.5) depends on the prior information \(P(q_1)\), about the actual detected object being a pedestrian, which can be based on a predefined model, or it can be estimated during a tracking process. Under the strong assumption of \(P(q_1) = P(q_2)\), the Naive-product rule becomes

\[ F_{\text{Nprod}} = \frac{\prod_{i=1}^{nc} \mathcal{L}_i}{\prod_{i=1}^{nc} \mathcal{L}_i + \prod_{i=1}^{nc} (1 - \mathcal{L}_i)} \quad (6.6) \]

### 6.2.3 Trainable fusion rules

A trainable fusion rule [Kuncheva, 2004], [Ludwig et al., 2009] is, essentially, a classifier (hereafter called fusion-classifier) that receives the outputs from single-classifiers and outputs a confidence score, or a decision, concerning the object class. In this work, five trainable fusion rules were tested, each one corresponding to one of the five single classifiers described above. Both single classifiers and fusion-classifiers are trained in conjunction using the same examples of the training set however, the single classifiers have to be trained in advance of the fusion-classifiers. This is necessary because the fusion-classifiers are trained using a likelihood training set \(\mathcal{U}_{\text{train}}\) generated from the outputs of the single classifiers trained ahead. The Algorithm 7 details the training process of the fusion rules.
Algorithm 7 Training process of the fusion rules.

**Input:** $D$: training set.  
$nc$: number of single classifiers.  
$nf$: number of fusion methods ($nf = 5$).  
$\Theta_k$: set of single classifier models; ($k = 1 \cdots nc$).

**Output:** set of fusion-classifiers model $\{F_k\}; (k = 1 \cdots nf)$.

1. $U_{train} \leftarrow$ empty matrix.  
2. Train the single classifiers, using $D$, to obtain the models $\Theta_k$.  
3. for $k=1:nc$ do  
   4. Employ $\Theta_k$ in $D$ to obtain the likelihood set $L_{train}^k$;  
   5. $U_{train} \leftarrow [U_{train} \cup L_{train}^k]$: concatenate the $k^{th}$ likelihood set into the likelihood training set;  
5. end for  
7. for $k=1:nf$ do  
8. Use $U_{train}$ to train the $k^{th}$ fusion method $F_k$.  
9. end for

### 6.2.4 Multistage method

The proposed multistage method has a training strategy decomposed in two phases, bagging and boosting, applied separately in the feature subsets $D^{\text{laser}}$ and $D^{\text{vision}}$, where $D_{\text{train}} = \{D^{\text{laser}}, D^{\text{vision}}\}$. One of the contributions is the proposed bagging procedure, or bootstrap aggregation, where a subset $D_{\text{bagg}}$ is selected from $D_{\text{train}}$, the bagging set, for reusing in the training process in order to improve the final classification by means of an ensemble of classifiers (the boosting process). In the bagging procedure, the adopted partitioning criteria is based on the response of a base-linear single classifier, which is ‘boosted’ by a second non-linear single classifier trained in the subset $D_{\text{bagg}}$. Basically, $D_{\text{bagg}}$ contains all the samples (true positives TP, true negatives TN, false positives FP, and false negatives FN) within a margin defined in the first stage. Figure 6.1 illustrates this procedure with a hypothetical bi-dimensional example.

The proposed strategy for combining the classifiers has the following steps:

1. two component classifiers are considered in the training process: a base-linear classifier (trained in $D_{\text{train}}$), and a non-linear, and more robust, classifier (trained in $D_{\text{bagg}}$);

2. the subset $D_{\text{bagg}}$ is formed re-sampling $D_{\text{train}}$ with $N_s$ samples closer to the decision boundary; the criteria adopted to select the $N_s$-th part of $D_{\text{train}}$ is
6.2. COMBINING CLASSIFIERS

**Figure 6.1:** Illustrative example of the bagging process: $D_{bagg}$ is obtained considering the samples inside the margin defined by a base-linear classifier.

- based on the decision response $L_i$ of the base-linear classifier;
- the subset ($D_{bagg} \subset D_{train}$) is used as training set for the second, and non-linear, component classifier;
- the proposed ensemble is formed with the above two classifiers whose parameters were learned using $D_{train}$ and $D_{bagg}$ respectively, and the final decision is achieved considering the joint decision of the classifier’s ensemble: the boosting part.

Actually, the process described above can be performed in both feature spaces, $D^{laser}$ and $D^{vision}$, separately or in conjunction; it means that the training method proposed here can be carried out in three feature sets (Algorithm 8 is used for each feature set): laser, vision, and combined feature set.

**Figure 6.2** illustrates the proposed cascade ensemble. This classifier composition can be used jointly, in both stages, or in a separate way. Considering the laser-stage, the final classification decision is defined jointly by $fc_1$ and $fc_2$, trained with $D^{laser}_{train}$ and $D^{laser}_{bagg}$ respectively, where all the samples, within the decision margin of the first classifier, are inputs for the second classifier $fc_2$. The same functional principle is
Algorithm 8 Training process of the cascade ensemble.

**Input:** $D_{train}$: training dataset

- $N_s$: number of selected samples
- $n$: number of classifiers ($n = 2$)
- $f_c$: Classifiers ($f_{c1}$: linear; $f_{c2}$: non-linear)

1: $D_{bagg} \leftarrow$ empty set;
2: train $f_{c1}$ using $D_{train}$ to obtain the output scores $L_1$;
3: based on $L_1$, select $N_s$ samples of $D_{train}$ within the decision-margin defined by $f_{c1}$ to compose $D_{bagg}$;
4: train $f_{c2}$ using $D_{bagg}$;

**Output:** set of trained model $\{f_c\}; (i = 1 \cdots n)$

---

**Figure 6.2:** Functional diagram of the multistage ensemble.

valid for the vision-stage, and also when both laser and image features are used.

### 6.2.5 Centralized fusion architecture

In the centralized fusion architecture, the sensor fusion is carried out at the feature level \(i.e.,\) the LIDAR and the vision-based features are combined in a common space (see Fig. 6.3). Therefore, the classification module can use all the available features to assess the object class. However, in this type of fusion architecture, the training set composition demands an extra effort since the labeled examples annotated in the image frame should have a unique correspondence in the laser scan.
6.2. COMBINING CLASSIFIERS

6.2.6 Decentralized fusion architecture

In the decentralized fusion architecture, each classifier ensemble is supposed to be specialized in a part of the feature set. Consequently, one classification module can be devoted to the LIDAR feature space and other module can operate, independently or not, on the vision-based feature space (see Fig. 6.4). Usually, classifiers fusion occurs at the combination level, where the confidence score or the likelihood value from each classifier ensemble can be fused using nontrainable rules, or trainable fusion methods, or cascade schemes. Although both the centralized and decentralized architectures can use information from the feature sets separately or in conjunction, the decentralized architecture is more attractive in the sense that it is possible to approach all techniques and fusion methods in a more flexible way. Moreover, in the decentralized scheme, it is possible to model the classifiers using independent training sets.

Figure 6.4: Decentralized sensor fusion architecture: the LIDAR and the vision-based features are processed separately, therefore the classification decision can be achieved per feature space or by combining the outputs from each sensor-based module.
6.2.7 Experiments

The experiments were performed to evaluate the different classifier combination strategies using the classification dataset $D_C$. The training set $D_{\text{train}}$ contains 1100 examples, where 550 are positives and 550 negatives, while the testing set $D_{\text{test}}$ has 400 positive and 1000 negative examples (see Table 3.5). The results are assessed in terms of the metrics $AUC$, $Acc$, $BER$, $TP_{10\%}$, and $AUC_{10\%}$, and also by ROC curves, using the testing set $D_{\text{test}}$, where $D_{\text{test}} = \{D_{\text{laser test}} \cup D_{\text{vision test}}\}$.

Regarding the centralized architecture, experiments with the single classifiers (LDA, NBC, GMM, SVM, and ANN) were performed using the combined feature set and the results, in the testing set, were compared with the results obtained with the same single classifiers using, separately, the LIDAR and the vision-based features. Performance results over $D_{\text{laser test}}$, $D_{\text{vision test}}$ and $D_{\text{test}}$ sets are summarized in Table 6.1\(^4\), where the best results are highlighted in bold format, and by ROC curves of Figs. 6.5(a), 6.5(b), and 6.5(c) respectively. The NBC classifier achieved the highest scores on the experiments using single classifiers. The results show the benefit of using the combined feature set $D_{\text{test}}$ compared to the performance obtained on $D_{\text{laser test}}$ and $D_{\text{vision test}}$.

![ROC plots](image)

Figure 6.5: ROCs for the single classifiers according to the feature set.

The experiments conducted with the decentralized architecture encompasses many approaches, such as: nontrainable fusion rules, trainable rules, and the multistage scheme. The experimental results obtained with the nontrainable rules are summarized in Table 6.2, and the corresponding ROCs are shown in Figs. 6.6(a), 6.6(b), and

\(^4\)The ANN classifier was trained with a different strategy than that used in previous sections. Thus, performance results reported in Table 6.1 differ from those reported in Tables 4.1 and 5.3.
6.2. COMBINING CLASSIFIERS

Table 6.1: Classification performance using single-classifiers

<table>
<thead>
<tr>
<th>Classifier</th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>TP_{10%}</th>
<th>AUC_{10%}</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.903</td>
<td>0.834</td>
<td>0.121</td>
<td>0.575</td>
<td>0.019</td>
</tr>
<tr>
<td>NBC</td>
<td>0.932</td>
<td>0.868</td>
<td>0.115</td>
<td>0.850</td>
<td>0.039</td>
</tr>
<tr>
<td>GMM</td>
<td>0.938</td>
<td>0.865</td>
<td>0.123</td>
<td>0.825</td>
<td>0.050</td>
</tr>
<tr>
<td>SVM</td>
<td>0.897</td>
<td>0.842</td>
<td>0.121</td>
<td>0.477</td>
<td>0.015</td>
</tr>
<tr>
<td>ANN</td>
<td>0.902</td>
<td>0.851</td>
<td>0.128</td>
<td>0.618</td>
<td>0.024</td>
</tr>
</tbody>
</table>

Results on the laser set $D_{laser}^{test}$

<table>
<thead>
<tr>
<th>Classifier</th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>TP_{10%}</th>
<th>AUC_{10%}</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.902</td>
<td>0.850</td>
<td>0.164</td>
<td>0.728</td>
<td>0.032</td>
</tr>
<tr>
<td>NBC</td>
<td>0.956</td>
<td>0.875</td>
<td>0.113</td>
<td>0.870</td>
<td>0.067</td>
</tr>
<tr>
<td>GMM</td>
<td>0.950</td>
<td>0.886</td>
<td>0.139</td>
<td>0.840</td>
<td>0.059</td>
</tr>
<tr>
<td>SVM</td>
<td>0.928</td>
<td>0.841</td>
<td>0.139</td>
<td>0.772</td>
<td>0.048</td>
</tr>
<tr>
<td>ANN</td>
<td>0.901</td>
<td>0.830</td>
<td>0.168</td>
<td>0.680</td>
<td>0.038</td>
</tr>
</tbody>
</table>

Results on the vision set $D_{vision}^{test}$

<table>
<thead>
<tr>
<th>Classifier</th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>TP_{10%}</th>
<th>AUC_{10%}</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.936</td>
<td>0.889</td>
<td>0.118</td>
<td>0.850</td>
<td>0.043</td>
</tr>
<tr>
<td>NBC</td>
<td>0.973</td>
<td>0.927</td>
<td>0.073</td>
<td>0.963</td>
<td>0.075</td>
</tr>
<tr>
<td>GMM</td>
<td>0.973</td>
<td>0.908</td>
<td>0.135</td>
<td>0.988</td>
<td>0.073</td>
</tr>
<tr>
<td>SVM</td>
<td>0.967</td>
<td>0.914</td>
<td>0.076</td>
<td>0.938</td>
<td>0.071</td>
</tr>
<tr>
<td>ANN</td>
<td>0.962</td>
<td>0.899</td>
<td>0.099</td>
<td>0.902</td>
<td>0.067</td>
</tr>
</tbody>
</table>

Results on the combined set $D_{test}$

6.6(c). The majority vote and the average rule achieved good results with LIDAR and vision features; on the other hand, the Nprod-rule had the best scores on the combined feature set.

Table 6.2: Classification performance using non trainable fusion rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>TP_{10%}</th>
<th>AUC_{10%}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average-rule</td>
<td>0.925</td>
<td>0.864</td>
<td>0.111</td>
<td>0.785</td>
<td>0.033</td>
</tr>
<tr>
<td>Max-rule</td>
<td>0.897</td>
<td>0.790</td>
<td>0.148</td>
<td>0.575</td>
<td>0.021</td>
</tr>
<tr>
<td>Nprod-rule</td>
<td>0.929</td>
<td>0.864</td>
<td>0.112</td>
<td>0.775</td>
<td>0.041</td>
</tr>
<tr>
<td>Mvote-rule</td>
<td>n/a</td>
<td>0.866</td>
<td>0.102</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Results on the laser set $D_{laser}^{test}$

<table>
<thead>
<tr>
<th>Rule</th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>TP_{10%}</th>
<th>AUC_{10%}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average-rule</td>
<td>0.967</td>
<td>0.923</td>
<td>0.088</td>
<td>0.917</td>
<td>0.076</td>
</tr>
<tr>
<td>Max-rule</td>
<td>0.911</td>
<td>0.739</td>
<td>0.187</td>
<td>0.723</td>
<td>0.036</td>
</tr>
<tr>
<td>Nprod-rule</td>
<td>0.964</td>
<td>0.911</td>
<td>0.100</td>
<td>0.912</td>
<td>0.074</td>
</tr>
<tr>
<td>Mvote-rule</td>
<td>n/a</td>
<td>0.891</td>
<td>0.016</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Results on the vision set $D_{vision}^{test}$

<table>
<thead>
<tr>
<th>Rule</th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>TP_{10%}</th>
<th>AUC_{10%}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average-rule</td>
<td>0.984</td>
<td>0.956</td>
<td>0.047</td>
<td>0.983</td>
<td>0.084</td>
</tr>
<tr>
<td>Max-rule</td>
<td>0.965</td>
<td>0.869</td>
<td>0.093</td>
<td>0.938</td>
<td>0.067</td>
</tr>
<tr>
<td>Nprod-rule</td>
<td>0.984</td>
<td>0.960</td>
<td>0.044</td>
<td>0.998</td>
<td>0.085</td>
</tr>
<tr>
<td>Mvote-rule</td>
<td>n/a</td>
<td>0.929</td>
<td>0.068</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Results on the combined set $D_{test}$
Figure 6.6: ROCs for the nontrainable fusion rules according to the feature set.

Table 6.3: Classification performance using trainable fusion rules

<table>
<thead>
<tr>
<th></th>
<th>AUC</th>
<th>Acc</th>
<th>BER</th>
<th>TP_{10%}</th>
<th>AUC_{10%}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Results on the laser set $D_{test}^{laser}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{F}$-LDA</td>
<td>0.937</td>
<td>0.850</td>
<td>0.124</td>
<td>0.805</td>
<td><strong>0.051</strong></td>
</tr>
<tr>
<td>$\mathcal{F}$-NBC</td>
<td>0.933</td>
<td>0.876</td>
<td>0.107</td>
<td>0.802</td>
<td>0.041</td>
</tr>
<tr>
<td>$\mathcal{F}$-GMM</td>
<td>0.932</td>
<td><strong>0.885</strong></td>
<td>0.108</td>
<td><strong>0.820</strong></td>
<td>0.047</td>
</tr>
<tr>
<td>$\mathcal{F}$-SVM</td>
<td><strong>0.938</strong></td>
<td>0.879</td>
<td><strong>0.105</strong></td>
<td>0.820</td>
<td>0.047</td>
</tr>
<tr>
<td>$\mathcal{F}$-ANN</td>
<td>0.935</td>
<td>0.849</td>
<td>0.129</td>
<td>0.780</td>
<td>0.050</td>
</tr>
<tr>
<td>Results on the vision set $D_{test}^{vision}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{F}$-LDA</td>
<td>0.959</td>
<td>0.906</td>
<td>0.116</td>
<td>0.892</td>
<td>0.069</td>
</tr>
<tr>
<td>$\mathcal{F}$-NBC</td>
<td><strong>0.969</strong></td>
<td><strong>0.925</strong></td>
<td><strong>0.091</strong></td>
<td><strong>0.945</strong></td>
<td><strong>0.077</strong></td>
</tr>
<tr>
<td>$\mathcal{F}$-GMM</td>
<td>0.960</td>
<td>0.904</td>
<td>0.119</td>
<td>0.892</td>
<td>0.068</td>
</tr>
<tr>
<td>$\mathcal{F}$-SVM</td>
<td><strong>0.969</strong></td>
<td>0.920</td>
<td>0.097</td>
<td>0.915</td>
<td><strong>0.077</strong></td>
</tr>
<tr>
<td>$\mathcal{F}$-ANN</td>
<td>0.964</td>
<td>0.914</td>
<td><strong>0.091</strong></td>
<td>0.920</td>
<td>0.074</td>
</tr>
<tr>
<td>Results on the combined set $D_{test}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\mathcal{F}$-LDA</td>
<td>0.974</td>
<td>0.921</td>
<td>0.113</td>
<td>0.983</td>
<td>0.078</td>
</tr>
<tr>
<td>$\mathcal{F}$-NBC</td>
<td>0.977</td>
<td>0.939</td>
<td>0.062</td>
<td><strong>0.990</strong></td>
<td>0.076</td>
</tr>
<tr>
<td>$\mathcal{F}$-GMM</td>
<td>0.973</td>
<td>0.934</td>
<td>0.071</td>
<td>0.980</td>
<td>0.075</td>
</tr>
<tr>
<td>$\mathcal{F}$-SVM</td>
<td><strong>0.983</strong></td>
<td>0.949</td>
<td>0.057</td>
<td>0.983</td>
<td>0.084</td>
</tr>
<tr>
<td>$\mathcal{F}$-ANN</td>
<td><strong>0.983</strong></td>
<td><strong>0.958</strong></td>
<td><strong>0.051</strong></td>
<td>0.980</td>
<td><strong>0.085</strong></td>
</tr>
</tbody>
</table>

Experimental results for the ensemble using trainable fusion rules are given in Table 6.3, and the corresponding ROCs are shown in Fig. 6.7(a) for $D_{test}^{laser}$, in Fig. 6.7(b) for $D_{test}^{vision}$, and in Fig. 6.7(c) for the complete feature set. Regarding the multistage ensemble, Table 6.4 and Figs. 6.8(a), 6.8(b), and 6.8(c) give the results for this method.

In summary, the results from the experiments on multiclassifier ensembles, compared with single classifiers, demonstrate that the nontrainable and the trainable
rules outperform the single classifiers in the vision and in the combined feature sets, while the performance in the LIDAR feature set is more balanced. Additionally, it can be observed that the nontrainable rules achieved better performance scores than the trainable methods, in particular the $F_{	ext{Mvote}}$ and the $F_{	ext{Average}}$. Despite the increase in classification performance obtained with the multiclassifier fusion rules, it is important to notice that, in general, the ensemble methods also increase the complexity of the solution and demands more computing time in the algorithms. Regarding the multistage ensemble, many configurations were experimented, in K-cross validation.
sessions, for deciding which arrangement to use in the ensemble module. Although multiple variations of the multistage ensemble have been trained (changing the classifier method and/or the feature space), some of them with remarkable results in the training dataset, the final selected ensemble did not achieve the expected results over the testing datasets.

6.3 Sensor fusion for vehicle velocity estimation

described by a direct discrete-time linear dynamic system:

Stochastic measurement models, using DGPS and encoder data, were used for the estimation of the vehicle speed. The main objective is to fuse information from an encoder to compensate (decrease) errors and uncertainties of the GPS system. It is important to note that the main state variable of concern is the linear velocity, in Cartesian coordinates, of the vehicle along a GPS-based ‘navigation’ plane which is defined in ENU coordinate system [Drake, 2002] (details about the conversion of GPS to ENU coordinates are given in Chapter 7). Two model will be considered for the kinematic of the vehicle\(^5\): (1) White noise acceleration, or second-order model, and (2) Wiener process acceleration, or third-order model [Bar-Shalom and Li, 2001]. Kinematic models for the vehicle, and the process and observation models, for both sensors, are given in the sequel.

Incremental encoder

The vehicle (ISRobotCar) has an incremental encoder mounted in a mechanical part attached to rear axis of the vehicle. Hence, the number of pulses recorded from the encoder is proportional to the rotational movement of the rear axle. Considering that the vehicle moves when the rear axle moves, the estimated increment in the position of the vehicle \(\Delta p\), in the time interval \(h\), is given by:

\[
\Delta p = \left(\frac{2\pi R_{rw}}{n_e}\right)n_p(h)
\] (6.7)

\(^5\)It is, by no means, a rigorous model for vehicle dynamics.
where \( np(h) \) is the number of counted pulses from the encoder (measured in the time interval), \( n_e \) is a constant (a manufacturer parameter) which represents the number of pulses per revolution of the encoder, and \( R_{rw} \) is the average radius of wheels, plus tires, of the rear axle. Considering that the ‘initial time’ is \( k_i = 1 \), the expression for the encoder measurements, at a given sample-time \( k \), is:

\[
z_k = \left( \frac{2\pi R_{rw}}{n_e} \right) np(k)
\]

where \( np(k) \) is the sum of the pulses from \( k_i = 1 \) to the current sample-time \( k \).

### Discrete-time linear kinematic models

The dynamic of the vehicle, defined in *direct discrete-time* [Bar-Shalom and Li, 2001], is expressed by the general linear state equation:

\[
x_k = Fx_{k-1} + \Gamma \upsilon_k
\]

where \( x_k \) is the \( n \times d \) state vector\(^6\), \( F \) is the state transition matrix, \( \Gamma \) is the noise gain, and \( \upsilon_k \) is the sequence of zero-mean white Gaussian process noise with covariance defined by \( Q_k = \Gamma \sigma^2_c \Gamma' \), where \( \sigma^2_c \) is variance of the noise \( \upsilon_k \).

Describing the measurement model by a vector difference equation with the uncertainties, somewhat unpredictable disturbances, entering the system as additive white Gaussian noise \( \omega_k \), the measurement model is given by:

\[
\hat{z}_k = Hx_k + \omega_k
\]

where \( H \) is the measurement matrix, and \( \omega_k \) is the measurement noise/error with covariance \( R_k = E[\omega_k\omega'_k] = I_{n\times n}\sigma^2_\omega \), where \( I_{n\times n} \) is the identity matrix and \( n \) is the order of the model.

In the White noise acceleration (WNA) model [Bar-Shalom and Li, 2001], it is assumed that the velocity undergoes at least slight changes \( i.e. \), the acceleration is not constant. This can be modeled by a zero-mean white noise \( \upsilon \) as: \( \ddot{x}_k = \upsilon_k \)

\(^6\)\( n \) indicates the number of state variables, and \( d \) is the number of axis/dimensions in a given reference system.
[Bar-Shalom and Li, 2001]. In this model, noise enters in system in the form of a white-noise acceleration. On the other hand, for the piecewise constant Wiener process acceleration (WPA) model [Bar-Shalom and Li, 2001], or third-order model, the derivative of the acceleration (jerk) is not constant, and its changes is modeled by a zero-mean white noise $\ddot{x}_k = \upsilon_k$ i.e., the acceleration is a Wiener process [Bar-Shalom and Li, 2001].

State-space models using the encoder

The unidimensional state vector\(^7\) for the WNA kinematic model, using encoder readings, is given by

$$x = [p^e, \dot{p}^e]'$$

where the elements of $x$ correspond to the position and linear speed of the vehicle respectively i.e., it corresponds to a second-order model. The transition and the process noise matrices, for the encoder-based WNA model, are

$$F = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} h^2/2 \\ h \end{bmatrix}$$

(6.12)

where $h$ is the sample-time interval. For the WPA model, also designated by white-noise jerk model [Bar-Shalom and Li, 2001], the state vector is

$$x = [p^e, \dot{p}^e, \ddot{p}^e]'$$

(6.13)

which corresponds to a third-order model $n = 3$. In direct discrete-time, the white process noise $\upsilon_k$ is the acceleration increment during the $h$ sampling period and it is assumed to be a zero-mean white sequence. The transition and the process noise matrices are

$$F = \begin{bmatrix} 1 & h & h^2/2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} h^2/2 \\ h \\ 1 \end{bmatrix}$$

(6.14)

The covariance matrices, in both models, were considered constant, hence $Q_k = Q$ and $R_k = R$, and the values $\sigma^2_\upsilon = 0.1$ and $\sigma^2_\omega = 0.01$ were obtained empirically. The

\(^7\)The encoder gives ‘position’ measurements in one dimension.
covariance and measurement matrices are shown in Table 6.5.

Table 6.5: Discrete-time models for single incremental encoder

<table>
<thead>
<tr>
<th>WNA or 2\textsuperscript{nd}-order model</th>
<th>WPA or 3\textsuperscript{rd}-order model</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{H} = \begin{bmatrix} 1 &amp; 0 \end{bmatrix} )</td>
<td>( \mathbf{H} = \begin{bmatrix} 1 &amp; 0 &amp; 0 \end{bmatrix} )</td>
</tr>
<tr>
<td>( \mathbf{Q} = \begin{bmatrix} \frac{1}{4} h^4 &amp; \frac{1}{2} h^3 \ \frac{1}{2} h^3 &amp; h^2 \end{bmatrix} \sigma_v^2 )</td>
<td>( \mathbf{Q} = \begin{bmatrix} \frac{1}{4} h^4 &amp; \frac{1}{2} h^3 &amp; \frac{1}{2} h^2 \ \frac{1}{2} h^3 &amp; h^2 &amp; h \ \frac{1}{2} h^2 &amp; h &amp; 1 \end{bmatrix} \sigma_v^2 )</td>
</tr>
<tr>
<td>( \mathbf{R} = I_{2\times2} \sigma_w^2 )</td>
<td>( \mathbf{R} = I_{3\times3} \sigma_w^2 )</td>
</tr>
</tbody>
</table>

**DGPS**

The Differential-GPS, which operates with a ground referenced station (base station), has an absolute precision measurement much more accurate than the GPS system. The base station communicates with the moving station (mounted on the vehicle) at UHF frequency. The information shared between the stations has an average cycle of 200 ms (5 Hz), which supports the RTK designation. However, GPS and also DGPS are prone to a sort of errors [Hall and Llinas, 1997], [Bar-Shalom and Li, 2001], [Caron et al., 2006] which makes its utilization in precise and reliable state estimation quite difficult. In the case discussed here, part of the location measurements suffered one of the most common problem observed in GPS, the mask effect or multi-path problem; Fig. 6.9(a) exemplify this problem, which occurred in a trajectory that is partially covered by buildings. Moreover, the lack of credibility of GPS in some parts of the trajectory occurred due to temporary lost of communication between the rover and the base station. Examples of this particular problem is shown in Fig. 6.9(b).

Inconsistencies in the DGPS data affect the performance of the stochastic filter because the measurement residual (or innovation), which depends directly on the measurements, affects the ‘updated state estimate’ stage and, consequently, the state vector estimation yielded by the filter will be also inconsistent. A solution for this problem was obtained using a data association stage, or association validation, which is based on the normalized innovation squared. Experimental results using data association, and without measurement validation, are presented and discussed in Section 6.3.1.
The DGPS system provides position measurements of the vehicle in two coordinates, \textit{i.e.}, \textit{x}-\textit{y} in local Cartesian (ENU) coordinates [Drake, 2002], expressed by:

\[
\mathbf{z}_k = [\mathbf{\bar{p}}^g, \mathbf{\bar{p}}^g]' \tag{6.15}
\]

as consequence of (6.15), the state vector has to be augmented to include a second dimension.

**State-space models using the DGPS**

For the WNA model, using DGPS position measurements in the \textit{x}-\textit{y} navigation reference frame, the state vector is

\[
\mathbf{x} = [\mathbf{p}^g, \mathbf{\dot{p}}^g, \mathbf{p}^g, \mathbf{\dot{p}}^g]' \tag{6.16}
\]

while the transition and the process noise matrices, for this second-order model, are

\[
\mathbf{F} = \begin{bmatrix}
1 & h & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & h \\
0 & 0 & 0 & 1
\end{bmatrix}, \quad \mathbf{\Gamma} = \begin{bmatrix}
h^2/2 & 0 \\
0 & h \\
0 & h^2/2 \\
0 & 0 & h
\end{bmatrix} \tag{6.17}
\]

\footnote{Section 7.4.1 provides more details regarding transformation of GPS data (defined in WGS84 coordinates) to ECEF and ENU coordinates.}
Regarding the DGPS-based WPA model, the state vector is augmented to include acceleration, thus

\[ \mathbf{x} = [p_x^g, \dot{p}_x^g, \ddot{p}_x^g, p_y^g, \dot{p}_y^g, \ddot{p}_y^g]' \]  

(6.18)

and the expressions for \( \mathbf{F} \) and \( \Gamma \) are

\[
\mathbf{F} = \begin{bmatrix}
1 & h & h^2/2 & 0 & 0 & 0 \\
0 & 1 & h & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & h & h^2/2 \\
0 & 0 & 0 & 0 & 1 & h \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad
\Gamma = \begin{bmatrix}
h^2/2 & 0 \\
0 & h \\
1 & 0 \\
0 & h^2/2 \\
0 & h \\
0 & 1
\end{bmatrix}
\]

(6.19)

The estimated speed \( \vartheta_k \) (the magnitude of the velocity vector) was obtained by the normalization of the velocity along each coordinate, hence \( \vartheta_k = \sqrt{\dot{p}_x^g k + \dot{p}_y^g k} \). The covariance and the measurement matrices, for both models, are shown in Table 6.6.

<table>
<thead>
<tr>
<th>Table 6.6: Discrete-time models for DGPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>WNA or 2\textsuperscript{nd}-order model</td>
</tr>
<tr>
<td>------------------------------------------</td>
</tr>
<tr>
<td>( \mathbf{H} = \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix} )</td>
</tr>
<tr>
<td>( \mathbf{Q} = \begin{bmatrix} \frac{1}{4}h^4 &amp; \frac{1}{2}h^3 &amp; 0 &amp; 0 \ \frac{1}{2}h^3 &amp; h^2 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; \frac{1}{4}h^4 &amp; \frac{1}{2}h^3 \ 0 &amp; 0 &amp; \frac{1}{2}h^3 &amp; h^2 \end{bmatrix} ) 0.1</td>
</tr>
<tr>
<td>( \mathbf{R} = I_{4\times4}0.01 )</td>
</tr>
</tbody>
</table>

### 6.3.1 Experiments using the Kalman Filter

This section presents experiments for the ISRobotCar speed estimation using encoder, DGPS, and their integration. The Kalman Filter (KF) was used for stochastic state estimation and filtering. In practical problems, a ‘gating’ or ‘validation’ stage is necessary to select the measurement(s) to be used in the state update, in order to keep the filter consistent. The normalized innovation squared (NIS), or *Mahalanobis*
distance, was used as statistical validation region (VR). Given a measurement innovation $\nu_k$ and its predicted covariance $S_k$, see (6.22), the NIS-based validation region $V_k$ is defined as

$$ V_k = \nu'k S^{-1}_k \nu_k. \quad (6.20) $$

The criteria adopted to discard inconsistent measurements depends on a NIS-based threshold $\gamma_d$, also known as gate, where $d$ is the dimension of $\nu$. It was assumed that $\nu$ follows a Gaussian distribution so, the gate threshold $\gamma_d$ was assigned from $\chi^2$ test table, corresponding to a determined gate probability [Bar-Shalom and Fortmann, 1988]. In summary, the KF algorithm is composed by the following stages:

**Prediction stage**

\[
\begin{align*}
\hat{x}_{k}^- &= F\hat{x}_{k-1}^- \quad \text{State prediction} \\
P_{k}^- &= FP_{k-1}F' + Q \quad \text{State prediction covariance} \\
\end{align*}
\]

(6.21)

**Validation stage**

\[
\begin{align*}
\nu_k &= [z_k - H\hat{x}_{k}^-] \quad \text{Innovation} \\
S_k &= (H_k P_{k}^- H_k' + R) \quad \text{Innovation covariance} \\
V_k &= \nu'k S^{-1}_k \nu_k \quad \text{Validation region or 'gating'}
\end{align*}
\]

(6.22)

**Correction stage**

\[
\begin{align*}
K_k &= P_{k}^- H_k' S^{-1}_k \quad \text{Kalman gain} \\
\hat{x}_k &= \hat{x}_{k}^- + K_k \nu_k \quad \text{Corrected state estimate} \\
P_k &= (I_{n \times n} - K_k H_k)P_{k}^- \quad \text{Corrected state covariance}
\end{align*}
\]

(6.23)

**Speed estimation using encoder measurements**

The results using encoder measurements, for the 2nd and 3rd-order models (see Table 6.5), are presented in Fig. 6.10 where $R_{rw}=0.2286$, $n_e=312$, and $h=0.2s$. It can be seen that for $n = 3$ (third-order model), the estimated speed reach values higher than the solution with $n = 2$, which is consequence of the additive noise on the acceleration.

In the Kalman filter with encoder (designated by KF$^e$), implemented to obtain the
estimations shown in Fig. 6.10, the validation stage (6.22) did not affect the results. On the other hand, (6.22) can be of major importance when DGPS is used. This is discussed, and demonstrated, in the next section.

**Speed estimation using DGPS measurements**

The results shown in Fig. 6.11 were obtained using a KF without the validation stage. Inconsistencies on the DGPS measurements affect the state estimation, which is evidenced by some abrupt changes in the speed, e.g., for \( \vartheta_k > 30 \text{ Km/h} \). These ‘outliers’ present on the speed occur when the system loses the differential (RTK) signal due to communication interruptions between the base and the rover stations.

To solve this problem, the validation gate (6.22) was used to select the measurement to be used in the state correction/update stage of the KF. Measurements which do not ‘fall’ inside the validation gate are rejected and the state vector is updated without the correction stage, i.e., the KF gain and the corresponding innovation do not affect the state updating, that is:

\[
\hat{x}_k = \hat{x}_k^-
\]

\[
P_k = P_k^-
\] (6.24)
Figure 6.11: Estimated speed $\vartheta = \sqrt{\dot{p_x}^2 + \dot{p_y}^2}$, without the validation stage, using DGPS measurements. The results for the models with $n = 2$ and $n = 3$ are shown in dashed-blue and black respectively.

The results shown in Fig. 6.12 were obtained with the validation gate incorporated in the KF (designated by $\text{KF}^g$). As expected, the solution using KF and validation regions, denoted by $\text{(KF|VR)}$ in the legend of Fig. 6.12, smoothed the results.

Figure 6.12: DGPS-based estimated speed $\vartheta$, for both models, using the $\text{KF}^g$ algorithm with validation region (VR).
6.3. SENSOR FUSION FOR VEHICLE VELOCITY ESTIMATION

Speed estimation using DGPS and encoder

Finally, a practical and efficient solution is proposed for vehicle’s speed estimation using the Kalman filters $\text{KF}^e$ and $\text{KF}^g$. In this proposed solution, both KF algorithms work separately and in simultaneous way thus, they share the same time index $k$. When a DGPS measurement is rejected by the validation stage of $\text{KF}^g$, the $\text{KF}^e$ filter plays the role of speed predictor in (6.21). Therefore, when a DGPS measurement is rejected by $\text{KF}^g$, the components of the state vector and the state covariance are updated using (6.21), except the estimated velocity which is updated according to (for the WNA model):

\[
\begin{align*}
\dot{\hat{p}}_x^e &= \hat{p}_k^e / \eta \\ 
\dot{\hat{p}}_y^e &= \hat{p}_k^e / \eta 
\end{align*}
\]

where $\eta = \sqrt{2}$. Figure 6.13 illustrates, for $n = 2$, the resulting solution for speed estimation in three circumstances: (1) using $\text{KF}^e$ (results in black), (2) using $\text{KF}^g$ (results shown in grey), and (3) $\text{KF}^g$ and $\text{KF}^e$ fusion (in blue). It can be seen that the third solution demonstrates the most promising results.

Figure 6.13: Speed $\dot{\vartheta}$ estimated using DGPS and encoder measurements. When a DGPS measurement does not pass the data association stage, $\dot{\vartheta}$ is estimated according to expression (6.25).
This page is intentionally left blank
Chapter 7

Pedestrian detection using LIDAR and vision

The previous chapters addressed mainly the problem of pedestrian classification, serving as the basis to deal with the problem of pedestrian detection, which is the focus of this chapter. While pedestrian classification deals with labeled segments (in the laser space) and labeled bounding-boxes (in the image plane), the domain of pedestrian detection consists in raw laser-scans and full image frames. Thus, detection is much more demanding than classification because the position and the scale (or distance) of the pedestrians in the sensor FOV are not known.

The pedestrian detection system, represented by the diagram of the Fig. 7.1, is evaluated concerning three aspects: (1) the type of features/descriptors; (2) the classification and decision methods; (3) with/without context-based information. The feature vectors used in the classifiers are HOG-COV, and Haar-like features. The single classifiers LDA, SVM and ANN, and negative-rejection cascades are the classification methods chosen for the experiments. Finally, intrinsic and extrinsic attributes of the detection windows are used in the context-based approach. The intrinsic attribute is the estimated speed of the object, and the extrinsic one refers to the object position in a contextual map defined in GPS coordinates.
7.1 Pedestrian detection without contextual information

One of the key problems in monocular image-based pedestrian detection is the huge amount of negatives (potential false alarms) in contrast with the number of positives, what demands vast processing time and a high confidence detector. The adopted approach to deal with this problem was to use the sensor fusion architecture, given in Fig. 7.1, which generates, from the LIDAR segmentation module, a set of ROI in the image frame. Inside each ROI, a classification method is used in the form of a multiscale sliding-window which is shifted in position and size for searching pedestrian evidence. Thus, the proposed solution has the LIDAR-based system acting as primary object detection, and the decision making depends on the vision-based system. Concisely, the main processing modules represented in Fig. 7.1 are:

1. LIDAR-data processing: comprehends a set of pertinent data processing tasks, necessary to decrease complexity and processing time of subsequent stages, such as: filtering-out isolated/spurious range-points, discarding measurements that occur out a predefined FOV, and data alignment.

2. LIDAR Segmentation: this module outputs a set of segments obtained by a range-data segmentation method.

3. Laser to image transformation: defined as the set of rigid coordinate transformations necessary to project the segments into the image plane. This module outputs the set of ROIs.
4. Decision module: involves a classification method used to detect, using the sliding-window technique, potential pedestrians inside the ROIs.

Since the first three topics were introduced and discussed in previous chapters, the decision module (highlighted in Fig. 7.1 and detailed in Fig. 7.2) will be emphasized in this chapter. Basically, the decision module is characterized by a classification method, in the form of a detection window, followed by a non-maximum suppression filtering and a decision making stage. A classification method is used in the form of a multiscale detection window in order to scan the ROIs in searching for pedestrian evidence. Therefore a detection window is shifted by horizontal and vertical step factors, and the scale is estimated using the depth information provided by the ROIs, that is, LIDAR measurements are also used for scale estimation. This approach decreases the computational processing time, restricting the zones of interest in the image to a dozen of ROIs at most, and reducing the false positives. For instance, the number of detectors generated by this LIDAR-based approach is in average thousands of times lower than the usual full-scanning image approach. Moreover the detection rate can be improved since range data is directly used to define the detection scales. To summarize, the LIDAR-based ROI generating method has two major advantages: (1) decreasing the time-processing and complexity; (2) enabling range (scale) information.

The multiscale sliding window approach, which is part of the decision module shown in Fig. 7.2, generates a set of detection windows $DW$ per ROI. Each detector $DW_i$ is defined by an area $A$, in pixels, a confidence score (classified as a positive), and a set of attributes $s_i$. An inevitable problem that arises in multiscale sliding window approach is the occurrence of multiple detection windows in the same ‘neighborhood’ area in the ROI. To solve this problem, a non-maximum suppression method [Enzweiler and Gavrila, 2009], [Dollar et al., 2009] is used to discard multiple-detector occurrence around close/similar locations. The ratio $\gamma$ between the intersection and the union
area of overlapping detection windows is calculated and, for $\gamma > 0.6$ the detector with the greatest confidence score is retained and the remaining are discarded.

### 7.2 Criterion used for performance evaluation

Per-frame evaluation was chosen as the preferred methodology for the evaluation of the pedestrian detection system, as in [Dollar et al., 2009], [Enzweiler and Gavrila, 2009] and [Enzweiler and Gavrila, 2011]. Therefore, the basis for the system performance evaluation relies on comparing all the detection windows in a given frame with the set of ground-truth $G$ in the same frame. An element of $G$ is defined by an area $A^G_j$ and an associated label: class-0 (occluded or under a minimum scale) or class-1 (entire body pedestrian).

The decision process necessary to establish the correspondence among the detection windows and the ground-truth uses a similar ratio as discussed before i.e., the area $A_i$ of a given detector has a match with a ground-truth $A^G_j$ if $\Upsilon(i,j)n$ exceeds the threshold $A_{thr}$:

$$\Upsilon(i,j) = \frac{\text{Area}(A^G_j \cap A_i)}{\text{Area}(A^G_j \cup A_i)} > A_{thr}$$

(7.1)

where $A_{thr} = 0.25$ was chosen based in [Enzweiler and Gavrila, 2009]; although quite arbitrary\(^1\), this value is reasonable due to the average dimensions of the elements in $G$. Each detection event in $DW$ should be matched at most once with an element of $G$. Unmatched elements of $DW$ count as false positives, and unmatched elements of $G$, labeled as class-1, count as missing. On the other hand, elements of $G$ labeled as class-0 are ignored, that is, these bounding-boxes do not need to be matched; however, when a match occurs it is not counted as $FP$. Figure 7.3 illustrates laser-segments, left part of the figure, and their projections onto the image plane, represented by dashed regions. An element of $G$ pertaining to the class-0 is highlighted by the blue rectangle, while an example of the class-1 is marked in red.

\(^1\)Other possible value for $A_{thr}$ is 0.5 as in [Dollar et al., 2009]
7.3. CLASSIFICATION METHODS USED IN A DECISION MODULE

Finding a suitable balance between complexity and generalization is a key challenge faced on classifier selection. One of the criteria behind the classifiers choice was towards an allowable processing time\(^2\) in the training and testing of the algorithms. Furthermore, the classification methods should achieve, as much as possible, high detection rate and low false positive rate on the test set.

To reduce CPU-time and memory resources during the training of the classification methods, the proposed SVM-based resampling method presented in Section 5.3 was used to reduce the cardinality of the negative training samples. Thereby, instead of dealing with millions of negatives, a much more reduced number of negative samples, 151528 to be precise, was considered in the training set composition. Although reduced in number, the selected samples are representative in the sense that they are elements of the margin, i.e., support vectors (see Section 5.3). Regarding the classification methods, three single classifiers (LDA, SVM, ANN) and four cascades were used in the decision module. The cascade methods implemented for the experiments were: SVM-Cascade, SRM-Cascade, S-AdaBoost and M-AdaBoost cascades, and Haar-AdaBoost cascade. All these methods were compared in terms of performance metrics (\(TPR, FPR, Error\)) and ROC curves using the test dataset.

\(^2\)Several weeks of CPU time were necessary for training and testing the algorithms, using implementations in C/C++ and Matlab.
Figure 7.4: Block diagram illustrating the main processing modules comprised in the pedestrian detection system when contextual information is incorporated into the system.

7.4 Context-based information

Context represents input-dependent information, other than from the object pattern itself, used to improve the detection performance [Duda. et al., 2001]. In this work, context refers to the estimated speed and the position of the object in the scenario, as opposed to the object pattern itself, which refers to a given model learned from the set of ‘local’ features e.g., HOG and COV. The contextual information, incorporated in the pedestrian detection system, is based both on prior knowledge of the object position in a semantic map of the environment, and on the object’s speed derived from the LIDAR data. In short, context in this thesis consists of (i) current location of the detected object in the semantic map, and (ii) the estimated speed of the object. Figure 7.4 shows the diagram of the pedestrian detection system integrating context information. Note that a tracking stage was incorporated in the LIDAR-based system to estimate the speed of the objects.

In the decision module the contextual information is processed in the form of prior probabilities according to a maximum a-posteriori (MAP) decision framework. For this reason, the contextual information has to be modeled by a probabilistic distribution in order to obtain the prior probabilities. Moreover, the response of the
classification method, used in the decision module, has to be posed in probabilistic terms; more specifically, the response of the classification method defines the likelihood, or class-conditional probability, used in the MAP decision.

7.4.1 Context-based system using object position in semantic map

In the context-based system, the position of the objects, which have been detected by the LIDAR-based system, are defined in GPS coordinates according to a semantic map of the scenario where the LIPD dataset was recorded. Context-based prior probabilities are used as function of the position, actually presence, of the objects in specific regions of the map. This approach is explained in the sequel.

A semantic map of the roads traveled by the ISRobotCar was built with the aid of satellite imagery from the Google Earth©, as illustrated in Fig. 7.5. A set of regions in the map was selected and identified as regions with a high potential of pedestrian occurrence. The idea was to assign a high confidence score (or prior knowledge) to the zones in the map that are more likely to contain pedestrians, hereafter called incidence regions/zones and, on the other hand, to assign a low prior probability to the remaining zones of the map. The incidence regions defined in the map are: the crosswalks, regions near restaurants and cafeterias, and the zone which covers the roundabout in front of the main secretariat building. Some examples of incidence zones are shown in Fig. 7.5(b) and Fig. 7.5(c).

The semantic map, and consequently the incidence regions, are defined in GPS coordinates, more specifically, GPS data is defined in terms of latitude, longitude and altitude ($lla$) in the World Geodetic System 1984 (WGS84). However, the objects (segments) detected in the LIDAR-based system have coordinates defined by the adopted Cartesian system described in Section 4.1. Thus, it is necessary to establish a correspondence between the GPS coordinate system and the ‘local’ coordinate system of the laser, or vice versa.

A solution for obtaining the objects coordinates, in the LIDAR reference system, with respect to a point in the semantic map is to convert the GPS coordinates into local navigation coordinates. This ‘local’ navigation system is determined by the east, north and up (ENU) reference system [Drake, 2002]. Therefore, it is necessary
to transform the regions of the map, defined in WGS84 coordinates, and the objects coordinates to the ENU reference system in order to determine if a detected object, in a given frame, is inside or not an incidence region.

It is important to note that first, the $xy$ position of a given detected object, in LIDAR coordinate system, should be converted to a reference point in the vehicle (aligned with the GPS rover-station), and then converted to ENU coordinate system. In summary, the transformation of any point $P_{GPS}$ in GPS coordinates to ENU is a three stage process:

1. Define a reference point in the map, here denoted by $P_{ref}$, in $lla$ coordinates;
2. Express $P_{GPS}$, defined in $lla$ coordinates, in Earth Centered Earth Fixed (ECEF) coordinates. In this work, the $lla2ecef$ Matlab function was used for this purpose; in Matlab notation: $P_{ECEF} = lla2ecef(P_{GPS}, 'WGS84')$;
3. Convert $P_{ECEF}$ to the ENU coordinate. Denoting by $P_{ENU}$ a point in the navigation coordinates, in Matlab notation we have: $P_{ENU} = ecef2lv(P_{ECEF}, P_{ref}, \text{ellipsoid})$, where $\text{ellipsoid}$ represents the Ellipsoid fitted around the Earth globe. Using the Matlab $almanac$ function: $\text{ellipsoid} = almanac('earth', 'ellipsoid', 'kilometers', 'WGS84')$.

Designating by $P$ the prior probability of pedestrian occurrence in an incidence zone, experiments were performed considering: (1) mutual exclusiveness and exhaustiveness of the events, i.e., if an object is located in a non-incidence zone its prior is

\[ P = \text{whatever the point represents the position of the rover-station or one of the corners of the polygon that defines an incidence region.} \]
equal to $1 - P$, and (2) the prior for both classes on non-incidence zones are uniform and, on the other hand, the priors for objects on incidence zones are mutually exclusive and exhaustive. The latter is motivated by the fact that positive events \textit{i.e.}, pedestrians, should have priority in realistic protection systems.

### 7.4.2 Context-based system using object speed

Pedestrians speed, represented by the random variable $x$, was estimated using the LIDAR-based measurement model presented in Section 4.5. The empirical distributions $p(x|\omega_i)$ that describe the estimated speed of PED and nPED are shown in Fig. 7.6. The normalized values of those distributions are used in the form of prior probabilities for context-based pedestrian detection, as discussed in the previous section. Therefore, the prior probabilities are obtained applying the normalization, \textit{e.g.}, $P(\omega_1) = \frac{p(x|\omega_1)}{p(x|\omega_1) + p(x|\omega_0)}$.

### 7.5 Experiments

The objective of the experiments described in this section is to demonstrate the detection performance as function of (i) the classification methods, (ii) the type of
image-based features \textit{i.e.}, HOG, COV and Haar, and (iii) when contextual information is available. The experimental results reported in the next sections are function of modifications in the decision module, which is represented by the diagram shown in Fig. 7.2. More specifically, modifications on the classifiers and on the features \textit{i.e.}, cases (i) and (ii), affect only the first functional block of the decision module (see Fig. 7.2), while the experiments with contextual information, case (iii), have influence in the decision making stage. On the other hand, all the parameters and variables of the methods and techniques chosen for the LIDAR-based system (\textit{e.g.}, segmentation, tracking, data association) and any other processing stage\footnote{excepting, of course, the classification and decision making of the decision module}, are rigorously the same for all the three mentioned cases. In other words: the results reported in the sequel depend, exclusively, on the classification methods, and the corresponding features, and on the prior probabilities used in the decision making.

### 7.5.1 Experiments with single classifiers

Three single classifiers, namely LDA, SVM and ANN, were employed for pedestrian detection. LDA and SVM were trained with $D_{\text{train}}$ directly, however the ANN was trained with a balanced set, selected from $D_{\text{train}}$, to avoid overfitting in the majority class; notice that the number of negatives has approximately 29 times more samples than the positive part (see Table 3.6). Table 7.1 summarizes the results obtained with the single classifiers used as classification method in the decision module.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>$TPR$</th>
<th>$FPR$</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.92917</td>
<td>0.01523</td>
<td>0.01524</td>
</tr>
<tr>
<td>SVM</td>
<td>0.87858</td>
<td>0.00808</td>
<td>0.00810</td>
</tr>
<tr>
<td>ANN</td>
<td>0.81619</td>
<td>0.00766</td>
<td>0.00770</td>
</tr>
</tbody>
</table>

Additionally, four nontrainable fusion rules were used, for single classifiers fusion, in the decision module: average $F_{\text{Average}}$, maximum $F_{\text{Max}}$, minimum $F_{\text{Min}}$, and majority vote $F_{\text{Mvote}}$. The experimental results obtained with these nontrainable fusion rules are shown in Table 7.2.
7.5. EXPERIMENTS

Table 7.2: Nontrainable fusion rules performance.

<table>
<thead>
<tr>
<th>$F$</th>
<th>TPR</th>
<th>FPR</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{\text{Avage}}$</td>
<td>0.92074</td>
<td>0.01472</td>
<td>0.01473</td>
</tr>
<tr>
<td>$F_{\text{Max}}$</td>
<td>0.94435</td>
<td>0.01643</td>
<td>0.01644</td>
</tr>
<tr>
<td>$F_{\text{Min}}$</td>
<td>0.66948</td>
<td>0.00250</td>
<td>0.00256</td>
</tr>
<tr>
<td>$F_{\text{Mvote}}$</td>
<td>0.88364</td>
<td>0.00889</td>
<td>0.00891</td>
</tr>
</tbody>
</table>

7.5.2 Experiments with the Haar-AdaBoost cascade

The well-known negative rejection cascade method proposed by [Viola and Jones, 2001], using Haar-like features, in its openCV version, was used in the experiments with the LIPD detection dataset. A total of 11 stages were trained\(^5\), under the following conditions: 0.99 true positive rate, 0.5 false positive rate, Discrete AdaBoost algorithm\(^6\), complete set of Haar-templates. Results of experiments with the Haar-AdaBoost rejection cascade are shown in Table 7.3, where the highest values of TPR and FPR are highlighted in bold. Moreover, Fig. 7.7 illustrates the evolution of the true positive rate (a), and of the false positive rate (b) as function of the number of stages.

Table 7.3: Detection performance using Haar-like AdaBoost cascade.

<table>
<thead>
<tr>
<th>$F$</th>
<th>TPR</th>
<th>FPR</th>
<th>Error</th>
<th>Nc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haar-AdaBoost</td>
<td>0.88461</td>
<td>0.01489</td>
<td>0.00769</td>
<td>8</td>
</tr>
<tr>
<td>Haar-AdaBoost</td>
<td>0.85329</td>
<td>0.00766</td>
<td>0.00769</td>
<td>11</td>
</tr>
</tbody>
</table>

7.5.3 Experiments with the S-AdaBoost and the M-AdaBoost using HOG-COV features

Experiments in the training set with the S-AdaBoost and the M-AdaBoost, using HOG-COV descriptors, were described in Section 5.4.2. In this section, the experiments are performed in the test set, where the results in terms of TPR and FPR for both methods are shown in Fig. 7.8.

In black it was plotted the results for the S-AdaBoost, while the M-AdaBoost has

---

\(^5\) The cascade training process, alone, demanded more than 15 working days.

\(^6\) Other variant AdaBoost algorithms are: Real, Gentle, Modest; see [Vezhevets, 2008], [Freund and Schapire, 1996] for a review.
Figure 7.7: Results, on the test set, regarding the true positive rate (a), and the false positive rate (b), for the Haar-AdaBoost rejection cascade as function of the number of stages.

the testing results presented in blue. At a first sight, one can argue that the S-AdaBoost (black curves) presents better results than the modified variant. However, experiments on the test set serve as evidence of the expected behavior of the cascade in an unseen scenario and can not be used for model selection. Thus, it is almost impossible to predict that the S-AdaBoost, and with 5 stages, would be the best model for the test (unseen) set. On the other hand, the M-AdaBoost tends to behave as the usual rejection cascades (c.f. Haar-AdaBoost cascade and the SVM-cascade), which is, as the number of stages increases, FPR decreases significantly and TPR tends to keep high values.

7.5.4 Experiments with the SVM-cascade and the SRM-cascade

The SRM-cascade, proposed by O. Ludwig [Ludwig et al., 2011], uses linear SVMs to compose the stages of a negative-rejection cascade, similar to the SVM-cascade. However, the SRM-cascade follows a learning strategy which selects a suitable number of features for each ensemble stage in such a way as to control its complexity, in the Vapnik sense, in order to minimize the structural risk of each classification stage. The results, in the test set, for the SVM-cascade (in blue) and the SRM-cascade (in
Figure 7.8: Results, on the test set, regarding the true positive rate (a), and the false positive rate (b), for the S-AdaBoost (curve in blue), and for the M-AdaBoost (in black).

(a) True positive rate on the test set.  
(b) False positive rate on the test set.

Figure 7.9: Results, on the test set, regarding the true positive rate (a), and the false positive rate (b), for the SRM-cascade (blue-square markers) and for the SVM-cascade (black-circle markers).

(a) SVM-cascade and SRM-cascade TPR.  
(b) SVM-cascade and SRM-cascade FPR.

black) are shown in Fig. 7.9. The curves give sufficient condition to conclude that the SRM-cascade achieved slightly better results than the SVM-cascade, although this difference would be more significant as the number of stages increases. Nevertheless, the SRM-cascade training process demands much more CPU-time than the SVM-cascade, what means several hours of difference. Theoretical foundations of the SRM-cascade and its generalization ability has been demonstrated in [Ludwig et al., 2012].
7.5.5 Experiments with the context-based system: incidence regions

The context-based system affects the performance of the decision module in the MAP decision making stage (see Fig. 7.4). As described in previous sections, the contextual information is processed in terms of prior probabilities. Therefore, the class-conditional probability, or simply likelihood, is obtained from the classification method to be used in the decision module.

In the experiments with contextual information, the SVM-cascade was chosen to perform the object classification. Thus, the outputs of each stage of the cascade, obtained from the training set, were modeled according to a probabilistic distribution. Figure 7.10 shows the distributions for the first stage of the SVM-cascade. The histogram, plotted as function of the SVM scores, was modeled by a Normal distribution which will be used as likelihood function in the MAP decision. In this case, the problem reduces to the particular case of decision making using univariate-Normal densities. If \( P(\omega_1) = P(\omega_0) \), the decision boundary lies in the point \( x = 0 \). If the prior probabilities are not equal, that is, \( P(\omega_1) \neq P(\omega_0) \), the decision threshold shifts away from the more likely class. For the experiments using context information from the semantic map, the prior probability of pedestrian \( P(\omega_1) \), when the object lies in an incidence zone, was varied from 0.1 to 0.9 and the results are shown in Fig. 7.11. Note that, due to the mutually exclusive condition between the events, \( P(\omega_0) = 1 - P(\omega_1) \).

Denoting by \( T_\Lambda = 0 \) the decision threshold when \( P(\omega_0) = P(\omega_1) \), as \( P(\omega_{(0,1)}) \) varies during the experiments using context, \( T_\Lambda \) also varies. Let \( \mathcal{N}(\mu_1, \sigma_1^2) \) and \( \mathcal{N}(\mu_0, \sigma_0^2) \) be the Normal distributions for the positive and negative classes respectively, and denoting the posterior probability of the event be a pedestrian by \( P(\omega_1|x = 0) \), the value of \( T_\Lambda \) for any \( P(\omega_1) > 0 \) is the solution of the quadratic equation:

\[
2(\sigma_1^2 - \sigma_0^2)T_\Lambda^2 + 4(\sigma_0^2\mu_1 - \sigma_1^2\mu_0)T_\Lambda - (4\sigma_1^2\sigma_0^2W + 2\sigma_0^2\mu_1^2 - 2\sigma_1^2\mu_0^2) = 0
\]  (7.2)

\(^7\)Both classes are likely to occur.
7.5. EXPERIMENTS

Figure 7.10: Histogram and Normal distribution fitted to the output scores of the SVM in the first stage of the SVM-cascade. The decision is indicated by the vertical line (in red). At $x = 0$ the SVM guarantee 98% of $TPR$, according to the threshold $Thr_{tp}$ chosen for the cascade training (see Section 5.4.2).

where,

$$W = \log \frac{\sigma_1(P(\omega_1|x = 0) - P(\omega_1)P(\omega_1|x = 0))}{\sigma_0(P(\omega_1) - P(\omega_1)P(\omega_1|x = 0))}$$

Equation (7.3) is valid when the detected object is inside an incidence zone, otherwise $P(\omega_1)$ should be replaced by $P(\omega_0)$ in (7.3). Notice that the variable $x$ denotes the SVM-cascade score, where $x = 0$ is the ‘threshold’ when $P(\omega_1) = P(\omega_0)$.

7.5.6 Experiments with the context-based system: objects speed

The experiments reported in this section were performed using the SVM-cascade. The class-conditional probability density functions of the SVM-cascade (see Fig. 7.10) where combined with the probability mass functions that describe the object speed.
Figure 7.11: Results for the decision module using the MAP decision, where the likelihoods are given by the SVM-cascade and the priors come from the context-based module. The performance results are shown as function of the prior probability of the object be a pedestrian in the incidence zones (represented by $P$ in the texts arrows).

(see Fig. 7.6) through the MAP formulation, thus an object is classified as PED if

$$P(\omega_1|x) = \frac{P(x|\omega_1)P(\omega_1)}{P(x|\omega_1)P(\omega_1) + P(x|\omega_0)P(\omega_0)} > P(\omega_0|x)$$  \hspace{1cm} (7.4)

where $x$ is the SVM-cascade score and $P(\omega_i)$ are the priors that describe the object speed. The resulting ROC is shown in Fig. 7.12 (curve in blue) superimposed with the ROC for the SVM-cascade, without context (curve in black). Figure 7.12 shows that the context-based approach using object’s speed improved the performance of the SVM-cascade in terms of the $FPR$ for all the stages, although the $TPR$ was negatively affected. From the results, it can be concluded that the benefit of this approach is more significant for $nc > 9$. 


7.5. EXPERIMENTS

Figure 7.12: Results for pedestrian detection using the SVM-cascade and priors derived by the distributions of the estimated speed of the objects in the scene.

Summary

Results on the test set show that the single classifiers achieved good results in terms of \( FPR \), which is more evident for the SVM and ANN classifiers. As concerns the heterogeneous fusion schemes, \( F_\text{Min} \) fusion rule achieved the lowest \( FPR \). In terms of \( TPR \), the result using LDA was satisfactory, while \( F_\text{Max} \) obtained the highest value of \( TPR \), as expected. In conclusion, among the classification methods using single classifiers, a single SVM and the \( F_\text{Vote} \) rule are equivalent, while a single LDA is close to the \( F_\text{Average} \).

The cascade methods reached promising results in terms of \( TPR \), but the \( FPR \) results were slightly higher in comparison with the single classifiers. The SVM-cascade and the SRM-cascade are clearly equivalent in terms of performance on the test dataset. The Haar-AdaBoost cascade, on the other hand, had a divergent behavior among the cascade methods i.e., the \( FPR \) scores were very good but the \( TPR \) scores were quite low. In summary, the main difficulty on cascade methods is to predict the ‘optimal’ value for the number of stages.
Experiments using the context-based system demonstrate that contextual information enhances the performance of a classifier. Although the experiments were performed using the SVM-cascade, it is reasonable to expect similar tendency in the performance of other classification methods. Results using the estimated objects speed were not very promising. On the other hand, the use of incidence regions enhanced the classification performance when the prior was above 0.5, which demonstrates that the probability of pedestrian occurrence in some specific zones (e.g., crosswalks) is higher than other zones.
Chapter 8

Conclusion and Future Work

The problem of pedestrian classification and detection, in an urban environment, was studied and addressed by three systems: (1) LIDAR-based system; (2) Vision-based system; and (3) Information fusion system. To obtain solutions for this problem, many research topics were covered, such as: multi-sensor data acquisition, data preprocessing and filtering, binary classification, LIDAR and camera calibration, feature extraction, sample selection, parameter estimation, information fusion, decision making.

As concerns the LIDAR-based system, the reported results for pedestrian classification using the laser-based feature set are satisfactory, but the problem of pedestrian detection using only single or four-layers LIDAR still demands more research. Furthermore, studies and experiments are needed in a more systematic laser-based dataset; namely, extensive set of temporal sequences of ground-truth, indexed-time labeled segments (in laser space), should be necessary to explore novel methods that use temporal integration. The proposed multivariable segmentation method is suitable to integrate more attributes e.g., image-based features, in a straightforward way. In this case, only the threshold Thr would be changed according to a 'leaning' strategy. In terms of object tracking, the approach described in this thesis needs more research towards statistical decision methods for the problem of data association which, in my opinion, is a critical point in stochastic object tracking frameworks. Although solution have been proposed in [Premebida and Nunes, 2006], [Premebida et al., 2007] it is a open issue to be further explored.
CHAPTER 8. CONCLUSION AND FUTURE WORK

The problem of pedestrian classification and detection, using the Vision-based system, can be posed in terms of the trade-off between overfitting and generalization i.e., model selection (which is one of the key problem in pattern recognition). The main objective of pedestrian detection is to achieve high true positive rates and low false positives in the test dataset. Ensemble of classifiers, and rejection-cascades in particular, tend to be the right solution for the problem due to the strong imbalanced characteristic of realistic datasets. However, ensemble methods usually demand huge CPU-time on the training (which can represent months of work), they increase the complexity of the algorithms, and there is a lack of a strong and consistent theory to support, optimally, model selection for cascades or ensemble of classifiers to guarantee desirable performance on test (unseen) datasets.

Regarding the Information fusion system, it can be concluded that the integration of LIDAR and vision, in a complementary way, allows a significant decrease on the number of detection windows and consequently on the number of false positives. LIDAR and camera data fusion gives flexibility and robustness to the detection system and facilitates the integration of context-based information. Moreover, the results reported in Chapter 7 demonstrate enhancements on the performance in detecting pedestrians when a context-awareness information is available.

Perspective of further research

Finally, topics for future work are proposed. Open issues, worthy of future research, are discussed below.

Decision making: fusion of multi-classifiers, in heterogeneous or homogeneous decision frameworks, deserve more investigation. Moreover, consistent temporal tracking should reinforce dynamic (time-based) decision making situations. Additionally, the integration of classification methods using LIDAR and image-features concurrently, in the decision module, would be more explored.

Context-based pedestrian detection: the integration of contextual information in the pedestrian detection framework can improve its performance, as demonstrated by experiments using the estimated speed of the objects and their presence in ‘critical zones’. Further research in this topic would naturally be beneficial for the detection system, such as: explore daytime periods; use ‘global’ image-based features, which
would capture global properties of the scene; integrate LIDAR-based attributes of the detected objects.

**Infrastructure information**: on-board sensors are, definitively, required in detecting vulnerable road users but, FOV limitations and occluded ‘objects’ constitute very difficult problems for vehicle-mounted protection systems. Thus, infrastructure-based systems, in communication with ‘intelligent vehicles’, seem to be very useful for providing a more complete and complementary perception of the environment. In particular, camera-based systems, mounted in specific places of the environment, could communicate with the vehicle-based systems and provide high-level information of the surrounding which, on the other hand, could be explored by the vehicles to enhance the overall protection system.
Appendix A

Feature selection

Correlation coefficient

The correlation coefficient between two features $f_1$ and $f_2$, considered as two random variables, is the normalized quantity:

$$ cc = \frac{\text{cov}(f_1, f_2)}{\sigma_{f_1} \sigma_{f_2}} \quad (A.1) $$

where $\sigma_{f_i}$ is the standard deviation of $f_i$, and $\text{cov}(f_1, f_2) = \sigma_{f_1 f_2}^2$ is the covariance of the two variables. Two random variables whose correlation coefficient is zero are said to be uncorrelated (it is equivalent to their covariance being zero). If (A.1) has magnitude unity, they are linearly dependent, i.e., the two variables are correlated.

Two correlated variables bring the same information, in terms of information theory, and are considered redundant (they are linearly dependent), thus, one variable can be ‘chosen’ and the other can be ‘discarded’ with no loss of information. The conditionally independent among the features is a strong prerequisite for some classification methods, such as the Naive Bayes. Moreover, the complexity of the system is reduced when correlated features are rejected.

Information-based feature selection

Considering a discrete random variable $Y$, defined in the domain $\mathcal{Y}$, with probability
distribution \( p(y) = P\{Y = y\}, y \in \mathcal{Y} \), the entropy of \( Y \) is defined by:

\[
H(Y) = - \sum_{y \in \mathcal{Y}} p(y) \log(p(y))
\]  
(A.2)

and the conditional entropy \( H(Y|X) \) of \( Y \) given \( X \) with domains \( \mathcal{Y} \) and \( \mathcal{X} \), respectively, and joint probability distribution \( p(y, x) \) is defined as:

\[
H(Y|X) = - \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(y, x) \log(p(y|x))
\]  
(A.3)

The mutual information between \( Y \) and \( X \) is defined as:

\[
I(Y; X) = - \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(y, x) \log\left( \frac{p(x, y)}{p(x)p(y)} \right)
\]  
(A.4)

thus, the relation between the mutual information and entropy is more evident in the following expression:

\[
I(X; Y) = H(Y)H(Y|X) = H(X)H(X|Y)
\]  
(A.5)

**Relevance** The mutual information (A.4) has been used in [Meyer et al., 2008] [Peng et al., 2005] as measure of relevance of a set of random variables to another random variable \( (Y) \). Defining \( X = \{X_j : j \in A = \{1, \cdots, n\}\} \) as the set of input variables, and \( X_S \) as a subset of input variables, the measure of relevance of a variable \( X_i \) is the conditional mutual information

\[
I(X_i; Y|X_S) = I(X_S,i; Y)I(X_S; Y)
\]  
(A.6)

where \( X_S \) is a set of variables previously selected, and \( Y \) is the discrete output (the object class/label). Equation (A.6) express how much a variables \( X_i \) is relevant to the output \( Y \) given \( X_S \), ans its normalized quantity is used to select relevant features [Peng et al., 2005].

**Redundancy** The redundancy among \( n \) random variables is defined by [Meyer et al., 2008]

\[
R(X_1; \cdots; X_n) = \sum_{i=1}^{n} H(X_i)H(X_1,\cdots,n)
\]  
(A.7)
Considering the redundancy between two variables, $X_i$ and $X_j$, (A.7) takes the form

$$R(X_i; X_j) = H(X_i) + H(X_j) - H(X_{i,j}) = I(X_i; X_j)$$  \hspace{1cm} (A.8)

In summary, the relevance measure concerns the relation between inputs and output, while the redundancy applies to input variables.
This page is intentionally left blank
Bibliography


[Premebida et al., 2009b] Premebida, C., Ludwig, O., and Nunes, U. (October, 2009b). Exploiting lidar-based features on pedestrian detection in urban scenarios. USA.


