Henry Antônio Roncancio

**Detecção e Classificação de Objetos em
Ambientes Externos para Navegação de um
Veículo de Passeio Autônomo**

Utilizando Fusão de Dados de Visão Artificial e Sensor Laser

Dissertação apresentada à Escola de Engenharia
de São Carlos da Universidade de São Paulo,
como parte dos requisitos para obtenção do título
de Mestre em Ciências, Programa de Engenharia
Mecânica

Área de Concentração: Dinâmica das Máquinas &
Sistemas

Orientador: Prof. Assoc. Marcelo Becker

São Carlos
2015

ESTE EXEMPLAR TRATA-SE DA
VERSÃO CORRIGIDA. A VERSÃO
ORIGINAL ENCONTRA-SE
DISPONÍVEL JUNTO AO
DEPARTAMENTO DE ENGENHARIA
MECÂNICA DA EESC-SP

Henry Roncancio Velandia

**Object Detection and Classification in Outdoor
Environments for Autonomous Passenger
Vehicle Navigation**

Based on Data Fusion of Artificial Vision System and LiDAR Sensor.

Dissertation presented to São Carlos Engineering
School at University of São Paulo, as part of the
requirements to obtain the title of Master in Me-
chanical Engineering

Area: Systems and Machines Dynamics
Supervisor: Assoc. Prof. Marcelo Becker

São Carlos
2015

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO,
POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS
DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

R769o Roncancio Velandia, Henry Antonio
Object Detection and Classification in Outdoor
Environments for Autonomous Passenger Vehicle
Navigation Based on Data-fusion of an Artificial Vision
System and Lidar sensor. / Henry Antonio Roncancio
Velandia; orientador Marcelo Becker. São Carlos, 2014.

Dissertação (Mestrado) - Programa de Pós-Graduação
em Engenharia Mecânica e Área de Concentração em
Dinâmica de Máquinas e Sistemas -- Escola de Engenharia
de São Carlos da Universidade de São Paulo, 2014.

1. Autonomous Vehicles. 2. Computer Vision. 3.
Pedestrian Detection. 4. Obstacle Recognition. 5. Point
Cloud Segmentation. 6. Data Fusion. 7. World
Perception. 8. Machine Learning. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **HENRY ANTONIO RONCANCIO VELANDIA**.

Título da dissertação: "Detecção e classificação de objetos em ambientes externos para navegação de um veículo de passeio autônomo utilizando fusão de dados de visão artificial e sensor laser".

Data da defesa: 30/05/2014

Comissão Julgadora:

Resultado:

Prof. Associado **Marcelo Becker (Orientador)**
(Escola de Engenharia de São Carlos/EESC)

APROVADO

Prof. Associado **Jun Okamoto Junior**
(Escola Politécnica/EP-USP)

APROVADO

Prof. Dr. **Stefano Cattani**
(Universidade de Parma)

APROVADO

Coordenador do Programa de Pós-Graduação em Engenharia Mecânica:
Prof. Associado **Marcelo Areias Trindade**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Denis Vinicius Coury**

Dedictory

This work is dedicated to all the people that believe in and dream about a world with autonomous vehicles, as a way of living in a safer, more comfortable and sustainable place.

I dedicate it to my mother for making me a dreamer.

Acknowledgment

I would like to greatly thank Prof. Marcelo Becker for his support in carrying out this project.

I would like to thank Prof. Alberto Broggi from Vislab Srl. at Parma University for having invited me to take part in the Vislab's projects. It was a great experience for me. I would like to thank PhD. Stefano Cattani, PhD. Paolo Medici and PhD. Rean Isabella Fedriga for sharing their knowledge with me.

I would also like to thank the colleagues and Professors of LabRoM and Machine Dynamics Lab at the Engineering School of São Carlos, especially André Carmona Hernandez, Henry Borrero Guerrero, Raphael Wegner, Ayrton Conceição Junior, Björn Jensen, André Salomão Brito Mendonça, Rafael Coronel Bueno Sampaio, Luciano Diniz, Rodolfo Favaretto, John Faber Arcila, Hugo Velasco and Adriana Bonilla for the time spent supporting this project.

I would like to thank FAPESP for its financial support.

In general, I would like to thank Brazilian people for the friendly welcome to USP and São Carlos.

“It is vital to understand, moreover, that technological innovation does not merely combine and recombine machines and techniques. Important new machines do more than suggest or compel changes in other machines - they suggest novel solutions to social, philosophical, even personal problems. They alter man’s total intellectual environment - the way he thinks and looks at the world.”

Alvin Toffler. *Future Shock*. 1970.

Abstract

RONCANCIO, H. **Object Detection and Classification in Outdoor Environments for Autonomous Passenger Vehicle Navigation Based on Data-fusion of an Artificial Vision System and Laser Sensor.** 2014. Dissertation (Master of Science) - Engineering School of São Carlos, University of São Paulo (EESC - USP), 2014.

This research project took part in the SENA project (Autonomous Embedded Navigation System), which was developed at the Mobile Robotics Lab of the Mechatronics Group at the Engineering School of São Carlos, University of São Paulo (EESC - USP) in collaboration with the São Carlos Institute of Physics. Aiming for an autonomous behavior in the prototype vehicle this dissertation focused on deploying some machine learning algorithms to support its perception. These algorithms enabled the vehicle to execute artificial-intelligence tasks, such as prediction and memory retrieval for object classification.

Even though in autonomous navigation there are several perception, cognition and actuation tasks, this dissertation focused only on perception, which provides the vehicle control system with information about the environment around it. The most basic information to be provided is the existence of objects (obstacles) around the vehicle. Information about the sort of object it is also provided, i.e., its classification among cars, pedestrians, stakes, the road, as well as the scale of such an object and its position in front of the vehicle.

The environmental data was acquired by using a camera and a *Velodyne* LiDAR. A ceiling analysis of the object detection pipeline was used to simulate the proposed methodology. As a result, this analysis estimated that processing specific regions in the

image (i.e., Regions of Interest, or RoIs), where it is more likely to find an object, would be the best way of improving our recognition system, a process called image normalization. Consequently, experimental results in a data-fusion approach using laser data and images, in which RoIs were found using the LiDAR data, showed that the fusion approach can provide better object detection and classification compared with the use of either camera or LiDAR alone. Deploying a data-fusion classification using RoI method can be executed at 6 Hz and with 100% precision in pedestrians and 92.3% in cars. The fusion also enabled road estimation even when there were shadows and colored road markers in the image. Vision-based classifier supported by LiDAR data provided a good solution for multi-scale object detection and even for the non-uniform illumination problem.

Keywords: Artificial Intelligence. Mobile Robotics. Advanced Driver-Assistance Systems. Pedestrian Recognition, Car Detection, Fusion Sensor.

Resumo

RONCANCIO, H. **Detecção e Classificação de Objetos em Ambientes Externos para Navegação de um Veículo de Passeio Autônomo utilizando Fusão de Dados de Visão Artificial e Sensor Laser**. Dissertação de mestrado - Escola de Engenharia de São Carlos, Universidade de São Paulo (EESC - USP), 2014.

Este projeto de pesquisa fez parte do projeto *SENA* (Sistema Embarcado de Navegação Autônoma), ele foi realizado no Laboratório de Robótica Móvel do Grupo de Mecatrônica da Escola de Engenharia de São Carlos (EESC), em colaboração com o Instituto de Física de São Carlos (IFSC). A grande motivação do projeto SENA é o desenvolvimento de tecnologias assistidas e autônomas que possam atender às necessidades de diferentes tipos de motoristas (inexperientes, idosos, portadores de limitações, etc.).

Vislumbra-se que a aplicação em larga escala desse tipo de tecnologia, em um futuro próximo, certamente reduzirá drasticamente a quantidade de pessoas feridas e mortas em acidentes automobilísticos em estradas e em ambientes urbanos. Nesse contexto, este projeto de pesquisa teve como objetivo proporcionar informações relativas ao ambiente ao redor do veículo, ao sistema de controle e de tomada de decisão embarcado no veículo autônomo. As informações mais básicas fornecidas são as posições dos objetos (obstáculos) ao redor do veículo; além disso, informações como o tipo de objeto (ou seja, sua classificação em carros, pedestres, postes e a própria rua mesma), assim como o tamanho.

Os dados do ambiente são adquiridos através do emprego de uma câmera e um Velodyne LiDAR. Um estudo do tipo ceiling foi usado para simular a metodologia da detecção dos obstáculos. Estima-se que, após realizar o estudo, que analisar regiões específicas da imagem, chamadas de regiões de interesse, onde é mais provável encontrar um obstáculo,

é o melhor jeito de melhorar o sistema de reconhecimento. Observou-se na implementação da fusão dos sensores que encontrar regiões de interesse usando LiDAR, e classificá-las usando visão artificial fornece um melhor resultado na hora de compará-lo com os resultados ao usar apenas câmera ou LiDAR. Obteve-se uma classificação com precisão de 100% para pedestres e 92,3% para carros, rodando em uma frequência de 6 Hz. A fusão dos sensores também forneceu um método para estimar a estrada mesmo quando esta tinha sombra ou faixas de cor. Em geral, a classificação baseada em visão artificial e LiDAR mostrou uma solução para detecção de objetos em várias escalas e mesmo para o problema da iluminação não uniforme do ambiente.

Palavras-Chave: Inteligência Artificial. Robótica Móvel. Sistema Avançado de Assistência ao Motorista (ADAS). Reconhecimento de pedestres. Detecção de carros. Fusão de sensores. Aprendizagem de máquina.

Publications

1. RONCANCIO, H. ; HERNANDES, A. C. ; BECKER, M. Ceiling analysis of pedestrian recognition pipeline for an autonomous car application. In: 2013 IEEE Workshop on Robot Vision (WORV 2013), 2013, Clearwater Beach. 2013 IEEE Workshop on Robot Vision (WORV), 2013. p. 215.
2. RONCANCIO, H. ; HERNANDES, A. C. ; BECKER, M. “*Vision-based System for Pedestrian Recognition using a Tuned SVM Classifier*” - In: IEEE Workshop on Engineering Applications 2012 (WEA-2012), 2012, Bogotá. Proc. of IEEE Workshop on Engineering Applications 2012, 2012. v. 1. p. 1-6.
3. RONCANCIO H., BECKER M, BROGGI A., CATTANI S. Traversability Analysis Using Terrain Mapping and Online-trained Terrain Type Classifier. IEEE Intelligent Vehicle Symposium 2014. June 8-12. Dearborn, U.S.A.
4. HERNANDES, ANDRE C. ; BRITO, ANDRE S. ; RONCANCIO, HENRY ; MAGALHAES, DANIEL V. ; BECKER, MARCELO ; SAMPAIO, RAFAEL C. B. ; JENSEN, BJORN T. GISA: A Brazilian platform for autonomous cars trials. In: 2013 IEEE International Conference on Industrial Technology (ICIT 2013), 2013, Cape Town. 2013 IEEE International Conference on Industrial Technology (ICIT), 2013. p. 82.
5. RONCANCIO, H. ; HERNANDES, A. C. ; BECKER, M. . “*Reconhecimento de Pedestres à Frente de um Veículo de Passeio Usando uma Máquina de Vetor de Suporte Ajustada*” - In: CONEM 2012 - Congresso Nacional de Engenharia Mecânica, 2012, São Luiz - MA. Anais do CONEM 2012. Rio de Janeiro : ABCM, 2012. v. 1. p. 1-10.

6. BRITO, ANDRE S. ; HERNANDES, ANDRE C. ; RONCANCIO, H ; JUNIOR, A. ; JESEN, B. ; MAGALHAES, D. V. ; Becker, Marcelo . GISA: Plataforma de teste para navegação autônoma em ambiente urbano. In: VII Simpósio Brasileiro de Engenharia Inercial - SBEIN 2012, 2012, São José dos Campos, SP. Annais do Simpósio Brasileiro de Engenharia Inercial, 2012.

List of Figures

1.1	Deaths for road user category (World Health Organization , 2013).	5
1.2	Sensors' mounting locations on <i>Gisa</i> prototype.	6
1.3	<i>Gisa</i> vehicle.	7
1.4	Side view of <i>Gisa</i> vehicle.	8
1.5	Bird's-eye view of <i>Gisa</i> vehicle.	9
1.6	<i>Gisa</i> 's operation basis.	14
1.7	Object detection scheme based on data-fusion.	16
2.1	Machine Learning Modeling.	20
2.2	Steps to obtain the <i>HOG</i> descriptor (DALAL AND TRIGGS , 2005).	22
2.3	Local oriented gradients for a 16x8-cell grid. <i>HOG</i> descriptor. Gradients vary among examples, which allow this feature to be used as descriptor.	24
2.4	Local oriented gradients for a 16x8-cell grid. <i>HOG</i> descriptor for pedestrians.	25
2.5	SVM margin. This figure shows the hyperplane of separation for two classes. Points over the dotted line are the support points. JAGGI (2008).	26
2.6	Flow diagram for SVM tuning.	30
2.7	Cross-validation cycle.	31
3.1	Point cloud visualization. It is particularly colored according to the azimuth angle. Following the rainbow color pattern, 0° is set to violet color.	34

3.2	Spherical coordinate System. Source: Wikipedia.	35
3.3	Organized cloud. (WEGNER , 2013).	36
3.4	Organized data schema.	37
3.5	Organized scheme for normal values. The normal is calculated based on the figure 3.4.	37
3.6	Green arrows are normals to the road. Red arrows are normal to the cyclist.	38
3.7	Plane and normal vector for Points A, B and C.	39
3.8	Difference of normal calculation. (IOANOU AND TAATI , 2012).	39
3.9	Procedure for regrouping points in objects.	41
3.10	Velodyne HD-32's vertical and horizontal resolution.	42
3.11	Distance between two <i>Velodyne</i> points.	43
3.12	Resulting segments from initial grouping. Road segments in several colors, except the red ones that are unknown. Color in segments changes on the basis the distance to the vehicle and following the rainbow pattern.	44
4.1	Imaginary scenario with 10 pedestrians and 1 car. It is used for comparing two pedestrian classifiers (left and right side) using precision and recall. Objects inside the circle means those are classified as pedestrian.	50
4.2	Imaginary scenario with 7 pedestrians and 3 cars. It is used for comparing two pedestrian classifiers (left and right side) using precision, recall and F_1 score. Objects inside the circle means those are classified as pedestrian.	51
4.3	INRIA's Normalized positive images (DALAL AND TRIGGS , 2005).	52
4.4	LabRoM's normalized positive images.	53
4.5	Data Distribution used in CV method.	55
4.6	(a) Non-pedestrian image. (b) Pedestrian image.	56
4.7	Examples of car positive images.	58
4.8	Examples of car negative images.	59

4.9	Machine learning pipeline.	60
4.10	Pedestrian recognition pipeline.	61
4.11	Machine learning pipeline.	62
4.12	Ceiling analysis results.	67
5.1	Image overlaid with the classified point cloud.	70
5.2	Image with objects boxes. Blue ones are cars, green ones are stakes and red ones are pedestrians. White ones correspond to actually the regions defined for the points. Color boxes are bigger because actual objects are supposed to be larger than the region defined by the points.	71
5.3	Pedestrian detection fusion scheme.	74
5.4	Road estimation using watershed segmentation	75
5.5	Segmentation using graph-cut algorithm with min-cut/max-flow optimization. Figure (a) shows a blue box that defines a region with potential car. This region was obtained using the pre-classification with LiDAR. Figure (b) is a mask image that describes the pixels in the image according to these rules: light-gray pixels are probably foreground/object, dark-gray pixels are probably background. Balanced-gray pixels are true background, and the white pixels are true foreground/object. In figure (c), colored pixels are the object/foreground extraction.	78
5.6	Example of a directed capacitated graph with two terminals in an image of 3 x 3 pixels. Edge costs are reflected by their thicknesses. The figure describes the process of labelling an image, creating the graph, and then cutting the graph to produce a segmentation of the image.(BOYKOV AND KOLMOGOROV , 2004).	81
6.1	ROS's Graph Visualization.	87
6.2	Fusion sensor visualization node.	88

7.1	Pedestrian classification using sliding-window method.	90
7.2	Classification using only LiDAR data.	91
7.3	Classification using fusion of LiDAR and Image. The road has been segmented using watershed, redder regions show the resulting segmented road. Red boxes are pedestrians; blue boxes are cars; and green boxes are stakes.	93
7.4	Pedestrian classification using fusion of LiDAR and Images.	94
7.5	Car classification using fusion of LiDAR and Images.	95
7.6	Segmentation using graph-cut algorithm with min-cut/max-flow optimization.	97

List of Tables

2.1	Confusion matrix. Relationships between the predicted and actual class. .	23
3.1	Configured parameter for classification.	48
4.1	Preliminary test using the INRIA two-set original split.	54
4.2	Tuned SVM performance evaluation with CV method.	55
4.3	Metrics calculated for car classifier.	58
4.4	Simulation Table for Ceiling Analysis.	63
4.5	Metrics calculated for various classifier settings and data.	64
4.6	Estimation of potential upside due to each element.	67
6.1	Computer Specifications.	86
7.1	Number of objects in the test scenario	98
7.2	Pedestrians in only the street.	98
7.3	Pedestrian in the entire scenario.	99
7.4	Cars in only the street.	99
7.5	Cars in the entire scenario.	99
7.6	Pedestrian detection results. Comparison among the approaches of this dissertation.	100

A.1 Sensors on *Gisa* prototype 113

List of Abbreviations

<i>Term</i>	<i>Description</i>
ACC	Active Cruise Control
ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
BOF	Bayesian Occupancy Filter
CCD	Charge-coupled device
CV	Cross-Validation Method
DoN	Difference of normals
EESC	Escola de Engenharia de São Carlos
FP	False positive
FoV	Field of View
GPS	Global Positioning System
HOG	Histogram of Oriented Gradients
HFoV	Horizontal Field of View
HSV	Hue-Saturation-Value
ICMC	Instituto de Ciências Matemáticas e Computação
INRIA	Institut National de Recherche en Informatique et en Automatique
IMU	Inertial Measurement Unit
IPM	Inverse Perspective Mapping
LabRoM	Laboratório de Robótica Móvel - EESC - USP
LASER	Light Amplification by Stimulated Emission of Radiation
LIDAR	Light Detection and Ranging
LKS	Lane Keeping System

<i>Term</i>	<i>Description</i>
MIT	Massachusetts Institute of Technology
PDBS	Point-Distance Based Segmentation
RBF	Radial-Based Fuction
RGB	Red-Green-Blue
RoI	Regions of Interest
ROS	Robot Operating System
RVIZ	ROS interface for Robot Visualization
RTK-GPS	Real Time Kinematic Global Position System
SENA	Sistema Embarcado de Navegação Autônoma
SIFT	Scale-Invariant Feature Transformation
SVM	Support Vector Machine
TBM	Transferable Belief Model
USP	Universidade de São Paulo
VFoV	Vertical Field of View
WHO	World Health Organization

List of Symbols

<i>Symbol</i>	<i>Description</i>	<i>Units</i>
$J(\theta)$	Cost function	-
f_i	Similarity function used as Kernel	-
θ	Parameter Vector of the estimated hypothesis	-
$l^{(i)}$	Landmark based on example i	-
$x^{(i)}$	Feature vector for example i	-
$cost_1$	Cost for positive examples	-
$cost_0$	Cost for negative examples	-
n	Number of features	-
m	Number of examples	-
N	The normal vector of the point cloud surface	-
r	Radial distance from the sensor to any point in the world	-
θ	The azimuthal angle that describes a point cloud in the world	-
ϕ	The polar angle in the vertical field of view associated with each laser beam	-

<i>Symbol</i>	<i>Description</i>	<i>Units</i>
x	Relative position of the camera to the velodyne sensor in the x direction	[m]
y	Relative position of the camera to the velodyne sensor in the y direction	[m]
z	Relative position of the camera to the velodyne sensor in the z direction	[m]
α	Rotation around the x-axis to align the camera to the velodyne sensor	[degrees]
β	Rotation around the y-axis to align the camera to the velodyne sensor	[degrees]
β_{openi}	Opening angle of the camera around the y-axis	[degrees]
γ	Rotation around the z-axis to align the camera to the velodyne sensor	[degrees]
γ_{openi}	Opening angle of the camera around the z-axis	[degrees]

Contents

Abstract	xi
Resumo	xiii
Publications	xv
List of Figures	xvii
List of Tables	xxi
List of Abbreviations	xxiii
List of Symbols	xxv
1 Introduction	1
1.1 Insight into Intelligent Vehicles Perception	1
1.2 Motivation - Why Spend Time and Money Designing Autonomous Vehicles?	4
1.3 <i>Gisa</i> Prototype	6
1.4 Review of Related Work	10
1.5 Problem Statement	13
1.6 Document Outline	18
2 Machine Learning Methods	19
2.1 Machine Learning on Computer Vision	20
2.1.1 Image Normalization	21
2.1.2 Histogram of Oriented Gradients (HOG)	21
2.1.3 Metrics for Model Validation	23
2.1.4 Tuned Support Vector Machine (SVM)	26
2.1.5 Cross-Validation Method (CV)	28

3	Laser Detection and Classification	33
3.1	Point Cloud Visualization	33
3.2	The Point Cloud in the Spherical Coordinates System	34
3.3	Surface Orientation. The Concept of Normal to the Surface	36
3.4	Difference of Normals (DoN)	37
3.5	Segmentation	40
3.5.1	Distance Between Two Points	40
3.5.2	Regrouping Process	43
3.6	Classification based on LiDAR	47
4	Object Classification Based on Computer Vision	49
4.1	Pedestrian Detection	49
4.1.1	Training Data	51
4.1.2	Preliminary Tests	52
4.2	Car Detection	57
4.2.1	Car Training Data	57
4.3	Ceiling Analysis Approach	59
4.3.1	Ceiling Analysis Simulation	62
4.3.2	Image Pre-Processing Element	63
4.3.3	Simulation Data	64
4.3.4	Ceiling Analysis	64
4.4	The Fusion Approach	68
5	Data-Fusion Classification	69
5.1	LiDAR Data and Image Overlaying	69
5.1.1	Synchronization Policy	69
5.1.2	Fusion Process	70
5.1.3	Image Cropping	72
5.2	Object Classification Schema	73
5.3	Road Estimation	73
5.3.1	Road Estimation using Watershed Segmentation	73
5.4	Foreground Extraction Using Graph-Cut Algorithm	77
5.4.1	Graph-Cut Theory	79
5.4.2	Min-Cut and Max-Flow Problem	80

6	Software Implementation	83
6.1	Middle-Ware Platform: Robot Operating System (ROS)	83
6.2	Software Description	84
6.3	Data Fusion Synchronization	84
6.4	ROS Nodes and Topics in the Visualization Graph	85
6.5	Hardware - Computer Specifications	86
7	Experimental Results	89
7.1	Classification Using Only Computer Vision	89
7.2	Classification Using Only Point Cloud from LiDAR	91
7.3	Classification Using Data Fusion Approach	92
7.4	Car Segmentation Using Foreground Extraction	96
7.5	Measurements on Classification Data	98
7.6	Classification Results	98
8	Conclusions and Future Work	103
	Appendix A Sensors in SENA Project	113
	Appendix B Cost Function Modeling	115
B.0.1	Model for Multiple Features	115
B.0.2	Cost function	116
B.0.3	SVM Cost Fuction using RBF kernel f	117

Chapter 1

Introduction

1.1 Insight into Intelligent Vehicles Perception

The first requirement to navigate in the world is to be able to sense it. However, even though sensors provide data from the environment, some kind of processing is needed to understand it correctly. This is the same process by which our senses comprehend images from the environment, however many things that happen to us are unnoticeable. For instance, a driver should know all the traffic signals in order to recognize them in the street; but, he might miss any of them even though he is able to see it. In the task of understanding the scenario, the data should be analyzed to seek for clues about the scene, from simple examination to other more complex analysis, such as *object detection and classification*. Analysis to better *understand* the environment is commonly called *perception* (RUSSEL AND NORVIG , 1994).

Current sensors yield a variety of information. Some of the most well-known data are: *point clouds* of the environment in a wide field of view and range, global position information, inertial measurements, proximity detections and *visual* data that our senses sometimes do not pick up. (See Appendix A). Perhaps not surprisingly, even though the current sensors appeared recently compared to our evolved senses, they are able to sense much more of the world in particular scenarios. LiDAR range sensors, for example, detect objects even at night (without any artificial light) and entirely around the vehicle

(VELODYNE , 2013). Cameras, which provide data quite similar to our eyes, are now able to adjust themselves automatically to variations in the illumination and brightness. They are equipped with functions of auto-balance, auto-exposure, auto-gain and auto-brightness, which allow them to adjust themselves quickly to the changing environment. They are even able to do shadow correction in real time (ALLIED VISION TECNOLOGIES , 2014). Nonetheless, this sensor processing does not solve the problem of perception by itself because it goes beyond what is “seen.” It certainly helps, but it still needs to extract the useful information.

The next step after sensing is perception, which is the task of carefully watching the scene in order to correctly understand the rules of the environment and other objects’ behavior. Perception is a basic task for almost all robots, but for *autonomous* and *driver-assistance vehicles* it is particularly special. The great difference is that such a vehicle is moving in an *outdoor* environment, so it has to deal with uncontrolled conditions, changing scenarios and, therefore, *risky* situations, which become even more critical in *passenger vehicles*. So, these vehicles should have *intelligence* to successfully navigate in such a scenario. If we study this from an evolutionary point of view, the need for perception in moving beings became an essential part for the appearance of intelligence. In fact, “brains [where intelligence resides] are an evolutionary prerequisite for guided movement in primitive animals and the reason for this becomes obvious. Clearly, active movement is *dangerous* in the absence of an internal plan subject to sensory modulation.” (LLINÁS , 2002). But, what does “intelligence” actually mean? Going back to the field the biology, we can find the meaning in the brain’s functions. Among the functions that make the brain the body’s central control, *prediction* is its *ultimate* function (LLINÁS , 2002). Therefore, what makes the vehicle intelligent is its ability to predict - the behavior of other objects, the path to be followed, and the next action to be completed (considering the rules of the environment). Prediction is the key function for goal-oriented navigation, because the vehicle should anticipate the next action in the environment on the basis of what it senses. Moreover, any change in its immediate environment will require a reaction to avoid a collision or injuring someone.

Computer vision and point cloud processing techniques (regarding the *laser sensor*) are well-known robot-related techniques used for understanding the scene. They work on

the basis of mathematical models, not only to model the outdoor environment but also to resolve predictions (NG , 2012). The *machine learning* techniques are largely the core of prediction. Some models use supervised learning in which they are taught with training examples provided by a human, similar to the biological learning process. However, this is *artificial* instead of “natural.” Other unsupervised techniques make predictions based on statistics, clustering and feature extraction without any kind of training (NG , 2012). Nonetheless, as any with prediction, they are just approximations, and therefore, they cannot always provide enough scenario reliability to execute a future action. Some *data-fusion* approaches are carried out in order to compensate for the impossibility of having perfect prediction in unknown data (CASTELLANOS AND NEIRA , 2001) (OZGUNER , 2005) (MONTEIRO et al. , 2006) (NASHASHIBI et al. , 2008) (LUDWIG et al. , 2009) (BAIG , 2010). The main idea behind this is to fuse predictions to perceive the scenario in different ways, such as the diversity of sensors in humans.

Among the supervised approaches, *classifiers* are robust representation models that predict the class or label of new, unknown data based on a previous training; they become a machine’s representation of the world (NG , 2012). At the end, we also have a virtual, numeric world representation stored carefully in the computer’s memory. Every time the algorithm is started, the model is retrieved as if the robot vehicle is waking up. Supervised classifiers are associated to the field of neural networks (HAYKIN , 1999). Hence, most of the information is going to be transformed into numbers, that is, the numbers are weights in the connections among neurons, similar to a design blueprint. However, this doesn’t represent reality, but just by using those numbers the world can be recalled. In the *memory*, the connections and the weights are more important than the neurons themselves. The neurons are the elements in the system, but in certain way the weights are actually the useful data. A classifier is not only a memory to retrieve data, but it is also a kind of machine that is able to find a pattern even when the information is incomplete or noisy. This enables the system to make predictions (HAYKIN , 1999).

Current autonomous and driver-assistance vehicles are the first robots with an *artificial intelligence* capable of moving by themselves in the same highly dynamic world as humans. In contrast to humans, these vehicles have a particular behavior that does not mimic humans or other animals. They have their own brain design and even their own sensors.

In his book “I of the Vortex, from Neurons to *Self*,” (LLINÁS , 2002) Neuroscientist Dr. Llinas explored the nature of the mind and concluded the following: “The brain’s control of organized movement gave birth to the generation and nature of the mind.” Having talked about the *functional similarities* between a brain and the vehicle’s “brain,” such as prediction and memory, intended to enable movement, we can say that these vehicles have become the first *conscious* robots because, in fact, they are conscious of many things that are happening around them in order to move by themselves in the world. Following this sequence of ideas, they will probably evolve from performing machine tasks to being robot *selves*.

1.2 Motivation - Why Spend Time and Money Designing Autonomous Vehicles?

Worldwide road traffic accidents cause the death of nearly 1.3 million people every year, and injure 20-50 million more. This situation is even worse if we consider vulnerable road users (like pedestrians and cyclists). This category of road users represents today around 27% of the victims. Half of all road traffic deaths are among pedestrians (22%), cyclists (5%), and motorcyclists (23%) (World Health Organization , 2013).

In few countries of Central and South America, for instance, the deaths of vulnerable people became as high as 70% (World Health Organization , 2009). According to the last report, the Americas have 23% and Europe 27% of the world’s total percentage of pedestrian fatalities. Figure 1.1 shows the percentages for all categories (World Health Organization , 2013).

According to the World Health Organization (WHO), if this death trend keeps going, it is predicted that road traffic injures will be the fifth-leading cause of death in the world by 2030 (World Health Organization , 2011). The WHO has named this decade: “The Decade of Action for Road Safety,” after its plan to reduce that forecasted level. It expects that 5 million lives will be saved (before 2020) just by strengthening road safety management, making safer roads and safer vehicles, and improving education and post-crash response.

Figure 1.1: Deaths for road user category (World Health Organization , 2013).

When it comes to safer vehicles and roads, many efforts are being made to design autonomous vehicles that can deal with real traffic scenarios. Most car accidents are attributed to human (driver) error (World Health Organization , 2013), improper crossing of roadways, playing or working in the road, and failure to yield right of way, among other factors (National Highway Traffic Safety Administration of U.S. Government , 2009). The development not only of autonomous vehicles but also intelligent transportation systems will hopefully bring reduction in road fatalities. Even though implementing it on a large scale in the cities is still just a plan, current development focuses on designing pre-emptive systems to help the driver avoid collisions.

1.3 *Gisa* Prototype

Aiming to implement this kind of vehicles on actual roads, the Mobile Robotics Laboratory (LabRoM) at the São Carlos School of Engineering - University of São Paulo (EESC-USP) is currently developing the SENA Project (*Sistema Embarcado de Navegação Autônoma*). It provides the vehicle with a combination of various sensors, such as laser sensors, cameras, IMU (Inertial Measurement Unit), and GPS units, to perceive the environment that surrounds the vehicle and to control its motion. A general diagram is presented in Figure 1.2.

Figure 1.2: Sensors' mounting locations on *Gisa* prototype.

One of the main components of this vehicle is the perception module, namely the object detection module (pedestrians, cars, stakes and the road), which uses images captured by a camera and data from LiDAR sensors to perceive the outdoor environment. Figures 1.5 and 1.4 show the horizontal and vertical field of view of *Gisa*'s sensors.

On the *Gisa* prototype (See Figure 1.2), the LiDAR sensing is based on an optical

Figure 1.3: *Gisa* vehicle.

remote sensing technology that measures properties of light reflected to obtain the distance and/or other information regarding a particular distant object. This technology is also known as LADAR (Laser Detection And Ranging). The most robust LiDAR in the Gisa prototype is the Velodyne HDL-32, which is positioned on the roof of the vehicle to detect all objects around it. Furthermore, there is a camera looking in front of the vehicle.

Figure 1.4: Side view of *Gisa* vehicle.

Figure 1.5: Bird's-eye view of *Gisa* vehicle.

1.4 Review of Related Work

Perceiving the surrounding environment is not an easy task even for a human being who has such evolved senses. To detect an object when the light changes may be difficult for a human, even though eyes adapt by themselves to light changes. That is why many traffic accidents occur in the late afternoon/early evening. In addition, another cause of accidents to be considered is the lack of attention to surroundings (National Highway Traffic Safety Administration of U.S. Government , 2009).

Such problems in human perception posit the intelligent vision system as a way of perceiving the environment with more confidence. Moreover, the system might be more robust if it is based not only on a camera, but also on an object detection system, such as the LiDAR. For example, for pedestrian detection there are several approaches in the literature. They are mainly focused on Computer Vision with either mono or stereo cameras (BERTOZZI et al. , 2007), analysis of laser sensor data (KIDONO et al. , 2011), and even data fusion of both laser and camera (SERFLING et al. , 2009) (LUDWIG et al. , 2009) (BAIG , 2010). This dissertation will show detection results using these approaches. In fusion approach, the intent of this integration is to perceive the environment in different ways, just as in human perception. Furthermore, it is expected that by increasing the number of sensors, the information captured will be greater in content. Illumination changes are also one of the most difficult challenges to overcome.

Some detection systems based only on LiDAR were developed in (HABERMANN , 2010) (MIGUEL AND GOMES , 2010). The authors proposed using multiple sensors to improve their system by reducing measurement failures, and to use cameras to capture images of objects to improve their detection. The use of the fusion of LiDAR and camera look promising, not only for detection but also for classification.

Becker et al. (BECKER et al. , 2007) (BECKER et al. , 2009) used LiDAR in front of the vehicle to detect and classify obstacles on the road, and for such sensors, developed an algorithm for detecting the movement of those objects. To initially sort the objects, they grouped the detected points based on the fact that near points belong to the same object. After this, those groups or segments were classified according to their size. Moreover,

the objects were also sorted according to their geometrical characteristics. However, they found that the presence of shrubs and leaves produce false readings because the algorithm identifies them as moving objects on the road. To overcome these difficulties, they proposed the use of urban area maps and embedded cameras. According to the authors, the use of a camera increases overall system performance by fusing its data with the LiDAR. In this way, it may be possible to achieve a more detailed an obstacles classification and recognition system (for example, to differentiate cars from pedestrians, sidewalks, etc.).

Nashashibi et al. (NASHASHIBI et al. , 2008) proposed a strategy for data fusion through a "perceptual system." The system is based on two modules, a generator of hypotheses and a validation step. Each analysis in relation to a particular sensor or algorithm applied to the data (It is worth of noting that several algorithms can be applied to the same data.) provides a level of confidence, ranging between 0 and 1 in the real presence of a class object. Thus, the confidence levels obtained for sensor and/or algorithm are combined to obtain a final score to be used for decision making in classification. The hypothesis generator has the objective of locating regions in the image containing possible vehicles or pedestrians, or any object, so that this region can be further explored (COSTA et al. , 2011). This dissertation proposes using the data provided by the LiDAR to generate hypotheses. Polygonal segmentation and similarities between object points provide an initial hint about the obstacles. After generating the hypotheses, some algorithms of image processing are applied to regions of interest in order to improve the level of confidence. According to Nashashibi et al. (NASHASHIBI et al. , 2008), these fusion functions are already an improvement of the robustness of classification.

In addition, in (COSTA et al. , 2011) (MOHOTTALA et al. , 2009), the fusion of information was made by correlating the positions of objects detected by the LiDAR in the image coordinates system (matching) in order to study small regions in detail. The LiDAR and camera can capture the same region because of their mounting positions. Mohottala et al. (MOHOTTALA et al. , 2009) observed initially that the segmentation based only on the LiDAR sensor was sometimes not possible. For example, the data obtained from a dark object was not clear. Therefore, in this particular case, they used segmentation based on image processing. It is observed that there is a need for algorithms of recognition in

both image and LiDAR. In this context, an appropriate method to achieve a classification of objects in the image is the SIFT (Scale Invariant Features Transform). According to HU et al. (HU et al. , 2010), this transform can perform a robust classification against changes in objects' scale, rotation, point of view, noise and illumination. Moreover, it is useful for object recognition. The SIFT method is based on the following steps:

1. Detection of scales: The first phase is based on the search of the scales in the entire image in order to get a space of scales. Using a difference-of-Gaussian function, it is possible to identify potential interest points that are invariant to scale and orientation.
2. Location of key points: At each point of interest, the location and scale is determined, and according to its stability, this point becomes the key point.
3. Orientation Assignment: One or more orientations are assigned to each key point based on the directions of the gradient at that point. All future operations are performed on the data that has been processed in orientation, scale and location, thereby providing invariance to these transformations.
4. Descriptor: Gradients are obtained in the detected scale in the region around each key point. These descriptors represent shapes and illumination changes.

In another paper, Ruta et al. (RUTA et al. , 2008) had object detection problems caused by insufficient contrast between objects' boundaries and their backgrounds, especially for objects that appear in shadow or are seen against sunlight. However, Mohottala et al. (MOHOTTALA et al. , 2009) used a technique that can achieve robust segmentation even when the background color distributions are not well separated from the objects. The technique is based on graph-cut and min-cut/max-flow. Thus, this method was used to fuse data from the LiDAR and the camera. Initially, a segmentation based only on LiDAR was achieved, but results were not clear for some objects. Then they used LiDAR data as the initial data for the min-cut/max-flow technique. Therefore, the points referred to the background region and the objects' points were the starting points for the segmentation process. This dissertation will use this technique to segment objects from the background scene.

With regard to pedestrian detection, Geronimo et al. (GERÓNIMO et al. , 2010) performed very thorough research on most of the techniques of detection and pedestrian recognition. According to the authors, there are several challenges in this area. Firstly, the appearance of a pedestrian has a high variability in size, color and shape. Additionally, the pedestrian must be detected in an area with many other similar objects in the background, some of which may even conceal the pedestrian. Geronimo et al. (GERÓNIMO et al. , 2010) analyzed the cases of sensor fusion, and found that sensors based on range detection performed better in adverse weather conditions compared to the use of just a camera. In this context, it is noted again that sensor fusion provides a more robust approach for detecting and recognizing objects.

Finally, the problem of obtaining a strong classification can be understood as a strategy to maximize Detection Rate (DR) as well as minimizing the False Alarm Rate (FAR), as Nashashibi et al. (NASHASHIBI et al. , 2008) explained in their perception system. To maximize DR, objects are detected using the camera and LiDAR. To minimize the FAR, a validation step to measure the confidence of the classification provided by each sensor and for each classification algorithm is necessary. Some algorithms that provide such confidence measurement in classification are the Support Vector Machines (MALDONADO-BASCÓN et al. , 2007) and AdaBoost (BARÓ et al. , 2009) (NASHASHIBI et al. , 2008). This architecture enables adding sensors and algorithms to the system in order to refine classification.

1.5 Problem Statement

The main idea behind Advanced Driver-Assistance Systems (ADAS) and autonomous vehicles is to eliminate human error in driving by taking the control of the vehicle. Because human error is the main cause of car accidents, it is expected that the number of casualties and injuries due to these accidents will be reduced by giving control to computers.

Making an ADAS involves several tasks. Figure 1.6 shows the dynamics of those tasks. The first requirement to navigate in any environment is to be able to sense it, so several sensors enable vehicle's environmental perception. Cognition tasks make decisions about

Figure 1.6: *Gisa*'s operation basis.

the actions the vehicle should take in order to get to any specific point or to avoid collision. Finally, there should be some control systems and their actuator for issuing the decisions coming from the cognition tasks.

The *SENA* project aims to implement a fully autonomous car in the future. In the meanwhile, basic modules are being developed for getting a perception, cognition and control system in the car. *GISA* is actually a platform for developing advanced security systems looking forward to implementing autonomous vehicles. Therefore, in the path to make this kind of system, this dissertation focuses on perception tasks, such as recognizing the road and obstacles.

Concerning restrictions, due to the large number of possible obstacles around the vehicle, and the fact that it is difficult to recognize all of them, this dissertation concentrates on pedestrians, cars and stakes. Several simulation and tests are shown especially involving pedestrians, as they are considered the most important objects to be recognized in the roads. A great part of this research was also dedicated to road estimation. In the future, this perception system for pedestrian detection will hopefully bring reduction in

casualties as it is used together with an advanced security system in the car.

The sensors for the perception system are the Velodyne 360 and Stingray camera (ALLIED VISION TECHNOLOGIES , 2014). Technical specifications are in Appendix A. Velodyne is able to sense around the entire vehicle, but the camera only has 30.2° HFoV, which means this prototype is mainly intended to detect pedestrians in the road in front the vehicle. The main reason for using two different sensors is to take advantage of the data fusion to improve recognition. Figure 1.4 shows the mounting locations for these sensors.

Fusion is depicted in Figure 1.7. The Velodyne sensor, which is LiDAR, determines the position of objects in the vehicle's front on the basis of a distance measurement. It is calculated measuring the time that the light takes to return after being emitted by the sensor. Light is reflected by any object in front within a certain range. For Velodyne HD-32, the range is about 200 meters. The first step in fusion is object detection using LiDAR. This sensor provides spatial position of objects because it issues their distance and bearing with respect to the sensor origin, or, in other words, with respect to the vehicle. This sensor is very good at detecting objects, except for those that absorb light, like dark objects. The second step is to pre-classify the objects detected by the LiDAR. This dissertation gave more importance to pedestrian detection so that the classification is biased for pedestrians. That is, there will be more detected pedestrian objects, even though all of them are not actually pedestrians. Nevertheless, cars, stakes and the road itself are pre-classified by the LiDAR as well. The third step is to crop the section in the image where those objects are detected, with the purpose of using computer vision to confirm the hypothesis provided by the LiDAR. It is expected that by implementing image classifiers with better precision, the final detection system will be more reliable. Therefore, the last step is to perform the classification in the cropped image using a classifier trained previously. In summary, this approach is based on a hypothesis generation of objects using LiDAR, and hypothesis validation using computer vision.

This dissertation mainly tested pre-recorded data from sensors. The data-gathering was accomplished in city scenarios with several pedestrians and cars. A small database with several scenarios was created in order to fine tune and develop the algorithms for

Figure 1.7: Object detection scheme based on data-fusion.

the perception tasks. For data gathering, the next steps were taken: sensor installation, sensor configuration, sensor intrinsic calibration, extrinsic calibration between the camera and the LiDAR, sensor data synchronization, and data storage.

The main objective of this dissertation is to implement some perception tasks for an ADAS and for a potential autonomous vehicle. Thus, perception tasks provide object detection and classification of pedestrians, cars, stakes and the road. Moreover, they provide the position and scale of such objects in the environment. An estimated area limited by sidewalks and obstacles is provided for the road. Road estimation is useful to tell whether a pedestrian is on the road or not and also to know whether it is necessary to take any action to avoid injuring him.

1.6 Document Outline

This document initially presents the object detection and classification using each sensor alone. Afterwards, the fusion approach is presented, allowing the reader to establish the advantages of using fusion compared with using only one sensor. **Chapter 2** presents the machine learning techniques used for classification, and some descriptors used for object recognition. **Chapter 3** presents object detection and classification using the laser sensor. **Chapter 4** presents the computer vision techniques and a *ceiling analysis* of object detection. This analysis yields a way of knowing which path to improve object detection using sensor fusion is the best. **Chapter 5** presents the data-fusion approach in detail. **Chapter 6** presents the software implementation. **Chapter 7** presents several experimental results. The final conclusions are presented in **Chapter 8**.

Chapter 2

Machine Learning Methods

In the task of detecting pedestrians and objects in the vehicle's front, some algorithms are implemented for predicting the class of such objects. This process can be depicted in some stages. (See Figure 2.1.) First of all, to enable the learning more efficiently some descriptors, are extracted from the images. Descriptors provide a way of classifying the classes more easily because they “describe” particular characteristics of objects, which means such descriptors allow the classifier recognize a class among others. Representing the classes can be followed with either a multi-class or single-class approach. For the latter, a database is created with positives training examples that belong to the class to be detected, and negative training examples that represent any other class. The ultimate goal for the classifier is to represent the entire universe of both positive and negative examples in a model. Such a model is based on the type of the classifier (e.g. Support Vector Machines), and its parameters are determined by using a mathematical optimization. The main idea is to separate most of the training examples into two groups. That is why this model is also called a binary classifier. For supervised classifiers, this separation is accomplished through a learning process. The learning consists of an iterative process intended to determine the model's parameters that provide the best separation of the classes. Finally, this model is consequently used for determining the class of unknown samples, a process called prediction (HAYKIN , 1999).

These methods are the core for making prediction on new, unknown data. Supervised classifiers are the most widely-used machine learning models for predictions. Descriptors

Figure 2.1: Machine Learning Modeling.

and the methods for extracting them also fall into the machine learning field.

2.1 Machine Learning on Computer Vision

In computer vision, a methodology for recognizing objects in images can be followed. The following steps were carried out for each object class.

1. Image normalization - This process focuses on specific regions.
2. Extraction of object descriptors - They are the object characteristics.
3. Creation of a model that represents such descriptors (i.e., training a classifier).
4. Model validation using ground-true examples.

2.1.1 Image Normalization

The normalization aims to accomplish image resizing in order to make the target object most of the image. In addition to this, it tries to make the object the center of the image. A better classifier performance is achieved using normalization because the positive descriptors become more similar.

2.1.2 Histogram of Oriented Gradients (HOG)

This descriptor created by Dalal (DALAL AND TRIGGS , 2005) is invariant to local geometric and photometric transformations. The HOG is widely used in object recognition, especially in pedestrian recognition (ENZWEILER AND GAVRILA , 2009). This algorithm extracts local features from the image by calculating the distribution of local intensity gradients or edge directions. It has the advantage of doing a local contrast normalization to reduce non-uniform illumination. The idea behind this descriptor is that the object's local appearance and shape can be characterized very well by local edge directions.

Figure 2.2 explains the static HOG feature extraction chain. This is evaluated on a grid of normalized local histograms, which represents the distribution of the gradient orientations over small parts of the image. The steps are the following:

- i. First, a global image normalization (equalization) is used to reduce non-uniform illumination. An *HOG* descriptor uses gamma (power law) compression, either computing the square root or the log of each color channel. Texture image descriptors are commonly proportional to the local illumination, so this compression reduces the effects of local shadowing and non-uniform illumination in the final result.
- ii. Local image gradients are computed using first-order derivatives. They describe object's contour, silhouette and texture. The locally-dominant color channel is used, which provides color invariance in a better way.
- iii. Encode the local image content, which makes the descriptor less invariant to small changes in pose or appearance. It is encoded using the local gradient orientation

Figure 2.2: Steps to obtain the *HOG* descriptor (DALAL AND TRIGGS , 2005).

information, just like the *SIFT* feature. The image is split into smaller regions, i.e. “cells.” For each cell, and for all the pixels in the cell, the gradients are accumulated in a 1-D histogram. This cell-based histogram is the basic “orientation histogram” representation. Each orientation histogram divides the gradient angle according to the number of “bins” previously set. The gradient magnitudes of the pixels in the cell are used to vote into the histogram. For each orientation in the histogram of orientations, its frequency is defined for the gradient magnitudes, so the more pixels and the bigger their gradient magnitudes in a specific orientation, the larger its frequency.

- iv. Local groups of cells are normalized. It is performed by accumulating a measure

of local histogram “energy” over local groups of cells that are called “blocks.” The result is used to normalize each cell in the block. In addition, cells are shared among several blocks, but its normalizations are block-dependent. The cell features are included several times in the final output vector. Even though it is redundant this improves the performance.

- v. Finally, the descriptor takes histograms from all blocks to form the output vector.

2.1.3 Metrics for Model Validation

The performance of a binary classifier is quantified by calculating the number of correct, missing and unexpected predictions, as presented in Table 2.1.

Table 2.1: Confusion matrix. Relationships between the predicted and actual class.

	Actual Class(Expectation)	
Predicted Class (Observation)	TP (True Positive) Correct Result	FP (False Positive) Unexpected result
	FN (False Negative) Missing Result	TN (True Negative) Correct absent of result

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Number\ of\ Samples} \quad (2.1)$$

$$Precision = \frac{True\ Positives}{No.\ of\ Predicted\ positive} \quad (2.2)$$

$$Recall = \frac{True\ Positives}{No.\ of\ Actual\ positive} \quad (2.3)$$

$$F_1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2.4)$$

Figure 2.3: Local oriented gradients for a 16x8-cell grid. HOG descriptor. Gradients vary among examples, which allow this feature to be used as descriptor.

Figure 2.4: Local oriented gradients for a 16x8-cell grid. HOG descriptor for pedestrians.

2.1.4 Tuned Support Vector Machine (SVM)

An SVM is an algorithm that makes predictions on new data by creating a model of known data (training data). When the data set is labeled to a class, the SVM becomes a trained classifier. Basically, the SVM does a mathematical optimization to find the model that best separates the classes. This procedure is carried out by minimizing the cost function that represents the error between the data and the model (NG , 2012b). This cost is shown in (2.6). As a result, the best hyperplane of separation (hypothesis) is determined. In some classification problems the relation between class labels and features is predominantly non-linear, so that, in order to achieve the separation a special non-linear function is used to map the examples to a higher dimension, in which the features are more easily separated. The function that defines how the examples are mapped is known as the “Kernel”, and it can be either non-linear or linear. The radial-based function (RBF) shown in (2.7) is a non-linear function widely used as Kernel. It defines a similarity function between a landmark and all the examples (NG , 2012b). Another important aspect of SVM is that this model defines a margin of separation between the classes, based on support points, as illustrated in Figure 2.5 HAYKIN (1999).

Figure 2.5: SVM margin. This figure shows the hyperplane of separation for two classes. Points over the dotted line are the support points. JAGGI (2008).

This approach is named tuned SVM because it looks for the best Kernel-based SVM parameters that achieve the greatest F_1 score in the prediction. For the SVM with RBF Kernel, the model parameters are C , known as regularization in (2.6), and σ , which is the RBF width, as in (2.7) (HAYKIN , 1999). The following steps are carried out to tune the model (HSU et al. , 2010):

- i. *Feature scaling.* In order for the SVM with RBF Kernel to avoid features in greater numeric ranges dominating those in smaller ranges, a feature scaling is accomplished.

Let us assume that $x_j^{(i)}$ is the value of the feature j in the i^{th} training example. For each j dimension in the feature vector are found the $maxval_j$ and the $minval_j$ in all the training examples. Then, each value $x_j^{(i)}$ is scaled according its dimension j using (2.5).

$$x_j^{(i)} = \frac{x_j^{(i)} - minval_j}{maxval_j - minval_j} \quad (2.5)$$

All $maxval_j$ and $minval_j$ values should be stored for scaling an unlabeled feature vector when the SVM model is used for predicting.

- ii. *Looking for the best C and σ in a coarse grid.* Both parameters determine the model variance and bias. High bias means an underfitted model and high variance means an overfitting model. The final model should manage a good trade-off between bias and variance. The RBF is tuned by the parameter σ , small values of σ increase the model variance, the smaller this parameter the more overfitting the model is. The C parameter in the cost function regularizes the model by underfitting it, small values decrease the model variance. A good combination of these parameters accomplishes the required trade-off. In the coarse grid the parameters vary following the rule 2^x for $x = [-6, 5]$, $x \in \mathbb{Z}$. A vector of tuning values is then used for the grid, the same values are used for both C and σ .

$$J(\theta) = \frac{1}{2} \sum_{j=1}^n \theta_j^2 + \dots$$

$$C \sum_{i=1}^m [y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)})] \quad (2.6)$$

$$f_i = e^{-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}} \quad (2.7)$$

Where:

$J(\theta)$: Cost function

f_i : Similarity function used as Kernel

θ : Parameter Vector of the estimated hypothesis

$l^{(i)}$: Landmark based on example i

$x^{(i)}$: Feature vector for example i

$cost_1$: Cost for positive examples

$cost_0$: Cost for negative examples

n : Number of features

m : Number of examples

For more details about the cost function see the Appendix B.

- iii. *Looking at a fine grid near the previous values.* The finer grid is created by splitting into 10 equally divided parts the region near the value found previously. The finer grid becomes different for each tuning value.

2.1.5 Cross-Validation Method (CV)

In order for the model to be statistically well validated, its performance should be evaluated on “new data” (different from the training set) several times. Otherwise, it shows an over-optimistic result. Thus, the data is commonly split into two sets, namely a training set and a cross-validation set. For one split, a simple validation is carried out by calculating an estimator of risk (e.g. accuracy) on the cross-validation set using the model created with the training set. CV consists of averaging several validations corresponding

to different-data splits (ARLOT AND CELISSE , 2010). Even though this method is intended to establish the model statistical performance, CV is also carried out to find the best model for our data. Several models are created by varying both the model parameters and the training set, and its performance is subsequently evaluated in the cross-validation set. Figure 2.6 shows a flow diagram of tuning using the CV method.

Because the cross-validation set is used to tune the model, another independent data set for correct validation is needed. This additional set used for final validation is called the test set. A model with a high F_1 score (i.e., both high precision and recall), in the cross-validation set is considered a good one. The best model is finally validated on the test set in order to calculate a more realistic estimation. In this work, the CV evaluation is used to tune the model just by looking for the one that better makes predictions.

The type of CV method used is the repeated random sub-sampling validation. In other words, the data is randomly split into training and cross-validation set several times, and for every split, both the model accuracy and F_1 score are calculated, as depicted in Figure 2.7. The values are averaged in order to get the model statistical performance. Because the model performance depends not only on the model parameter but also on the data, a different performance is achieved for each split. A database with most of the scenarios as well as many examples is used for generating a better model.

Figure 2.6: Flow diagram for SVM tuning.

Figure 2.7: Cross-validation cycle.

Chapter 3

Laser Detection and Classification

Data coming from LiDARs is issued as point clouds. A point cloud is a data structure used to represent a collection of points and can be used to represent three-dimensional data. In a 3D point cloud, the points usually represent the X , Y , and Z geometric coordinates of an underlying surface (RUSU AND COUSINS , 2011). Sometimes, data is represented in a polar coordinate system for sensors with only one laser beam, i.e., the distance and the angles of bearing are the parameters in such a representation. For our sensor, Velodyne HD-32, that has 32 laser beams in the vertical field of view, the representation is in spherical coordinates. Any object capable of reflecting light can be detected and represented as a point or several points. For classification, points in the cloud are grouped on the basis of the distance among them; the resulting object can be pre-classified following geometrical consideration, e.g. the orientation of the surface generated by such points. The techniques for regrouping will be presented in the regrouping process section of this chapter.

3.1 Point Cloud Visualization

Figure 3.1 shows a ROS's visualization tool for point clouds (RVIZ). This cloud belongs to Velodyne HD-32 (360°); the vehicle is in the center of the cloud; this cloud is colored according to the θ (azimuth) angle.

Figure 3.1: Point cloud visualization. It is particularly colored according to the azimuth angle. Following the rainbow color pattern, 0° is set to violet color.

3.2 The Point Cloud in the Spherical Coordinates System

Commonly, the point cloud coming from *Velodyne* is organized in an X, Y, Z coordinate system. However, as another way of organizing the data, every point can be converted to spherical coordinates. The advantage of this method is that it is possible to compare the angles of each point in each direction and to structure the data in function of those angles (WEGNER , 2013). The data coming from *Velodyne* has a frequency of 10 Hz (frame rate). However, the scanning rate depends on the horizontal resolution. Since it has 32 lasers beams, the total amount of points for each rotation is around 2,500.

In this system, commonly noted as (r, ϕ, θ) : the radial distance is r , the polar angle is θ , and the azimuthal angle is ϕ . (See Figure 3.2.)

In order to visualize the new point cloud, the points are again arranged in a X, Y, Z

Figure 3.2: Spherical coordinate System. Source: Wikipedia.

system. Figure 3.3 shows the organized cloud representation. The angle ϕ can be associated with one of the 32 laser beams, just by approximation. The vertical angular resolution is 1.33 degrees, and the vertical field of view goes from +10.67 to -30.67 degrees. Therefore, ϕ represents the number of the laser. The point cloud in the original *Velodyne* X, Y, Z format is not sorted by the angle ϕ ; this is because the data package is arranged according to the laser firing, which is unsorted. The firing is unsorted to avoid potential ghosting caused primarily by retro-reflectors in the laser/detector pairs, so that the *Velodyne* has an interleaving firing pattern (VELODYNE, 2013). The polar angle θ can be associated with each sampling of the all 32 lasers beams. The θ resolution is defined by the sampling time of the sensor. According to the device datasheet the total time to fire all 32 lasers is 46.08 picoseconds, which means the θ resolution or, in other words, the horizontal angular resolution is around 0.16 degrees.

The data is arranged in a new schema where the rows represent the sampling of data for each θ , and the columns represents the laser/detector pair (of the sampling data) for each ϕ . This schema is called “organized cloud.” The idea behind this schema is to

Figure 3.3: Organized cloud. (WEGNER , 2013).

calculate the normals. Figure 3.4 presents how the point cloud is organized in order to calculate the normals in it. Figure 3.5 presents how the normal values are arranged in order to describe the surface in each point of the cloud.

3.3 Surface Orientation. The Concept of Normal to the Surface

The normal vector is a vector perpendicular to the surface or plane. The normal vector is commonly denoted $\vec{N}$ or $\vec{n}$. The normal provides information about the surface's orientation, which can be used to describe the surface itself. (See Figure 3.6.) For example, the normal magnitude for a horizontal surface is near one, which would describe a road.

Laser/Sampling Data	$\theta 1$	$\theta 2$	$\theta 3$	θn
$\phi 1$	Point A		Point C	
$\phi 2$				
$\phi 3$		Point B		
ϕm				

Figure 3.4: Organized data schema.

Laser/Sampling Data	$\theta 1$	$\theta 2$	$\theta 3$	θn
$\phi 1$				
$\phi 2$		Normal ABC		
$\phi 3$				
ϕm				

Figure 3.5: Organized scheme for normal values. The normal is calculated based on the figure 3.4.

Conversely, other normal values indicate that the surface probably belongs to object or obstacles. Figure 3.7 shows a surface compound of three points, A , B and C , and its normal vector. The z-normal is the normalized value of the normal vector in the z direction. When the z-normal is equal to one, it means that the surface would be ground-oriented, and probably drivable, like a road. If the value is close to zero, it means that the surface has another orientation, like, for example, a wall or a person.

3.4 Difference of Normals (DoN)

The DoN is used to study the surface around a point. Basically, it makes a comparison of the normals in the neighbor points in order to determine changes in the orientation of the surface. Therefore, it yields a way of detecting edges and boundaries among objects.

Figure 3.6: Green arrows are normals to the road. Red arrows are normal to the cyclist.

There are several methods to calculate the DoN. The method based on support radius makes an estimation of the normal for different radii (or area) around the point. Figure 3.8 shows the normal vector in a point for two radii or areas, one larger than the other. Finally, in the same figure, the DoN reflects the geometry of the surface. This method has some drawbacks in the points that belong to the boundary of the cloud because there is not enough information.

Alternatively, a method based on the point neighbors can deal with the drawback of the support radius. In this method the normals are compared with four of their neighbors around the point. In the previous section, the organized schema presented a matrix with the normal value in each point. To calculate the difference of normals to the surface in

Figure 3.7: Plane and normal vector for Points A, B and C.

Figure 3.8: Difference of normal calculation. (IOANOU AND TAATI , 2012).

a point in the 3D space $\vec{P}_0$, considering $\vec{N}_1, \vec{N}_2, \vec{N}_3, \vec{N}_4$ as the normals to the surface of 4 neighbor points to $\vec{P}_0$ are used the following equations (WEGNER , 2013). First, the mean value of the four normals is calculated using (3.1).

$$\vec{M}_0 = \frac{\sum_{i=1}^4 \vec{N}_i}{4} \quad (3.1)$$

Finally, the difference of normals at the point $\vec{P}_0$ is the sum of the differences of the normals for each neighbor, as presented in Figure 3.8. The norm of this vector describes the changes in the orientation of the surface.

$$D\vec{O}N_0 = \sum_{i=1}^4 (\vec{M}_0 - \vec{N}_i) \quad (3.2)$$

The normal in each point as well as the norm of the DoN describe the surface, so that they are used to segment the points in the cloud.

3.5 Segmentation

The segmentation is basically a process of grouping points in the cloud according to several criteria. The points are initially segmented in small objects, and subsequently, those are joined to the largest objects. The objects to be segmented are mainly the road, cars, pedestrians and stakes. The point's relative position with relation its neighbors is a key point to know if the points belong to the same object. In addition, the normal to the surface in the z direction allow us to associate points, especially for both big objects and the road. Finally, the DoN yields information about edges and small objects, so that we can join points related to sidewalks and pedestrians. In general, if the points are close enough and have a similar normal value, then they probably belong to the same object.

The algorithm searches for objects point by point, making a special search for large normals, which defines most of the time whether the point belongs to the road or not.

3.5.1 Distance Between Two Points

The method used for calculating this distance is based on the fact that the neighbor points have a similar distance to the sensor. Therefore, instead of defining the distance from one point to another based on a reference system, the distance between the points (3.3), is calculated as an approximation to the arc of the circle that has the point $\vec{P}_n$ on its circumference and its origin in the sensor (WEGNER , 2013), as is presented by eq. (3.5). These methods apply specifically to the *Velodyne* sensor, because it has a 360° horizontal field of view. Figure 3.10 shows *Velodyne*'s point distribution and angle resolution.

Figure 3.9: Procedure for regrouping points in objects.

$$d = |\vec{P}_n - P_{n+i}| \quad (3.3)$$

For the nearest point, the distance is approximated by:

$$|\vec{P}_n - P_{n+1}| \approx \frac{2\pi * \rho}{n_{coll}} \quad (3.4)$$

$$d \approx f * i * \frac{2\pi * \rho}{n_{coll}} \quad (3.5)$$

where:

d is the distance between two points

n is the position of a point (the n^{th}) in the cloud

$\vec{P}_n$ is the point in the n^{th} position

ρ is the distance between the sensor and $\vec{P}_n$

Figure 3.10: Velodyne HD-32's vertical and horizontal resolution.

n_{coll} is the number of discrete steps in which the *circumference_line* is divided into, this is defined by the horizontal angular resolution of the sensor.

circumference_line is the circumference of a circle with origin at the sensor and radius ρ

i is the iterator for the relative position of a point on the same *circumference_line* of $\vec{P}_n$

$P_n + i$ is the point in the $n + i$ position

The relative position is relative to the n^{th} position.

Since the points are not perfectly straightened along the line from the sensor to $\vec{P}_n$ (due to the shape of the obstacles), a correction factor f is included.

If the points are in different *circumference_line*, so the (3.5) is extended to (3.6).

Figure 3.11 illustrates geometrically this equation.

$$|\vec{P}_{n,m} - P_{n+i,m+u}| \approx d \approx f * \sqrt{\left(i * \frac{2\pi \rho}{n_{coll}}\right)^2 + \left(u * \frac{\text{opening_angle} * \rho}{n_{lines}}\right)^2} \quad (3.6)$$

where:

Figure 3.11: Distance between two *Velodyne* points.

n_lines is the the number of lines in the point cloud, which in this case are the number of lasers

$\vec{P}_{n,m}$ is the point in the n^{th} line and m^{th} column

$opening_angle$ is the vertical field of view of the sensor. (See Appendix A.)

u is the iterator for the relative position of a point on the same *column* of $\vec{P}_m$

The distance was calculated using $n_lines = 25$, $n_coll = 2,500$.

3.5.2 Regrouping Process

After segmenting the cloud by checking neighbor points, those potential objects are analyzed to group them into a larger object, which may be more similar to the real object.

For each object the following data is calculated:

- Cartesian coordinates of its center (X, Y, Z)
- θ direction and distance to the sensor ρ from object center
- Mean normal in the x,y and z direction

- Index number of the first and last point in the object
- Cartesian coordinates of the corners
- Length, height and form (height/length)
- Distance to the left and right neighbor points
- Name and number assigned to the type of obstacle

Figure 3.12: Resulting segments from initial grouping. Road segments in several colors, except the red ones that are unknown. Color in segments changes on the basis the distance to the vehicle and following the rainbow pattern.

In the car scenario, the largest object to be segmented is the road. However, because its particular characteristics. (E.g. it is commonly flat.), the road segments are grouped using other operations (WEGNER , 2013):

- i. Grouping neighbor road segments that are separated fewer than n points. n is estimated using 3.7. The segments are merged if $n < 10$.

$$n = \text{abs}(\text{Road}[u]_{\text{max_index}} - \text{Road}[i]_{\text{min_index}}) \quad (3.7)$$

Where:

$\text{Road}[i]_{\text{min_index}}$ is the minimal *index* value of the points in the source object.

$\text{Road}[u]_{\text{max_index}}$ is the maximal *index* value of the points in the target object.

The *index* defines the position of any point in the cloud. Indexes are arranged in a way in which the nearest points to the sensor have the fewest indexes, and conversely, the farthest points have the largest indexes.

min_index is the minimal index in the road segment, and max_index is the maximal one. So $\text{min}/\text{max_index}$ determines the boundary points in these segments. u and i are the indexes of the road segments in the vector (or list) of pre-segmented roads. They have values between 0 and the number of detected roads. The use of u and i allows the algorithm to compare road segments among them. Figure 3.12 shows the road segments to be grouped (in several colors). For example, if a first road segment $\text{Road}[15]$ is composed of the following points: 2122 2123 2124 2125 2126, and the second, $\text{Road}[133]$: 3622 3623 3624 ... 3926 3927 3928. So, for the first, $\text{Road}[15]_{\text{min_index}} = 2122$ and $\text{Road}[15]_{\text{max_index}} = 2126$. For the second, $\text{Road}[133]_{\text{min_index}} = 3622$ and $\text{Road}[133]_{\text{max_index}} = 3928$. Therefore, the n value would be $\text{abs}(2126-3622) = 1496$. Which means they are not neighbors because there are 1496 points between these two road segments.

- ii. Grouping road segments with similar orientation in the θ direction. There are a range on θ for grouping segments. However, this range should be change according to ρ . When segments are near the sensor, small variations in θ corresponds to small variations on the distance among segments. However, when the segments are far

from the sensor, small variation on θ corresponds to large distances among segments. So, the criterion to group segments is defined by θ/ρ rather than θ alone.

- iii. Grouping the ones that have more than n/ρ adjacent points together, since the road will have more adjacent points along the range line of a laser beam.
- iv The remaining roads in the front and the back of the car are selected as the actual roads.

For the objects, the regrouping uses the following operations (WEGNER , 2013):

- i. Grouping obstacles with similar orientation in the same plane. Here, the distance to a *target point* is compared to the distance to the *target point's* plane.

The distance between objects' points' centers is:

$$d_{to_target} = |\vec{P}_0 - \vec{M}| \quad (3.8)$$

The distance between the point's center and the *target point's* plane is:

$$d_{projection} = |(\vec{P}_0 - \vec{M}) * \vec{n}| \quad (3.9)$$

Where:

$\vec{n} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ is the normalized vector of normals to the plane

$\vec{P}_0 = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$ is the point at the center of the plane

$\vec{M} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ the point at the center of the other object, which is going to be projected on the plane.

In addition, a tolerance for the distance f is set (0.1), which is used as criterion to group the object segments. So, the factor between distances should meet the following condition for grouping:

$$\frac{d_{projection}}{d_{to_target}} < f \quad (3.10)$$

- ii. Grouping close segments in the $X - Y$ plane (along the $Z - axis$) with similar orientation – for example, the head of a pedestrian with his body. The range is variable according the type of object's size (e.g., for big objects, in about 1.5 meters). Here, the mean of segment points' z-normals is calculated as criterion for merging segments. The closer they are, the more possible it is they belong to the same object.
- iii Grouping segments smaller than their closest adjacent segments – for example the arms of a pedestrian. For segments with a number of points fewer than 10, regrouping happens.

3.6 Classification based on LiDAR

Features related to the geometry of the object segmented were used for classification, such as aspect ratio, height, length, center in z position, size; and other characteristics related to the distance to its neighbors and the distance from the sensor to the object. Table 3.1 shows the setting for each object class. In addition, pedestrians were classified to at least 35 meters from the sensor in order to avoid noisy data.

Regarding the road, it is pre-classified, basically checking the point cloud in the each θ orientation as shown in section 3.5.2.

After classification the x , y and z positions, relative to the sensor, are calculated for visualization. *RVIZ*, a 3D visualization tool for ROS, was used to show the resulting point cloud.

This classifier was evaluated in a sequence of video. The pedestrians were classified correctly in 75% of the cases, cars in 86% and stakes in 76%. When an object was

Table 3.1: Configured parameter for classification.

Class	Position	Aspect tio	Ra-	Height(m)	Min # points	Mean z -normal
Pedestrian	Foreground	1.5-4		0.5-2	10	
Car	Foreground	0.1-1		0.5-2	50	< 0.8
Stakes	Foreground	> 3			10	
Big_obstacle	Background			> 0.5	5	
Small_obstacle	Foreground			> 0.05	5	

classified as pedestrian, 75% of the time it was a wrong classification; for cars, there were not unexpected results; and for stakes wrong classification occurred 15% of the time.

Missing results in pedestrian detection occur because other objects, such as cars and motorcycles, cover the pedestrians. Unexpected results happen because other objects, such as trees and stakes, are very similar to pedestrians. The results show that this classifier has a high recall because a lot of objects are classified as pedestrian, even though they are not pedestrians.

The high recall in pedestrian detection is due basically to inappropriate reconstruction during the segmentation process. When points are near each other and have similar orientation, they are merged, even if there is a hole among them. This condition is necessary in the reconstruction to join the arms and the legs with the body, but, on the other hand, it causes problems in detection.

Chapter 4

Object Classification Based on Computer Vision

4.1 Pedestrian Detection

This section presents a pedestrian recognition algorithm that uses a classifier based on a computer vision technique. This detection uses the machine learning techniques presented in Chapter 2. The F_1 score metric is used as the model performance estimator, and the repeated random sub-sampling validation is used as the cross-validation method. Both of these procedures show a different approach to the problem in relation to previous approaches (DALAL AND TRIGGS , 2005),(BERTOZZI et al. , 2007),(WANG et al. , 2008), (ZHANG et al. , 2010). Commonly, the model performance is validated through the recognition rate, or the accuracy of the prediction. In contrast, this dissertation tunes the RBF-Kernel SVM (with a cross-validation procedure) using the F_1 score metric because it yields a better result in the estimation.

To better understand the reason for using the F_1 score rather than other estimators, such as accuracy, precision or recall; this section illustrates two scenarios in which various pedestrian classifiers should recognize pedestrians and non-pedestrians in the road. Initially, let's consider a scenario in front of the vehicle in which there are 10 pedestrians and one car. Figure 4.1 shows this scenario at the top. Now, let's consider a vehicle moving

ahead with some algorithms for classifying all those obstacles in front. Figure 4.1 shows the results of classifying those objects using two different classifiers, which are depicted on the right and the left side respectively. Objects inside the circle means those were classified as pedestrians. Conversely, object outside the circle were classified as non-pedestrians. At the lower side are shown both the precision and recall results for each classifier. The classifier on the left side missed a pedestrian. That is why its recall is 90%. In addition, it had an unexpected result (i.e. a car classified as pedestrian); therefore, its precision is 90%. In the right-side classifier, the precision is 100% because all the objects classified as pedestrians are actually pedestrians; however, its recall is 20% because there are 8 missed pedestrians.

Figure 4.1: Imaginary scenario with 10 pedestrians and 1 car. It is used for comparing two pedestrian classifiers (left and right side) using precision and recall. Objects inside the circle means those are classified as pedestrian.

In the next scenario, Figure 4.2, the number of pedestrians changed to 7 and the number of cars to 3. Likewise, the figure depicts two classifiers to be compared. It is worth nothing that both classifiers have the same accuracy, which might indicate their performance is the same. However, the left-side classifier's recall is worse than that of the right. If algorithms inside the vehicle are intended to detect pedestrians, then the classifier had better avoid missing pedestrians. Actually, the classifier should have good

recall as well as good precision. The metric that takes into account both precision and recall is the F_1 score. For the current example, the right-side classifier is better than the left-side classifier, even though the accuracy is the same in both. This is the reason to use the F_1 score rather than the accuracy to evaluate the classifiers.

Figure 4.2: Imaginary scenario with 7 pedestrians and 3 cars. It is used for comparing two pedestrian classifiers (left and right side) using precision, recall and F_1 score. Objects inside the circle means those are classified as pedestrian.

4.1.1 Training Data

The images used to train and validate the classifier are from the INRIA database (*Institut National de Recherche en Informatique et en Automatique*)(DALAL AND TRIGGS, 2005), and from the LabRoM database. The database used is composed of more than 5,000 color images downloaded from INRIA. The images were taken in different weather conditions, such as rain and snow/sleet. Different illumination conditions were also used, including some images taken at night. In addition, the human bodies have different poses

(although all are upright), and there are even a few people riding bikes. Several normalized pedestrian images are shown in Figure 4.3. The negative examples were extracted from images in city scenarios with no pedestrians. Several images were randomly cropped from the scenario. The standard image size used for pedestrians was 64 x 128 pixels.

In addition, around 800 images from LabRoM database were selected from videos taken in the city around the EESC campus. There are several pedestrians together, pedestrians in the shadow, crossing streets, with bags and other similar situations. Figure 4.4 shows several positives examples of this database.

Figure 4.3: INRIA's Normalized positive images (DALAL AND TRIGGS , 2005).

4.1.2 Preliminary Tests

In this dissertation some preliminary test were carried out in order to correctly configure each of the steps in the machine learning process. (See Chapter 2.) These tests concerned pedestrian objects, but results can be extended to the other objects in the scene.

The first preliminary test was intended to determine the need of doing image normalization. Let us consider a database with two sets (original INRIA database), namely train and test sets (70% and 30% respectively) without any normalization in the images (i.e, positive examples with pedestrians, but with more objects in the scene). Using those examples, an SVM is trained and evaluated over ground-true labels. Table 4.1 shows those preliminary results. The HOG was used as the feature descriptor. (In this test, the

Figure 4.4: LabRoM’s normalized positive images.

feature vector is of 81 elements.)

When using normalized images, the tests showed a better result. The SVM (Linear Kernel) is still predicting with better accuracy; however, the precision is not good enough (71.2%). Even though the recall is 100%, the F_1 score (83.2%) is affected by the low precision. In this case, all the actual pedestrians are being predicted, but there are a few non-pedestrian data that is erroneously predicted as pedestrian. These values show that accuracy is not always a good estimator of model performance. A better estimator is the F_1 score, because it takes into account the precision as well as the recall. In this way, a better performance is accomplished using an RBF Kernel because the F_1 score is as high as 92.3%.

Unlike Linear Kernel, the RBF Kernel accomplishes better model precision. However, in some cases (tuned SVM with $C = 0.03$), the recall is not satisfactory. By evaluating several model parameter combinations, it is possible to find a better trade-off between precision and recall, such as the tune SVM with $C = 4$. In this case, both metrics are higher than 90%.

Table 4.1: Preliminary test using the INRIA two-set original split.

Classifier Data Description	Settings and	Accuracy in the Training Set	Accuracy in the Test Set	Recall	Precision	F_1 score
SVM (Linear Kernel) in Images without normaliza- tion		85.3%	82.5%			
SVM (RBF Kernel) in Im- ages without normalization		91.9%	74.9%			
SVM (Linear Kernel) in normalized images		96.0%	91.9%	100%	71.2%	83.2%
Tuned SVM ($\sigma = 0.01$, C $= 0.03$) in normalized im- ages		100%	89.7%	85.6%	100%	92.2%
Tuned SVM ($\sigma = 5$, $C =$ 4) in normalized images		94.6%	89.0%	93.6%	91%	92.3%

The tuned SVM with $C = 0.03$ also shows that a precision of 100% (All the images predicted as positive are actually positive.) is achieved when the model totally overfits the data; for this test the accuracy in the training set is 100%. It means the model is correctly predicting actual pedestrians because it created a good separation between classes. However, it cannot accomplish good recall because a few, both pedestrian and non-pedestrian examples were never considered for training. Consequently, both increasing the training data and adding different, new examples will improve the prediction. The following subsection makes a variation on the sets in order to prove it.

Test Using CV Method

In this test the database is split into three sets; 20% of the data is randomly selected for the test set. So, it remains the same in both coarse and finer testing. The other data is consequently used for CV in order to tune the SVM. Figure 4.5 shows this data distribution. Positive samples are pedestrian images (70%), and negative ones are city streets and country images.

Table 4.2 shows the average of both model accuracy and model F_1 score for 10 random sub-sampling validations. These values, greater than 95%, validate the SVM as a good model of the data. In addition, the best model performance in all the validations is

Figure 4.5: Data Distribution used in CV method.

subsequently evaluated in the test set. With this new data distribution the performance is better, and it is even slightly better after doing a fine tuning. Increasing the variance of the training samples gives the classifier an improvement in the accuracy, from 89% to 94.7%, and on the F_1 score, from 92.3% to 96.1%, in independent (unknown) data.

Table 4.2: Tuned SVM performance evaluation with CV method.

Classifier Settings (normalized images)	Averaged values in CV		Results in test set using the best model*			
	Accuracy	F_1 score	Accuracy	Precision	Recall	F_1 score
Coarse Tuned SVM (*best model: $\sigma = 4$, $C = 16$)	94.1%	95.7%	93.9%	97.3%	93.6%	95.4%
Finer Tuned SV (*best model: $\sigma = 4.2$, $C = 11.4$)	95.0%	96.3%	94.7%	95.7%	96.4%	96.1%

The improvement of the prediction accomplished in the CV test shows that an important factor in improving the classifier performance is to have a lot of well-differentiated training examples, provided that the model presents a good trade-off between bias and variance. In this way, the RBF-Kernel SVM used in this approach avoids the high bias because it has a tendency to overfit the data, which also ensures that by increasing the number and the variance of the training examples, the prediction can be improved. On the other hand, the cross-validation procedure used for tuning the model yields a way of avoiding overfitting it, just because we are looking for a good performance on another set

different from the training set (the cross-validation set). Moreover, the model performs well in the test set, which is totally independent. The search for the best F_1 score in the CV also enables the methodology to find a model with the best trade-off between precision and recall. This trade-off is only reached by tuning the model with the F_1 score rather than the accuracy.

Tuning the RBF-Kernel SVM model increases the model performance when compared with a linear-Kernel SVM model, and doing a finer tuning results in an even better performance. The CV test shows that this method has an accuracy of 95% and an F_1 score of 96.3% in the pedestrian recognition task.

Figure 4.6 shows a positive and a negative image of the same scenario, which were correctly predicted by the classifier. This result shows that in Computer Vision the background had no great effect in the classification.

Figure 4.6: (a) Non-pedestrian image. (b) Pedestrian image.

Even though the classification results were satisfactory for many scenarios, using the HOG and the tuned SVM, there are still errors in classification. Tests showed that in actual road scenarios there are variations with regard to the pedestrian's posture, the pedestrian's companions, and the pedestrian's objects that he normally carries, which produce misclassification.

4.2 Car Detection

The HOG descriptor was also used for car detection. Unlike the pedestrian detection for cars, there are three image shapes. Because the car regions cannot always be cropped using a standard shape, such as the rectangle in the pedestrian case, three shapes were selected for this situation: square, vertical-oriented rectangle, and horizontal-oriented rectangle. Both rectangles have a 2.25 ratio between sides. The number of pixels is not actually important in this detection, because the cropped image is first resized to either the square ratio or the 2.25 rectangle ratio. The final shape depends on the original image's ratio. The number of pixels per block (in relation to the HOG descriptor) is adjusted according to the number of pixels in the entire image. Twenty-five blocks are obtained when the image is split in 6 x 6 grid, which represents 36 cells; and for rectangles, 24 are obtained (9 x 4 split, 2.25 ratio). The last elements in the square image are not used to keep the same number of descriptors in training. The formula in (4.1) is used to determine the descriptor length. In this approach, the cells per block are 4, and the bins per cell are 9.

$$HOG\ descriptor\ length = No.\ Blocks * No.\ CellsPerBlock * No.\ BinPerCell \quad (4.1)$$

$$No.\ Blocks = (No.\ Horizontal\ cells - 1) * (No.\ Vertical\ cells - 1) \quad (4.2)$$

See Chapter 2 for details on HOG descriptor.

4.2.1 Car Training Data

The database for car detection was collected from videos taken around the city, and from images taken from Google. This database is composed of 2,300 images. It is balanced between positive and negative examples. Figures 4.7 and 4.8 show some training examples for negatives and positives. Negative images are resized to the nearest shape for car images

before extracting the HOG descriptors. Table 4.3 presents the metrics for the car classifier.

Figure 4.7: Examples of car positive images.

Table 4.3: Metrics calculated for car classifier.

Classifier Settings and Data description	Recall	Precision	F_1 score
Tuned SVM using CV procedure ($C = 22.6$ $\sigma = 0.029$). Descriptor size = 864	98.2%	94.4%	96.3%

Figure 4.8: Examples of car negative images.

4.3 Ceiling Analysis Approach

In order to demonstrate that a fusion approach is more useful for the object detection problem a ceiling analysis was carried out. This was done only for pedestrians, but the results can be extended to the other objects in the scene.

Ceiling analysis is known as a method to estimate which element of a pipeline machine learning system has a strong influence in prediction. Likewise, it allows us to estimate which element has a weak influence, and therefore, to limit the effort to improve its performance as it yields no significant change in the final result. This analysis is associated with the concept of ceiling effect, in which an independent variable no longer has an effect on a dependent variable after reaching a certain level (i.e., the ceiling). In the pipeline, this effect can be seen as the impossibility of any increase in the element's performance (the independent variable) to improve the prediction (the dependent variable) because it has already reached the "ceiling." In solving the machine learning problem, many resources might be spent trying to improve the elements' performance without actually reaching great results in the entire system. There are several reasons to do a ceiling analysis in

any stage of the development. The first is to optimize the resources spent in a task. Another is to modify the right processing element in order to actually get an upside in the prediction. The upside is a percentage (numeric value) of improvement. The last is to determine the benefits of a proposed method before developing it to work automatically. In order to determine the potential upside that might be provided by each processing element, a simulation of each one in its ideal performance is carried out. Thus, a metric is used, such as prediction accuracy, to determine the upside in the performance of the entire system when each element is simulated. To do that simulation, the test set is manipulated according to the task of the processing element. In this way, the pipeline is provided with ground-truth labels in each stage. Figure 4.9 presents an example of a machine learning pipeline of three elements and the simulation flow. The main idea behind this procedure is to simulate elements with 100% accuracy, and therefore, to hopefully increase the performance of the entire system. The simulation is done in a consecutive way. That is, simulated elements are added from left to right until completing the whole pipeline. This simulation is presented by Andrew Ng at NG (2012).

The advantage of this method is that it allows determining the most promising pipeline's elements to be modified as a way of more efficiently improving the recognition system.

Figure 4.9: Machine learning pipeline.

The pipeline presented in Figure 4.10 has been proposed in order to determine the potential upsides. These processing elements are the base for many systems, such as those implemented on (OLIVEIRA et al. , 2010)(DOLLAR et al. , 2012)(BESBES et al. , 2010). However, the pipeline is not limited to these elements, and each one might be split into several sub-elements.

Figure 4.10: Pedestrian recognition pipeline.

There are four processing elements in this approach: image pre-processing element, pedestrian descriptor extractor, classifier, and training. The first basically extracts a smaller part of the image in which it is more likely to find a pedestrian because it is easier to recognize the pedestrian when most of the image is only a pedestrian. The next one, the descriptor extractor, is the element that provides the features to the classifier. We have used HOG as descriptor; this descriptor yields features more easily recognized when the image is pre-processed because the features are more similar in images that have only pedestrians (i.e., normalized images).

There is also a training element connected to the classifier, as is presented in Figure 4.10, which generates the classifier's model using a supervised learning. This element establishes another line in the pipeline, which will be called the training line. The classifier is the last element in the pipeline and it determines whether the example is a pedestrian or not. Its performance depends on two lines, namely the training line and the processing line. Each line contributes in an independent way to the performance of the system, but they actually use similar methods. The training line implements image normalization and features extraction into the training data the same way the processing line implements these methods into the test data. Indeed, both lines use the same pedestrian descriptor extractor, which was HOG. Even though the image normalization and features extractor methods are in both lines, they will only be simulated in the processing line in this analy-

sis. The training element will be simulated in the training line, which basically simulates the ideal training method considering the ideal training data.

4.3.1 Ceiling Analysis Simulation

The simulation has several stages as described in Figure 4.11. The element that receives the test data is simulated in the first stage. In the following stages, the next elements are added and, it is worth noting that the elements already simulated continue to be simulated. There is actually no need to simulate the last element; it is assumed that it should provide the remaining upside to achieve an accuracy of 100%.

Figure 4.11: Machine learning pipeline.

The main idea in this analysis is to create a table that specifies the upside in each stage due to the current processing element simulated. An approach to this simulation is presented in Table 4.4. The first row shows the system accuracy without any simulation. It is the starting point to calculate the upside for the first element. The upside is the

result of the subtraction of the current improved accuracy from the previous accuracy. In the end, the simulation will show which element provides the largest upside.

Table 4.4: Simulation Table for Ceiling Analysis.

Stage	Element	Accuracy	Upside
No simulation	-	Current(%)	-
1st	1st	Improved (%)	(%)
(n-1) stage	(n-1)	Improved (%)	(%)
Last (n)	Last (n)	Expected (100%)	(%)

The ceiling analysis may be seen as a more reliable way for deciding in which element to put a focused effort because it makes sure that improving that element will actually have a better effect on the performance of the entire system.

4.3.2 Image Pre-Processing Element

In the processing line, this pre-processing aims to select a smaller part of the image where it is likely to find a pedestrian. There are two well-known approaches to this procedure. The first is looking for pedestrians in the whole image by sliding a search window through it. The second is an approach to regions of interest (RoI), which is based on the data fusion of LiDAR and camera. The laser sensor provides obstacle detection and obstacle's spatial position in the front. This spatial position on the laser's coordinate system is related to the camera's coordinate system in order to find the region of the obstacle in the image. The RoI approach has some advantages over the search window procedure. The first is that the processing time is several times shorter than in the search window. This is because in the search case, the window has to be slid through the whole image and resized to look for pedestrians in different scales, which requires more processing time for all the windows. Unlike the search window, the data fusion approach tries to find RoI using the laser so as to analyse a smaller part of the image, which reduces the processing time. Another advantage of RoI is that the number of false positives (FPs) is several times smaller than in the search window because it is more likely to detect actual obstacles with the laser sensor. Monteiro et al. analysed these elements at (MONTEIRO et al. , 2006).

In the training line, there is also a pre-processing procedure to normalize the image. This normalization aims to ensure that the image used for training is mainly pedestrian. In most of the cases, an expert selects a smaller part of the main image. Unlike the training line, in the processing line, the normalization should automatically be carried out in the elements.

4.3.3 Simulation Data

This simulation was carried out in an SVM with linear Kernel over a database compound of examples from INRIA, LabRoM and the Internet.

Table 4.5: Metrics calculated for various classifier settings and data.

Classifier Settings and Data description	Accuracy in Test Set	Recall	Precision	F_1 score
Linear SVM in Images without normalization	82.5%	-	-	-
Linear SVM ($C = 1$ by default) in normalized images	91.5%	90.6%	97.3%	93.8%
Tuned Linear SVM ($C = 3.2$) in normalized images	93.5%	92.9%	97.9%	95.3%
Tuned Linear SVM using CV procedure in normalized images	95.6%	95.7%	97.7%	96.7%

Table 4.5 presents the fact that the prediction increased from 82.5% to 91.5% when normalized images were used. In addition, by doing fine tuning, the accuracy reached 93.5%.

4.3.4 Ceiling Analysis

In the ceiling analysis, several simulations are carried out following the simulation flow. For each simulation, the prediction accuracy is determined and used to calculate the potential upside due to the current element simulated (NG , 2012). In this section, how the recognition pipeline elements are simulated is explained.

First, the pre-processing element is simulated by normalizing the images of the test set. The INRIA database has already normalized the images on that set, so it already yields a way of simulating the element for the whole data set.

Regarding the pedestrian descriptor extractor, its suitable simulation must meet some

conditions: to provide the classifier with enough features that are largely similar for examples of the same class and largely different for examples of different classes. It depends on the extractor method itself to meet these conditions; however, the extractor method can be sufficient for a simple database, but for a more complex one, it may not. For example, the HOG pedestrian extractor was used in the MIT pedestrian database, in which it provided the classifier with good features so that all the test images were correctly predicted (DALAL AND TRIGGS , 2005). On the other hand, for a more complex database, such as the INRIA, the HOG descriptor is seemingly not enough because we could not reach perfect recognition using the same classifier. Consequently, the descriptor should be adjusted according to the scenarios of the test. Other kinds of descriptors could be tested, but this is not the aim of this work. This analysis is trying to find out what descriptor extractor element makes in the entire system in order to simulate it.

The descriptor helps to make the task of the classifier simpler because, if the features among classes are largely different, they are then more easily separable. According to the machine learning theory, there are some feature manipulations that help the classifier create a better model. For example, by mapping the current features to a higher dimension, in which they are more easily separated (such as the case of SVM), the classifier's performance is improved (NG , 2012). Another way is to perform a feature scaling before tuning the classifier's model, such as mentioned in the tuned SVM section. These procedures show that the classifier can modify the features in order to improve its model. Consequently, it is difficult to consider the descriptor extractor as an independent variable to be simulated because the classifier usually does some additional processing on the features. Some tests intended to evaluate several pedestrian descriptors were able to quantify their performance only after the classifier by determining performance in the final prediction (GERÓNIMO et al. , 2010). Therefore, there is a close relationship between these processing elements, which makes it difficult to address them separately. This dissertation does not do a simulation of the descriptor extractor as such; any potential upside provided by the descriptor extractor will be jointly considered when the upside for the classifier is found.

With regard to the training element, there are basically two simulations. The first is the simulation of a larger distribution of training examples (i.e., examples with more

differences among them). The idea behind having a larger distribution is to cover a larger quantity of data to be predicted. In a recognition problem, the larger the number of scenarios to be predicted, the larger the number of different training examples. To achieve a good pedestrian recognition system in an autonomous car, the training examples should have as many scenarios of pedestrians in the street as possible. However, to conduct our ceiling analysis, we have limited these scenarios to those covered by the INRIA database. Therefore, to simulate this larger distribution, we have randomly added some examples from the test set to the training set, as presented in the CV test section. We have assumed that if there are still some test data that have not been recognized correctly, it is because they have not been included as examples in the training. In this way, to randomly add data to the training set will generate a better training if the data added are new ones. In summary, the larger distribution was simulated by gathering the whole data from INRIA database and creating new sets, and subsequently using the CV procedure in these sets to improve the trained model.

The second simulation in the training element is the training method itself. The most important factor is how the hypothesis created after training will generalize new data. Therefore, the right parameter to determine how well the model was trained is the generalization error (i.e., the error in the unseen data). The smallest generalization error is found when the model has the right trade-off between bias and variance, which basically depends on the model itself and its regularization procedure. As shown in the classifier section, there is a regularization parameter in the model that can be tuned to achieve a better regularization. This dissertation considers the model tuning as a way of achieving the best training method for the SVM classifier.

Seemingly, the best training method could be the one in which all the examples are correctly predicted after training. However, it is well known in machine learning that a method that exactly fits the training set does not ensure it will predict new data well.

The classifier is the last element of the pipeline and, therefore, it does not need simulation itself; it is assumed that if a perfect prediction is required the classifier should provide the last upside to achieve accuracy of 100%. In this dissertation, the classifier's model is the method that determines the performance of this element; we have used the

SVM with linear Kernel as the model.

Table 4.6: Estimation of potential upside due to each element.

Simulated Element	Variable	Method	Accuracy	Potential Upside
		HOG without normalization	82.5%	
Pre-processing	ROI	Image Normalization	91.5%	9.0%
Descriptor extractor	Descriptor	HOG	*	*
Training	Training Method	Tuned SVM	93.5%	2.0%
Training	Data	New sets and CV procedure	95.6%	2.1%
Classifier	Classifier's model	SVM with linear Kernel	100.0%	* 4.4%

Table 4.6 shows the processing elements, the independent variables that define the prediction, the method associated with that variable, the prediction result after adding that element in the simulated pipeline, and, finally, the potential upside due to that simulation. The first row shows the accuracy of the overall system without any simulation. This test used examples without normalization taken from the INRIA database; the test was done with the original sets. The results show that the largest potential upside is due to the image normalization (9.0%). The training element, taking into account its two simulated variables, shows a potential upside of 4.1%. Both classifier and descriptor extractor together must have a potential upside of 4.4%.

Figure 4.12: Ceiling analysis results.

According to our analysis, to select a smaller image where it is likely to find a pedestrian may be a more promising method to improve our system. As mentioned in the pre-processing method, the RoI approach has a few advantages over the sliding window.

Because many approaches to autonomous cars already have a LiDAR and a camera. This method is a good option to take advantage of both sensors in the car.

4.4 The Fusion Approach

The ceiling analysis showed that the pre-processing element of our pipeline is the one with the largest potential upside when it comes to improving the prediction, which means it is more worthy to work in this element as it yields a much larger improvement in the recognition system, namely, the image normalization in the processing line. Figure 4.12 shows how the processing elements should be modified in order to obtain a better performance in the prediction. The data fusion and the improvement of the training examples are considered promising approaches.

An automatic method for normalizing the images was developed. This is based on a data fusion technique of a CCD camera (AVT Stingray 033C) and a frontal, multilayer LiDAR sensor (Velodyne 32-HDL) installed in the prototype vehicle. (See Appendix A for devices specifications.) This LiDAR sensor yields the obstacle's location in the front. By making a correlation between both data, the obstacle's spatial location found through the laser is subsequently used to get another small image with only the obstacle from the entire image (taken from the front). As a result, an obstacle's normalized image was obtained.

Chapter 5

Data-Fusion Classification

The data fusion is carried out as a method of hypothesis generation and validation (COSTA et al. , 2011). This approach takes the pre-classified objects in the point cloud as hypothesis to subsequently validate them in the classification of computer vision. The pre-classification has a high recall to avoid missing detections, and the vision is balanced by tuning the classifier with the F_1 score. (See Section 2.1.3.)

5.1 LiDAR Data and Image Overlaying

In order to fuse the LiDAR data with the image of the camera positioned on the front of the car, the classified points coming from the LiDAR are superimposed on the image. This enables the algorithm to find regions in the image where there are likely cars, pedestrians and other objects. Then, images for each type of object are cropped and subsequently classified using image processing. As a result, the classification is improved.

5.1.1 Synchronization Policy

To overlay the LiDAR data and the image, both data should be synchronized. This is necessary because the frame rate in the LiDAR is $10Hz$, while in the camera is $25Hz$. Moreover, the classified point cloud is published at about $8-9 Hz$. Therefore, the entire data fusion should be synchronized at this low rate. Consequently, several image frames

Figure 5.1: Image overlaid with the classified point cloud.

are deprecated. The synchronization policy waits for the next image frame just after a classified point cloud is processed, which reduces the data-fusion task rate a little more.

An approximate time synchronization policy was implemented to look for an image frame every time a pre-classified message arrives. The fusion algorithm looks for the nearest time stamps among the frames to find the best complementary image in the queue.

5.1.2 Fusion Process

In order to superimpose both data correctly, the translation, rotation and opening angles of the camera with regard to the Velodyne sensor are measured:

Figure 5.2: Image with objects boxes. Blue ones are cars, green ones are stakes and red ones are pedestrians. White ones correspond to actually the regions defined for the points. Color boxes are bigger because actual objects are supposed to be larger than the region defined by the points.

Parameter	Description
-----------	-------------

x	Relative position of the camera to the Velodyne sensor in the x-axis
y	Relative position of the camera to the Velodyne sensor in the y-axis
z	Relative position of the camera to the Velodyne sensor in the z-axis
α	Rotation around the x-axis to align the camera to the Velodyne sensor
β	Rotation around the y-axis to align the camera to the Velodyne sensor
β_{openi}	Opening angle of the camera in the y-axis
γ	Rotation around the z-axis to align the camera to the Velodyne sensor
γ_{openi}	Opening angle of the camera in the z-axis

Then, the translation vector is built (RODRIGUEZ , 2008):

$$Trans = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (5.1)$$

The rotation matrix as following:

$$Rot_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{pmatrix} Rot_\beta = \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix} Rot_\gamma = \begin{pmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.2)$$

Finally, this procedure computes the new coordinates for each point:

$$Pos_{camera}^{\vec{}} = (Pos_{velodyne}^{\vec{}} - Trans^{\vec{}}) * Rot_\alpha * Rot_\beta * Rot_\gamma \quad (5.3)$$

Because of the limitations of the horizontal opening angle of the camera, only the points in the front are rotated and translated. For visualization, points are drawn according to their distances from the sensor (Figure 5.1); the nearer they are the bigger they are visualized; in addition, a color corresponds to each type.

5.1.3 Image Cropping

Cropping is the process of taking the “object image” from the entire image. Figure 5.2 show smaller, gray boxes inside the color object boxes. Gray boxes surround the original points for specific objects. They are not straight because points are basically clutter. Color boxes are obtained by looking the straight boxes that better fit those gray boxes, so that they are bigger than gray ones. In addition, color boxes are adjusted in size to meet the normal object dimensions. For example, for pedestrians, the boxes are adjusted to be vertical-oriented rectangles with a 2.25 ratio; for cars the boxes are adjusted to be squares or rectangles depending on the original gray boxes ratios. (See Section 4.2 for details.) Moreover, box size is also adjusted according to the object distance in the scene (distance provided by the LiDAR).

5.2 Object Classification Schema

Basically, after cropping the object, the hypothesis (LiDAR pre-classification) is validated using a computer vision classifier. Figure 5.3 shows the pedestrian detection scheme. This scheme is the same for cars.

5.3 Road Estimation

Road estimation is carried out using the assumption that road pixels are connected into the biggest traversable area. This area was determined in the point cloud by looking for the whole surface with horizontal orientation. Conversely, other points in the cloud are assigned to non-road areas. For a pixel that does not have an assigned class, the watershed algorithm is used to estimate whether it belongs to the road class or not. (ZHOU AND IAGNEMMA , 2010).

5.3.1 Road Estimation using Watershed Segmentation

The watershed implemented is a modification of the watershed based on basins. It is a marked-based segmentation algorithm. In this case the flooding process starts from a predefined set of marked pixels. The basins are labeled in accordance with the values assigned to the initial marks. When two basins have the same label, they are merged, and no additional watersheds are created, thus preventing the over-segmentation (LAGANI , 2011).

For road estimation, morphological operations and watershed are used. First, the resulting merged image showed in Figure 5.4a is used to know which pixel represents a pre-classified point. The segmentation was initially carried out using the flood-fill algorithm. The seed was set according to the value of the cells below the image, which is supposed to be the road in front of the vehicle. This assumption worked for most of the cases, but in some particular cases it was not valid, like when the vehicle was turning. A drawback of the flood-fill algorithm is that it sometimes cannot reach the entire road because some noise in the middle of the road makes the flood stop. Therefore, the flood-

Figure 5.3: Pedestrian detection fusion scheme.

(a) Pre-classified road points in green emerald.

(b) Mask for the watershed.

(c) White region is the estimated road.

(d) Fusion result.

Figure 5.4: Road estimation using watershed segmentation

fill algorithm can erroneously leave some road regions out because they are not reached by the flooding from the front of the vehicle. When only this operation is used for segmenting the road and non-road scenario, there will be a significant amount of wrong information fed back to the classifier. For this reason, the flood-fill algorithm was not suited to this problem. To make sure that most of the possible road regions are segmented correctly, the watershed algorithm was used instead of the flood-filling. In this case, even when some noisy information is in the middle of the road, the regions pre-classified as road will be segmented as road and, subsequently, by using morphological operations, they will be more easily connected.

One advantage of the watershed is that one can use ground-true labels as the starting point to define the regions. When the label of one pixel is uncertain, it is specified to the algorithm that that particular pixel should be segmented. In order for the watershed to determine which pixels should be classified, a mask is defined, in which a label for the known pixels and zero for the unknown ones is set. Figure 5.4b shows a mask with three labels (white, dark gray and light gray). Labels were assigned according to the pre-classified object. For some pixels in the image, there is a high probability that they are actually a road, such as the pixels pre-classified as road after the sensor fusion. Other pixels pre-classified as car and/or big objects were marked as not road because they likely are obstacles. In addition, there is another label for small objects. For the other pixels, the value in the mask image is set to zero.

Figure 5.4c shows the estimated, segmented regions. Morphological closing was applied to the image to reduce small holes in the regions, and morphological opening to improve boundary definition. Even though the original image is gray scale and not binary, the resulting image has a better uniformity in the gray scales values. This post-processing allows the segmentation algorithm to more easily define the regions. Figure 5.4d shows the final segmentation over the image.

5.4 Foreground Extraction Using Graph-Cut Algorithm

A foreground object refers mainly to an obstacle of interest in the scene. Conversely, the background of the image refers to all pixels back in the scene. The process of separating them is considered an image segmentation process. The graph-cut algorithm implements a graph over the image; the nodes are the pixels and the edges among nodes are defined by a similarity rule among pixels. Therefore, for segmenting the image, an optimization problem over the graph is solved using the min-cut/max-flow algorithm (BOYKOV AND KOLMOGOROV , 2004) (MOHOTTALA et al. , 2009).

In order for this algorithm to segment the image just labeling some of the image pixels as belonging to background, foreground, possible background and possible foreground is needed. In this dissertation, those labels were taken from the pre-classified data given by the LiDAR. Based on this partial labeling, the algorithm will then determine a foreground/background segmentation for the whole image. The label is set in a mask image with the same dimensions as the original image. There are two ways of specifying a partial foreground/background labeling for an input image. First, this can be done by defining a rectangle in which the object's possible foreground is included. All pixels outside of this rectangle will be then marked as possible background. Second, this can be accomplished by setting labels into the very pixels that belong to the background/foreground, and for the others setting a label of probably belonging to the background/foreground. The final segmentation will assign those latter pixels to background or foreground. Figure 5.5 presents the segmentation process.

Graph-cut is computationally very expensive. For an image of 640 x 480 pixels, the processing time is about 2 seconds for just one iteration. However, it also depends on the labeling. If there are many pixels labeled as probably foreground/background, then the algorithm is going to spend more time segmenting the image.

(a) Classification create a box around the car

(b) Mask.

(c) Resulting segmentation

Figure 5.5: Segmentation using graph-cut algorithm with min-cut/max-flow optimization. Figure (a) shows a blue box that defines a region with potential car. This region was obtained using the pre-classification with LiDAR. Figure (b) is a mask image that describes the pixels in the image according to these rules: light-gray pixels are probably foreground/object, dark-gray pixels are probably background. Balanced-gray pixels are true background, and the white pixels are true foreground/object. In figure (c), colored pixels are the object/foreground extraction.

5.4.1 Graph-Cut Theory

A directed weighted graph consists of a set of nodes V and a set of directed edges E that connect the nodes. For an image, the nodes usually correspond to pixels. A graph normally contains some additional special nodes that are called terminals. Such terminals correspond to the set of labels that can be assigned to pixels. For example, for the foreground extraction problem, there will be two terminals: object and background, respectively. In the image, those terminals are seed pixels; an image of 3 x 3 pixels with its seeds is shown in Figure 5.6(a). The terminals are usually called the source, s , and the sink, t . Figure 5.6(b) shows the graph for the upper image. The figure also shows the source and sink terminal based on the graph of Greig et al. (GREIG et al. , 1989). All edges in the graph are assigned some weight or cost; in the figure edge costs are reflected by their thicknesses. The cost of n-links corresponds to a penalty for discontinuity between the pixels. Typically, penalties indicate individual label-preferences of pixels based on observed intensities and pre-specified likelihood function (BOYKOV AND KOLMOGOROV , 2004b).

Graph-cut is an algorithm for foreground extraction created by Microsoft researchers (ROTHER et al. , 2004). First, a Gaussian Mixture Model(GMM) is used to model the foreground and background using provided labels coming from a mask. Then, a graph is built from this pixel distribution. Pixels are the nodes in the graph. There are two additional nodes in the graph: the source node and the sink node. Every foreground or object pixel is connected to the source node, and conversely, every background pixel is connected to the sink node. The weights of edges connecting pixels to source node/sink node are defined by the probability of a pixel being foreground/background. In addition, the weights among the other pixels are defined by the edge information or pixel similarity. If there is a large difference in pixel color, the edge between them will get a low weight. Finally, the min-cut algorithm is used to segment the graph

5.4.2 Min-Cut and Max-Flow Problem

An $s = t$ cut C on a graph with two terminals is a partitioning of the nodes in the graph into two disjoint subsets S and T such that the source s is in S and the sink t is in T . Figure 5.6(c) shows one example of a cut. The minimum cut problem on a graph is to find a cut that has the minimum cost among all cuts. The cut cost is directed as it sums up weights of directed edges specifically from S to T . The minimum $s = t$ cut problem can be solved by finding a maximum flow from the source s to the sink t . A maximum flow from s to t saturates a set of edges in the graph dividing the nodes into two disjoint parts, corresponding to a minimum cut. It is worth of noting that any cut corresponds to some assignment of pixels (nodes) to labels (terminals) (BOYKOV AND KOLMOGOROV , 2004b).

In other words, based on a label mask image or an image with seeds, the algorithm groups pixels into clusters of similar colors, i.e., K clusters for the background and K clusters for the foreground. Then some boundaries between foreground and background pixels are set. The segmentation is an optimization process that tries to connect pixels with similar labels, so that a penalty is set for placing a boundary in regions of relatively uniform intensity. This problem is efficiently solved using the Graph-Cut algorithm in pixels that belongs to the connected graph in which *cuts* are applied for composing an optimal configuration.

Figure 5.6: Example of a directed capacitated graph with two terminals in an image of 3 x 3 pixels. Edge costs are reflected by their thicknesses. The figure describes the process of labelling an image, creating the graph, and then cutting the graph to produce a segmentation of the image.(BOYKOV AND KOLMOGOROV , 2004).

Chapter 6

Software Implementation

The software implemented is based on the Object Oriented Programming paradigm. Therefore, objects in the real world were abstracted in the software. It was deployed over Linux OS (Ubuntu). For multi-task processing and data synchronization, ROS middle-ware was used.

6.1 Middle-Ware Platform: Robot Operating System (ROS)

ROS is an open-source robot operating system that has been widely used for universities to develop robotics projects. In recent years, several companies have been using it in industrial application as well. It includes services, hardware abstraction, low-level device control, implementation of message-passing between processes, and package management (www.ros.org). One of its advantages is its versatility to integrate pre-developed code in new applications, so there are already several packages available, from hardware drivers to high level functionality. The message-passing functionality also provides a way of splitting the processing load in several machines. ROS architecture is based on *nodes*, which are independent applications (.exe files) that distribute processing tasks. In addition, the communication among nodes is made using message-passing or *topics*. ROS is implemented on C++ (and Python) so that the programing is basically object-oriented

(MARTINEZ AND FERNANDEZ , 2013)(KANE , 2013).

6.2 Software Description

This dissertation implemented around 20 nodes over the ROS architecture (*fuerte* version) and 4 additional classes. In general, nodes were divided into data acquisition (raw data), pre-processing, training, prediction and visualization. For running all nodes at once several *launch* files were configured. They are written in *.xml* standard and they specify settings for nodes. Several 3rd-party libraries were included in the project; for example: *libsvm*(CHANG AND LIN , 2011) for machine learning; *filesystem* for data training administration; *OpenCV* for computer vision; *boost* for data managing; and *firewire* drivers for the camera and additional drivers for *Velodyne* sensor. The code was grouped into two ROS packages, one for machine learning processing and other for image and LiDAR processing. Several configuration files are stored inside the directory of the package so that running parameters, such as sensor calibrations and classifier settings, can be retrieved at start-up.

The software is obstacle-oriented, so that in each frame the obstacles are pre-classified in the point cloud. Then all those obstacles are cropped from the image, taking advantage of the fusion, to classify them using Computer Vision. Five to 20 objects are analyzed in each frame. There is a final visualization image that uses the topics from classification nodes to show which part of the image is road, pedestrian, car or stake. These topics publish object information that is passed among nodes, so that the processing task can be split and just the processed data is sent among nodes.

6.3 Data Fusion Synchronization

This synchronization is necessary because the frame rate is different for each sensor. The time policy enables the fusion to process, in an approximate way, the same scene in each sensor. This functionality is provided by ROS and can be extended to any number

of sensors.

- The algorithm subscribes to `/camera/image_raw` and `velodyne_classified` topics using the same *NodeHandle*.
- An approximate time policy `MySyncPolicy` is defined for `image` and `PointCloud2` data types.
- The synchronization policy is applied on both subscribers with a queue size of 20.
- In each synchronization, the synchronizer will call back the data-fusion function.

6.4 ROS Nodes and Topics in the Visualization Graph

Figure 6.1 shows the ROS visualization graph generated by `rqt_graph`. In this graph, ellipses are nodes and rectangles are topics. Topics are generated as objects messages inside ROS. For examples, the topic `/detected_pedestrians` has all pedestrians detected in the node `/crop_image_processor`, with its position and size (described for the topic attributes). The `/detected_pedestrians` topic is published by `/crop_image_processor`, and subscribed by `/fusion_sensor_visualization`. They are public, and any node can subscribe to them. The resulting processing of this stage is available for further development. There are also a lot nodes connected to the `/output` node. All those nodes are publishing images to the screen; for example, `/car_image_segmentation` displays the segmented car after applying foreground extraction in the screen. Figure 6.2 displays specifically the `/fusion_sensor_visualization` node. This node subscribes from three topics: `/estimated_road`, `/detected_pedestrians` (which includes also cars) and `/camera/image_raw`. It publishes to the output image `/fusion_sensor`, which in turns is output to the ROS `/output` general-purpose node. Note that this node does not care about LiDAR data, because LiDAR is already processed and published to the topics. These topics, which have the relevant information about the objects, publish fewer data than the `/velodyne_prepared` LiDAR original topic. As a result, useful data can be sent more easily to other machines in the car network, or they can be sent outside the car for panel visualization. Basically, the data is now about a few kilobytes rather than a few

hundreds megabytes. These examples show the advantages of using topics and how they make useful data more available.

6.5 Hardware - Computer Specifications

Intel Core i7 Fanless Embedded Box PC.

bogomips	: 5066.46
clflush size	: 65
cache_alignment	: 64
address sizes	: 36 bits physical, 48 bits virtual
power management	:
processor	: 3
vendor_id	: Genuine Intel
cpu family	: 6
model	: 37
model name	: Intel(R) Core(TM) i7 CPU E 610 2.53GHz
stepping	: 5
microcode	: 0 × 2
cpu MHz	: 1197.000
cache size	: 4096KB
physical id	: 0
siblings	: 4
core id	: 2
cpu cores	: 2

Table 6.1: Computer Specifications.

Figure 6.1: ROS's Graph Visualization.

Figure 6.2: Fusion sensor visualization node.

Chapter 7

Experimental Results

7.1 Classification Using Only Computer Vision

Figure 7.1 shows a first test using only a computer vision technique. This is the well-known sliding window method. In this method, several windows of different sizes are slid to look for pedestrians in different scales, so it is computationally more expensive. This classification is also based on the HOG descriptor and the SVM model. The database used is inside the OpenCV library. (See www.opencv.org for reference.) For each frame, the processing time goes from 3 seconds to 4 seconds, which is beyond our real-time requirements. The results show that this method has several false positives.

The OpenCV implementation is basically based on (DALAL AND TRIGGS , 2005) work. A database (positives samples as person, negatives samples as non-person) is used to train a linear SVM. Cross-validation is also performed to optimize the soft margin parameter C for a linear Kernel. Then, images are cropped considering several scales or window sizes; 1.05 is the coefficient for window increase. For each scale, features are extracted (using a scale-space pyramid method) and an expensive computational process ensues. The features extracted from HOG detection/window (64×128) gives a total of 3,780 features ($4 \text{ cells} \times 9 \text{ bins} \times 7 \times 15 \text{ blocks} = 3,780$). In summary, the detection window is scanned across the image at all positions and scales, and conventional non-maximum suppression is carried out on the output pyramid to detect object instances.

(a) False positive classification

(b) False positive in a car

(c) Pedestrian classification in several scales

(d) Classification in a group of people

Figure 7.1: Pedestrian classification using sliding-window method.

Figure 7.2: Classification using only LiDAR data.

7.2 Classification Using Only Point Cloud from LiDAR

Figure 7.2 presents the classification using only the point cloud. There are three classes: red boxes are pedestrians, blue boxes are cars and green boxes are stakes. In Figure 7.2(c) and (d), some false positives for pedestrians are shown. They are correctly classified in the fusion approach, as it is shown in the Figure 7.3(a) and (b).

7.3 Classification Using Data Fusion Approach

For these tests, the vision-based pedestrian classifier was trained with 8,400 examples taken from several databases, including the LabRoM database, which have special considerations regarding the pedestrian's pose, companions and additional objects. The windows were adjusted to the 2.25 ratio, and their sizes were also adjusted on the basis of the distance to the camera. Pedestrian images were resized to 64 x 128 pixels before extracting the HOG descriptor. For this test, the descriptor had a cell size of 16 x 16 pixels, and a block size of 32 x 32 pixels. The number of gradient bins was set to 9. The resulting feature vector had 756 features. Taking into account this number of training examples (8,400) and this number of features, an SVM with RBF Kernel was tuned for pedestrians. The C and σ parameters were tuned to 55.7 and 0.033 respectively. Using CV this SVM created a model of the data with a F_1 score = 95.4%, $recall$ = 94.5% and $precision$ = 96.3%. For the vision-based car classifier was also used an SVM with RBF Kernel and a HOG feature vector of 864 features; it was tuned with $C = 36$ and $\sigma = 0.027$. The resulting F_1 score was 96.1%, the $recall$ was 98% and the $precision$ was 94.2%. These classifiers were used as hypothesis validation for the LiDAR pre-classification. Several results are shown in Figures 7.3, 7.4 and 7.5.

(a) No false pedestrians

(b) No false pedestrians.

(c) Positives for pedestrian

(d) Positive for car

Figure 7.3: Classification using fusion of LiDAR and Image. The road has been segmented using watershed, redder regions show the resulting segmented road. Red boxes are pedestrians; blue boxes are cars; and green boxes are stakes.

(a) Pedestrian

(b) Different distances

(c) Pedestrian

(d) Person in a motorcycle

(e) Pedestrian in the background

(f) Pedestrians crossing

Figure 7.4: Pedestrian classification using fusion of LiDAR and Images.

(a) Pedestrian

(b) Different distances

(c) Pedestrian

(d) Person in a motorcycle

(e) Pedestrian in the background

(f) Pedestrians crossing

Figure 7.5: Car classification using fusion of LiDAR and Images.

7.4 Car Segmentation Using Foreground Extraction

Foreground extraction by the graph-cut algorithm is a very useful way to segment objects in the image. One advantage of this extraction is that just by specifying a small region the segmentation can be extended to entire image. Figure 7.6 presents several results in the segmentation. One disadvantage is the processing time, which, for a 680 x 480 image, can take up to 3 seconds.

(a) Group Car segmentation

(b) Group car segmentation

(c) Car segmentation

(d) Car segmentation

(e) Car segmentation

Figure 7.6: Segmentation using graph-cut algorithm with min-cut/max-flow optimization.

7.5 Measurements on Classification Data

An urban scenario with several cars and pedestrians was taken for testing the classification based on data fusion. The number of cars and pedestrians and cars were manually established by counting the objects in the video. These data were used as ground-true labels of the scenario and were assigned as the actual classes for the objects. The actual class is needed for calculating the classifier performance.

The classifier performance was established for two cases: the first considering only the objects in the street, and the second considering all the objects in the entire image. The detection and classification measurements are limited to objects closer than 20 meters (from the sensor). Table 7.1 presents the counting of objects in the test scenario.

Table 7.1: Number of objects in the test scenario

Objects	In the Street alone	Entire Scenario
<i>pedestrians</i>	23	37
<i>cars</i>	44	49

7.6 Classification Results

Table 7.2: Pedestrians in only the street.

	Actual Class(Expectation)	
Predicted Class (Observation)	23 Correct Result	0 Unexpected result
	2 Missing Result	0 Correct absent of result

Table 7.2 presents the pedestrian classification in the street. For this case the accuracy is 92 %, the precision is 100%, and the recall is 92%.

Table 7.3 presents the pedestrians classification in the entire scenario. For this case the accuracy is 88%, the precision is 92.5%, and the recall is 94.8%.

Table 7.3: Pedestrian in the entire scenario.

	Actual Class(Expectation)	
Predicted Class (Observation)	37 Correct Result	3 Unexpected result
	2 Missing Result	0 Correct absent of result

Table 7.4: Cars in only the street.

	Actual Class(Expectation)	
Predicted Class (Observation)	36 Correct Result	3 Unexpected result
	5 Missing Result	0 Correct absent of result

Table 7.4 presents the car classification in the street. For this case the accuracy is 81.8%, the precision is 92.3%, and the recall is 87.8%.

Table 7.5: Cars in the entire scenario.

	Actual Class(Expectation)	
Predicted Class (Observation)	37 Correct Result	6 Unexpected result
	6 Missing Result	0 Correct absent of result

Table 7.5 presents the car classification in the entire scenario. For this case the accuracy is 75%, the precision is 86%, and the recall is 86%.

Some missing results are associated with the object color. For the LiDAR there are still problems with dark objects because the light is not reflected. Errors in pedestrians are associated with pedestrian size and distance. Possible pedestrians beyond 20 meters are not considered in this test, so there will be some missing pedestrians. Some unexpected results are due to the shape similarity among pedestrians and stakes, trees and mail boxes.

Table 7.6: Pedestrian detection results. Comparison among the approaches of this dissertation.

Approach	Classified Object	Precision	Recall	Processing time/rate
LiDAR-based	Pedestrian (distance<20 m)	25%	95%	7.5-9.5 Hz
LiDAR-based	Pedestrian (distance<35 m)		75%	
Computer Vision alone	Pedestrian (22.8° V FoV , 30.2° H FoV)	Visually not good	Visually not good	3 to 4 seconds
Fusion	Pedestrian (only in the street, distance <20 m)	100%	92%	166ms/ $\approx$ 6Hz
Fusion	Pedestrian (entire scenario, distance <20 m)	92.5%	94.8%	166 ms/ $\approx$ 6Hz

Table 7.6 presents the results for pedestrian detection. It shows a comprehensive comparison of the approaches studied in this dissertation, that is, LiDAR alone, Computer Vision alone, and fusion of LiDAR and camera. The parameters for comparison are precision, recall and processing time (sometimes mentioned as processing rate, which is the inverse of the time). Results are restricted to pedestrian detection in a range of 20 meters. Detection beyond this range was not included in the comparison because the systems performance decreases as the range increases, for example, recall of detection decreased down to 75% beyond 35 meters, using LiDAR. Two scenarios were considered: pedestrian detection in the street and, pedestrian detection in the entire scenario, i.e., the street and the sidewalks.

Data in Table 7.6 show that the fusion approach improves the precision from 25% to 100% for pedestrian detection in the street alone. The improvement is attributed to the Computer Vision technique, which increases the precision of the detection. Vision technique alone can be expensive in terms of processing time, such as the sliding window method that requires from 3 to 4 seconds to analyze the entire image. However, by combining the Computer Vision technique with the LiDAR detection, that is, the Fusion

approach, the detection can reach the same high precision but with a better processing rate (6Hz).

Chapter 8

Conclusions and Future Work

The idea of intelligent vehicles comes mainly from the goal to make them autonomous. This is the main factor that enables the generation of intelligent algorithms in such robotic vehicles. To design robots that execute tasks by themselves without human intervention telling them explicitly what to do has been a major breakthrough in the artificial intelligence (AI). Sensors with advanced features to sense the world together with such AI-related computer science techniques, such as machine learning, are enabling autonomous vehicles to come to fruition.

Object detection and classification based on data fusion of an artificial vision system and LiDAR sensors provides the vehicle with world data in real time. It has been shown that the fusion approach performs with better processing times and detection results. When compared with the computer vision technique used alone, the fusion runs up to 25 times faster, and, when compared with the point cloud analysis alone, the fusion detects with higher precision. Support Vector Machines, the computer vision technique used for classification, reaches good predictions on new data when it is correctly tuned and trained with a comprehensive database. This technique has been added to the fusion approach using more than 8,000 examples for pedestrians and more than 2,300 examples for cars. Point cloud data coming from LiDAR adds the distance to objects and its spacial position, which enables the computer vision techniques to perform more efficiently. On the other hand, an approach based on LiDAR alone does not give enough confidence in classification, even though it is very useful in detection.

The perception of the world around a car is still a task that is hard to accomplish in real time. The computer vision and point cloud processing algorithms should deal up to 100 MB data to process only in one second (*Velodyne* data outputs up to 700,000 point for frame), and some processing requires up to 70,000 iterations per frame to give a meaning to that data. For example, pedestrian detection based on the sliding windows can take up to 4 seconds to analyze the entire image. In addition, to segment the image using the graph-cut algorithm can take up 2 seconds. Even for a middle-cost (US\$3,000) commercial computer with up-to-date technology (i7 2.53 GHz dual core, solid state drive), those processing tasks cannot be carried out in real time. On the other hand, the data-fusion approach presented in this dissertation improves the processing time of those methods, by reducing the classification time to 166 milliseconds, which it is about 6 Hz. Sensors have frame rates from 10 Hz to 25 Hz, but the fusion approach is accomplished only at 6 Hz because of the data processing. The classifier accuracy reached up 92% for pedestrians and 81.8% for cars. Even though the accuracy is low for cars, the detection rate or precision in cars was up 92.3%. Moreover, for pedestrians the precision was 100%. The classification of one image of 128 x 64 can take up 10 milliseconds, and for car detection, in which the image size is changing and it might even have up 680 x 480 pixels, the classification can take up 50 milliseconds.

The advantage of using the fusion approach is that it is more likely to find obstacles by doing a pre-selection of a small part of the image, a process known as *normalization*. The ceiling analysis in our machine learning problem provides a starting point for implementing the fusion. Results show that select a RoI where the system is more likely to find an obstacle is actually the better approach for reducing processing time and improving performance.

The object classification process in shadow and bright scenes is reached with the descriptor based on local oriented gradients, which is really robust to variations in illumination. In addition, the camera's auto-gain function also yields a way of adjusting to the quick variations while the vehicle is moving. The road estimation process is able to connect dark regions on the street (shadows) and color-marked pedestrian crosswalks to the final, estimated road region.

Perception systems for executing autonomous navigation are still being researched; however, in the meanwhile, most of this technology is used as preemptive assistance to the driver in order to avoid collisions. In the future, it is expected that road fatalities will be reduced by implementing this kind of recognition system in vehicles, and supporting them with a control system that is able to change car direction.

For future work, the hypothesis generator, which is provided by the LiDAR pre-classified data, should be improved with regard to the range detection and the classification itself. Grouping process in the point cloud is still a challenging task because, at the farthest distance, there is more noise in the cloud, which means that it is more difficult to segment and classify those objects. Other LiDAR sensors, such as the Ibeo LUX can support the point cloud classification in a fusion approach. For dark objects, there are still problems for the LiDAR sensor, especially in dark cars. For those cases, the detection should be supported by image analysis. A classifier fusion scheme on a cognition level might support such sensor disadvantages, like Bayes Networks. Optical flow analysis of the scene to find moving objects may be another useful technique to pre-classify objects. According to this dissertation's results, looking for RoIs becomes the more promising approach to improve the system.

Bibliography

- ALLIED VISION TECNOLOGIES. STINGRAY. The Transformer Camera. Technical Report 0, Allied Vision Technologies Gmbh (2014).
- ARLOT, S.; CELISSE, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, v.4p.40–79.
- BAIG, Q. (2010). Low level data fusion of laser and monocular color camera using occupancy grid framework. *Control Automation Robotics & Vision (ICARCV)*, , n.December, p.7–10.
- BARÓ, X.; ESCALERA, S.; VITRIA, J. (2009). Traffic sign recognition using evolutionary adaboost detection and forest-ECOC classification. *IEEE Transactions on Intelligent Transportation Systems*, v.10, n.1, p.113–126.
- BECKER, M.; HALL, R.; JENSEN, B.; KOLSKI, S. (2007). The use of obstacle motion tracking for car-like mobile robots collision avoidance in dynamic urban environments. In: *Proceedings of the XII International Symposium on Dynamic Problems of Mechanics (DINAME 2007)*,. 2007, number Diname, Ilhabbela, SP, Brazil.
- BECKER, M.; HALL, R.; KOLSKI, S.; MACEK, K.; SIEGWART, R.; JENSEN, B. (2009). 2D laser-based probabilistic motion tracking in urban-like environments. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, v.31, n.2, .
- BERTOZZI, M.; BROGGI, A.; ROSE, M. D.; FELISA, M.; RAKOTOMAMONJY, A.; SUARD, F. (2007). A Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier. *2007 IEEE Intelligent Transportation Systems Conference*, p. 143–148, September.

- BESBES, B.; LABBÉ, B.; ROGOZAN, A.; BENSRAHAIR, A. (2010). SVM-based fast pedestrian recognition using a hierarchical codebook of local features. In: *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*. 2010, number Mlsp, p. 226–231. IEEE.
- BOYKOV, Y.; KOLMOGOROV, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, v.26, n.9, p.1124–37, September.
- BOYKOV, Y.; KOLMOGOROV, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, v.26, n.9, p.1124–37, September.
- CASTELLANOS, J. A.; NEIRA, J. (2001). Multisensor Fusion for Simultaneous Localization and Map Building. *IEEE Transactions on Robotics*, v.17, n.6, p.908–914.
- CHANG, C.; LIN, C. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems*, p. 1–39.
- COSTA, R.; De Moura Lacerda, V.; CARMONA, A.; BRITO, A.; MARCELO, B. (2011). Classificação de Pedestres usando Câmera e Sensor Lidar. In: *X SBAI. Simpósio Brasileiro de Automação Inteligente*. 2011, volume X, p. 416–420.
- DALAL, N.; TRIGGS, B. (2005). Histograms of oriented gradients for human detection. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. 2005, volume 1, p. 886–893. IEEE.
- DOLLAR, P.; WOJEK, C.; SCHIELE, B.; PERONA, P. (2012). Pedestrian Detection: An Evaluation of the State of the Art. *IEEE transactions on pattern analysis and machine intelligence*, v.34, n.4, p.743–761, July.
- ENZWEILER, M.; GAVRILA, D. M. (2009). Monocular pedestrian detection: survey and experiments. *IEEE transactions on pattern analysis and machine intelligence*, v.31, n.12, p.2179–95, December.

-
- GERÓNIMO, D.; LÓPEZ, A. M.; SAPPA, A. D.; GRAF, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *IEEE transactions on pattern analysis and machine intelligence*, v.32, n.7, p.1239–58, July.
- GREIG, D.; PORTEOUS, B.; SEHEULT, A. (1989). Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*.
- HABERMANN, D. (2010). *Detecção e rastreamento de obstáculos com uso de sensor laser de varredura*. Tese (Doutorado), Escola Politécnica. USP.
- HAYKIN, S. (1999). *Neural networks: a comprehensive foundation*. Pearson Education, Inc.
- HSU, C.; CHANG, C.; LIN, C.; OTHERS. A practical guide to support vector classification. Technical Report 1, Department of Computer Science National Taiwan University (2010).
- HU, X.; ZHU, X.; LI, D.; LI, H. (2010). Traffic sign recognition using Scale invariant feature transform and SVM. *A special joint symposium of ISPRS*.
- IOANOU, Y.; TAATI, B. (2012). Difference of Normals as a Multi-Scale Operator in Unorganized Point Clouds. *3D Imaging, Modeling, . . .*
- JAGGI, M. (2008). *Geometry of Support Vector Machines*. www.m8j.net/.
- KANE, J. M. O. (2013). *A Gentle Introduction to ROS*. University of South Carolina.
- KIDONO, K.; MIYASAKA, T.; WATANABE, A.; NAITO, T.; MIURA, J. (2011). Pedestrian recognition using high-definition LIDAR. In: *Intelligent Vehicles Symposium (IV), 2011 IEEE*. 2011, number Iv, p. 405–410. IEEE.
- LAGANI, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing.
- LLINÁS, R. R. (2002). *I of the vortex: from neurons to self*. MIT Press.

- LUDWIG, O.; DELGADO, D.; GONCALVES, V.; NUNES, U. (2009). Trainable classifier-fusion schemes: an application to pedestrian detection. In: *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*. 2009, p. 1–6. IEEE.
- MALDONADO-BASCÓN, S.; LAFUENTE-ARROYO, S.; GIL-JIMÉNEZ, P.; GÓMEZ-MORENO, H.; LÓPEZ-FERRERAS, F. (2007). Road-sign detection and recognition based on support vector machines. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, v.8, n.2, p.264–278.
- MARTINEZ, A.; FERNANDEZ, E. (2013). *Learning ROS for Robotics Programming*. Packt Publishing, Birmingham, UK.
- MIGUEL, P.; GOMES, D. B. (2010). *LADAR Based Mapping and Obstacle Detection System for Service Robots*. Tese (Doutorado), Universidade Nova de Lisboa.
- MOHOTALLA, S.; ONO, S.; KAGESAWA, M.; IKEUCHI, K. (2009). Fusion of a camera and a laser range sensor for vehicle recognition. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, p. 16–23, June.
- MONTEIRO, G.; PEIXOTO, P.; NUNES, U. (2006). Vision-based pedestrian detection using Haar-like features. *Robotica*.
- NASHASHIBI, F.; KHAMMARI, A.; LAURGEAU, C. (2008). Vehicle recognition and tracking using a generic multisensor and multialgorithm fusion approach. *Int. J. Vehicle Autonomous Systems*, v.6, n.February 2006, p.134–154.
- National Highway Traffic Safety Administration of U.S. Government. TRAFFIC SAFETY FACTS 2009. Technical report, NHTSA's National Center for Statistics and Analysis, Washington (2009).
- NG, A. (2012). *Machine Learning*. www.coursera.org/course/ml, 1st edition.
- NG, A. Support Vector Machines. In: *Lectures on Machine Learning*, number x, chapter Part V, p. 1–25. Stanford University (2012).

-
- OLIVEIRA, L.; NUNES, U.; PEIXOTO, P. (2010). On exploration of classifier ensemble synergism in pedestrian detection. . . . *Systems, IEEE Transactions on*, v.11, n.1, p.16–27.
- OZGUNER, U. (2005). Environmental perception and multi-sensor data fusion for off-road autonomous vehicles. *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, p. 584–589.
- RODRIGUEZ, F. (2008). Extrinsic calibration between a multi-layer lidar and a camera. *Multisensor Fusion and*, p. 214–219, August.
- ROTHER, C.; KOLMOGOROV, V.; BLAKE, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*.
- RUSSEL, S.; NORVIG, P. (1994). *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., Upper Saddle River, New Jersey.
- RUSU, R. B.; COUSINS, S. (2011). 3D is here: Point Cloud Library (PCL). *2011 IEEE International Conference on Robotics and Automation*, p. 1–4, May.
- RUTA, A.; LI, Y.; LIU, X. (2008). Detection, Tracking and Recognition of Traffic Signs from Video Input. *2008 11th International IEEE Conference on Intelligent Transportation Systems*, p. 55–60, October.
- SERFLING, M.; LOEHLEIN, O.; SCHWEIGER, R.; DIETMAYER, K. (2009). Camera and imaging radar feature level sensorfusion for night vision pedestrian recognition. *2009 IEEE Intelligent Vehicles Symposium*, p. 597–603, June.
- VELODYNE. HDL-32E High Definition LiDAR Sensor. Technical report, Velodyne LiDAR, Inc. (2013).
- WANG, Z.-R.; JIA, Y.-L.; HUANG, H.; TANG, S.-M. (2008). Pedestrian Detection Using Boosted HOG Features. *2008 11th International IEEE Conference on Intelligent Transportation Systems*, p. 1155–1160, October.
- WEGNER, R. (2013). *Analysis of 3D-laser data*. Tese (Doutorado), Berner Fachhochschul.

World Health Organization. Global status report on road safety. Technical report, WHO Library Cataloguing-in-Publication Data, Geneva (2009).

World Health Organization. Decade of Action for Road Safety 2011-2020. Technical report, World Health Organization, Geneva (2011).

World Health Organization. Global Status Report on Road Safety 2013: Suporting a Decade of Action. Technical report, WHO, Geneve (2013).

ZHANG, G.; GAO, F.; LIU, C.; LIU, W.; YUAN, H. (2010). A pedestrian detection method based on SVM classifier and optimized Histograms of Oriented Gradients feature. In: *Natural Computation (ICNC), 2010 Sixth International Conference on*. 2010, volume 6, p. 3257–3260. IEEE.

ZHOU, S.; IAGNEMMA, K. (2010). Self-supervised learning method for unstructured road detection using fuzzy support vector machines. *Robots and Systems (IROS), 2010 IEEE*, p. 1183–1189.

Appendix A

Sensors in SENA Project

Table A.1: Sensors on *Gisa* prototype

Sensor	Transfer rate (Hz)	Specifications
Velodyne HDL-32E	10	LiDAR sensor with 32 laser/detector pairs A 360° horizontal field of view (FoV) A 41.3° vertical FoV Usable returns up to 100 meters and up to 2 cm precision
AsteRx2eh RTK-GPS	20	Geo-localization system Two antennas GPS e GLONASS. Horizontal precision (HP) of 1.3 m in the normal mode HP of 0.6 cm in the <i>RTK</i> mode
AVT Stingray F-033C	up to 60	Color CCD Camera, VGA (656 x 494) HFOV:30.2° VFOV:22.8°
ibeoLUX	12.5	LiDAR sensor of four lasers Horizontal FoV of 85° and vertical FoV of 3.2° 200 m range and 10 cm precision
IG - 500N	100	IMU with 3 gyroscopes, 3 accelerometers and 3 magnetometers

Appendix B

Cost Function Modeling

This appendix presents more details of the cost function for classifiers (NG , 2012b).

B.0.1 Model for Multiple Features

When there are several features, it is commonly used a linear regression with multiple variables as the model. It is also known as “multivariate linear regression.” The following equations are the ones for any number of input variables:

$$J(\theta) = \text{Cost function}$$

$$n = |x^{(i)}| \text{ the number of features}$$

$$x^{(i)} = \text{input features of the } i^{\text{th}} \text{ training example}$$

$$x_j^{(i)} = \text{value of feature } j \text{ in the } i^{\text{th}} \text{ training example}$$

$$y^{(i)} = \text{class of the } i^{\text{th}} \text{ training example}$$

(B.1)

Here, the form for the hypothesis function to accommodate multiple features is gotten:

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

Thus, it can be done matrix operations with θ and x , we will let $x_0^{(i)} = 1$. This makes the two vectors θ and x match each other element-wise. This hypothesis function can be more concisely represented as:

$$h_{\theta}(x) = \theta^T x$$

This is a vectorization of our hypothesis function. This version of the hypothesis function assumes the matrix x and vector θ store their values as follows:

$$x = \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & x_0^{(3)} \\ x_1^{(1)} & x_1^{(2)} & x_1^{(3)} \end{bmatrix}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

B.0.2 Cost function

For the parameter vector θ (of type $\Re^{n+1}$ or in $\Re^{(n+1) \times 1}$), the cost function is:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Where:

$$\text{cost}(\theta, x) = (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

The vectorized version is:

$$J(\theta) = \frac{1}{2m} (X\theta - \vec{y})^T (X\theta - \vec{y})$$

Where $\vec{y}$ denotes the vector of all y values.

B.0.3 SVM Cost Fuction using RBF kernel f

$l^{(i)}$: Landmark based on example i

$cost_1$: Cost for positive examples

$cost_0$: Cost for negative examples

n : Number of features

m : Number of examples

Given a new sample x :, the kernel is defined as:

$$\begin{aligned} f_i &= \text{similarity}(x, l^{(i)}) \\ &= \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right) \end{aligned} \tag{B.2}$$

It is expected to predict $y = 1$ if

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 + f_3 \geq 0$$

Given a set of training examples: $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})$

The landmarks for similariry function: $l^{(1)}, l^{(2)}, l^{(3)}, \dots$

are choosen according to: $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \dots, l^{(m)} = x^{(m)}$

So that,

$$\begin{aligned} f_1 &= \text{similarity}(x, l^{(1)}) \\ f_2 &= \text{similarity}(x, l^{(2)}) \\ &\dots \end{aligned} \tag{B.3}$$

Hypothesis: Given x , compute features $f \in \mathbb{R}^{m+1}$ in order to predict " $y = 1$ ". if $\theta^T f \geq 0$.

Solution: Training the SVM by minimizing the cost function $J(\theta)$ for θ :

$$J(\theta) = C \sum_{i=1}^m cost_1(\theta^T f^{(i)}) + (1 - y^{(i)})cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^n \theta_j^2$$