Universidade de Aveiro Departamento de Engenharia Mecânica
2017

José Diogo
Madureira Correia

Unidade de Perceção Visual e de Profundidade Para o Atlascar2

Visual and Depth Perception Unit for ATLASCAR2

Universidade de Aveiro Departamento de Engenharia Mecânica
2017

José Diogo
Madureira Correia

Unidade de Perceção Visual e de Profundidade Para o Atlascar2

Visual and Depth Perception Unit for ATLASCAR2

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Paulo Miguel de Jesus Dias, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

O júri / The jury

Presidente / President

Prof. Doutor Jorge Augusto Fernandes Ferreira
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Vítor Manuel Ferreira dos Santos
Professor Associado da Universidade de Aveiro (orientador)

Prof. Doutor Carlos Fernando Couceiro de Sousa Neves
Professor Coordenador da Instituto Politécnico de Leiria - Escola Superior de Tecnologia e Gestão

Agradecimentos / Acknowledgements

Deixo aqui os meus mais profundos agradecimentos à minha familia que sempre me ajudou e apoiu nesta jornada, em especial à Marisa Alves que esteve sempre ao meu lado mesmo nos momentos mais dificeis.

Agradeço aos Professores Vitor Santos e Paulo Dias pela orientação e apoio ao longo deste trabalho, assim como ao Eng. António Festas pela disponiblidade e prontidão para ajudar.

Ao ISR (Instituto de Sistemas e Robótica) da Universidade de Coimbra e a todos os seus colaboradores pela cedencia de equipamento e por tudo o apoio e acompanhamento durante a sua utilização.

Por fim, mas não menos importante, deixo os meus sinceros agradecimentos a todos os meus colegas que me ajudaram durante esta jornada.

Palavras-chave

Navegação autónoma; LIDAR; ATLASCAR; Fusão multisensorial; Calibração; Instalação de hardware.

Resumo

Este trabalho assenta na instalação de sensores *Light Detection And Ranging* e de visão numa plataforma movel à escala real, o ATLASCAR 2. Este veículo é um Mitsubishi i-MiEV que, no ambito deste trabalho, será equipado com dois scaners planares, um scaner 3D e uma camara. Estes sensores serão instalados na frente do veículo e suportados por uma infraestrurura desenvolvida em perfil da alumínio e fixa ao chassis do mesmo. A alimentação dos sensores é feita atravéz do circuito de baixa tensão do veículo e controlada por um quadro elétrico situado no porta bagagens juntamente com a unidade de processamento. A calibração destes sensores realizou-se atravéz um pacote de calibração multisensorial devenvolvido no Laboratorio de Automação e Robotica, ao qual foi adicionada a opção de calibrar um novo sensor 3D, Velodyne Puck VLP-16. Após a calibração dos sensores e no sentido de demonstrar as funcionalidades da plataforma, foi desenvolvida uma aplicação que combina os dados dos sensors *Light Detection And Ranging* devidamente referenciados e calcula e representa o espaço, disponivel para navegar em torno do veiculo.

Keywords Autonomous driving; LIDAR; ATLASCAR; Multisensor data merge; Calibration; Hardware installation.

Abstract This thesis is focused on the installation of multiple Light Detection And Ranging and vision-based sensors on a full sized mobile platform, ATLASCAR 2. This vehicle is a Mitsubishi i-MiEV. In the scope of this work it will be equipped with two planar scanners, a 3D scanner and a camera. The sensors will be installed in the vehicle's front supported by an infrastructure built in aluminium profile and connected to the vehicle's chassis. All sensors are powered by the car's low voltage circuit and controlled by a switched board placed in the trunk alongside with a processing unit. Sensor calibration is accomplished using a calibration package developed at the Laboratory of Automation an Robotics, to which an option to calibrate a new 3D sensor was added, Velodyne Puck VLP-16. After the sensor calibration and to demonstrate the functionalities of the platform, an application was developed that merges the data from the Light Detection And Ranging sensors, properly referenced, in a single frame and computes and represents the space free to navigate around the vehicle.

Contents

List of Figures

List of Tables

Acronyms

AD Autonomous Driving. 1, 3, 4, 6, 10, 22, 25, 43, 71

ADAS Advanced Driver Assistance Systems. 1, 3, 6, 7, 10, 25, 43, 71, 72

DARPA Defense Advanced Research Projects Agency. 7

GUI Graphical User Interface. iii, 7, 19, 21

ISR Institute of Systems and Robotics. 14, 15, 46

LAR Laboratory for Automation and Robotics. i, 1, 3, 6, 7, 15, 18, 21, 31, 46

LARtk Laboratory for Automation and Robotics Toolkit. 77

LIDAR Light Detection And Ranging. ii, iv, 1, 3, 4, 7, 9–11, 14, 19, 21, 25, 26, 35, 43, 45–47, 49–53, 55, 61–63, 71, 77

ODB-II On-Board Diagnostics. 72

OpenCV Open Source Computer Vision Library. 6, 22

PCL Point Cloud Library. 18, 20, 22, 46, 53

ROS Robot Operating System. 18–20, 43, 45, 53

SCOT Shared Computer-Operated Transit. 6

SMART Singapore-MIT Alliance for Research and Technology. 6

TF ROS Transform. 52

UPS Uninterruptible Power Supply. ii, 17, 33, 36, 75, 76

Chapter 1

Introduction

Advanced research in Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS) has been growing in large scale since the last decade, not only in the academic world but also on the automobile industry [29]. As a result, cheaper and more powerful sensors are emerging. A good example of this evolution are LIDAR sensors as they are becoming the largest segment of automotive detection and ranging sensors, complementing or replacing radars and cameras [17].

Since there are many hardware solutions for robot's perception and even more approaches for data fusion and analysis, having a platform equipped with multiple LIDAR and vision-based sensors, and prepared to integrate any kind of sensor, combined with an easy and fast calibration process is a great step forward in AD and ADAS research.

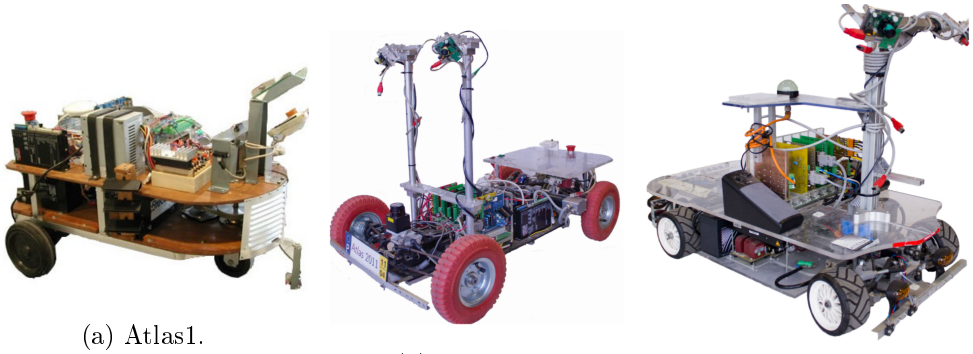
This dissertation provides a research automobile platform equipped with multiple LIDAR and vision-based sensors as an extension of the work made by Silva [2] on sensor automatic calibration and data fusion presenting an application to represent the free space around the platform.

1.1 ATLAS Project

1.1.1 History

The ATLAS Project was created in 2003 by the Group of Automation and Robotics at LAR on the Department of Mechanical Engineering of the University of Aveiro. ATLAS has the mission to develop and enable the proliferation of advanced sensing and active systems designed for implementation in automobile platforms [16]. The project begun by developing scale prototypes, some of them shown in figure 1.1, to enrol in robotic competitions such as the Portuguese Robotics Open (PRO) in which they were very successful. On ten years participating in PRO the Atlas prototypes won the first place in six of them and second and third places in three participations [6]. With the large background provided by years of experience in autonomous navigation in controlled environments the ATLAS Project toke a step forward. In 2010 the group of Group of Automation and Robotics begun the development of a real scale vehicle ATLASCAR, figure 1.2, equipped with several state of the art sensors [16].

The ATLASCAR 1 is a Ford Escort Station Wagon equipped with several sensors, actuators and computational units. Sensors collect data about the vehicle's state and its surroundings witch is then processed by the computational units.These units then send



(a) Atlas1. (b) Atlas2000. (c) AtlasMV.

Figure 1.1: ATLAS autonomous robots.

Figure 1.2: ATLASCAR1 - Ford Escort Station Wagon.

informations to the actuators allowing the vehicle to move and execute manoeuvrers autonomously. All the external sensors, actuators and processing units are powered by an auxiliary alternator that is connected to a buffer battery powering an inverter. The output from the inverter is stabilized by an UPS and then converted by two voltage regulators to 24V DC and 12V DC to power all the external devices. ATLASCAR 1 will serve as inspiration for this work.

Although ATLASCAR 1 served its purpose well, after years of modifications and experiments it is approaching its end-of-life and is going to be replaced by a more modern vehicle, a Mitsubishi i-MiEV, figure 1.3. ATLASCAR 2, the new Mitsubishi i-MiEV, is a fully electric vehicle that can go up to 150 Km without recharging its batteries with 16 KWh of capacity. Being an electric vehicle, ATLASCAR 2 has more potential to power all the external devices with less modifications than the ones required in ATLASCAR

1, as well as it is expected that it will be probably easier to control, thus reducing the number of actuators required.

Figure 1.3: ATLASCAR2 - Mitsubishi i-MiEV.

1.1.2 Related work at LAR

Research related to AD has been a major topic at LAR. Within the scope of this project several thesis and projects have been developed, some related to multisensor calibration like the work done by Pereira [3] on LIDAR calibration or the most recent work done by Vieira [2] on multisensor calibration described forward in more detail. Some other related with the hardware development for the Atlas prototypes and others related with perception algorithms using data from different types of sensors like the work developed by Azevedo [4]. A good summary of the work related with AD developed at LAR is the PhD thesis presented by Oliveira [6].

1.2 Problem description

In order to have a research platform for ADAS, the new Mitsubishi i-MiEV has to be equipped with multiple LIDAR and vision-based sensors. This task requires the installation of additional fixtures to support the devices as well as new cables and wires to provide power and communications between the processing units and the sensors or other devices. Because the main processing units will be placed in the vehicle's trunk and the majority of the sensors will be at the front, cables for power and communication need to run from the trunk to the vehicle's front without compromising the vehicle integrity and appearance. The same principles applies to the additional fixtures required for the sensors supports.

Having an instrumented prototype ready to acquire data is essential, but in order to be able to use this data for perception or for navigation, it is essential to have the multiple sensors registered in the same coordinate system. This can be accomplished by an extrinsic calibration procedure as the one descried in [2]. With the referenced data it is possible to combine the multiple scans from the LIDAR sensors and even data from vision-based sensors and merge them into a single frame. This allows for a better perception and, as a result, better decision-making.

On the other hand, to properly merge data from multiple sensors, more than just the extrinsic transformations between the sensors is required. For example, to merge the data from LIDAR sensors with the propose of obtaining a representation of the space available for ATLASCAR 2 to navigate is required to have the information of the sensors location on the vehicle. Nevertheless, the data has to be pre-processed to remove invalid objects such as ground detections and post-processed to remove useless points.

1.3 Objectives

The objectives of this work are to study the best location for the sensors in the new prototype, develop and build the infrastructure required to support the devices and install all the extra cables necessary for the communication between the processing units and the sensors and to provide power to them.

With the goal of multisensor data fusion, after the hardware installed it is necessary to calibrate the devices, develop software to merge the data in a single frame and, finally, use the merged data to create, in real time, a representation of the free space around the vehicle.

1.4 Work related to autonomous driving and multisensor fusion

1.4.1 Autonomous vehicles

Major corporations such as Audi and Nvidia, Volvo and Microsoft, Daimler and Uber and Waymo and Lyft have partnered up with the goal of bringing autonomous vehicles to consumers [30].

A good example of such partnership is the Uber ATC car (figure 1.4). This vehicle is being used in Pittsburgh to collect mapping data and test its self-driving capabilities [27]. Uber ATC car is equipped with state of the art LIDAR sensors as well as high resolution cameras and radars.

Another example is the self-driving vehicle from Google, Waymo. Google started this self-driving project in 2009 and since then it has been leading the research in AD. Waymo's fleet has more than 3 million miles self-driven, mostly on city streets. Waymo also has the title of the "World's first and only fully self-driving ride on public roads". Recently Waymo become a private company and toke a new step forward with a new fleet of fully self-driving vehicles (figure 1.5) with a fully-integrated hardware suite designed for the purpose of full autonomy [29].

Although major corporations have a strong role in AD research they are not the only ones. Many universities and research institutes dedicate their resources to the

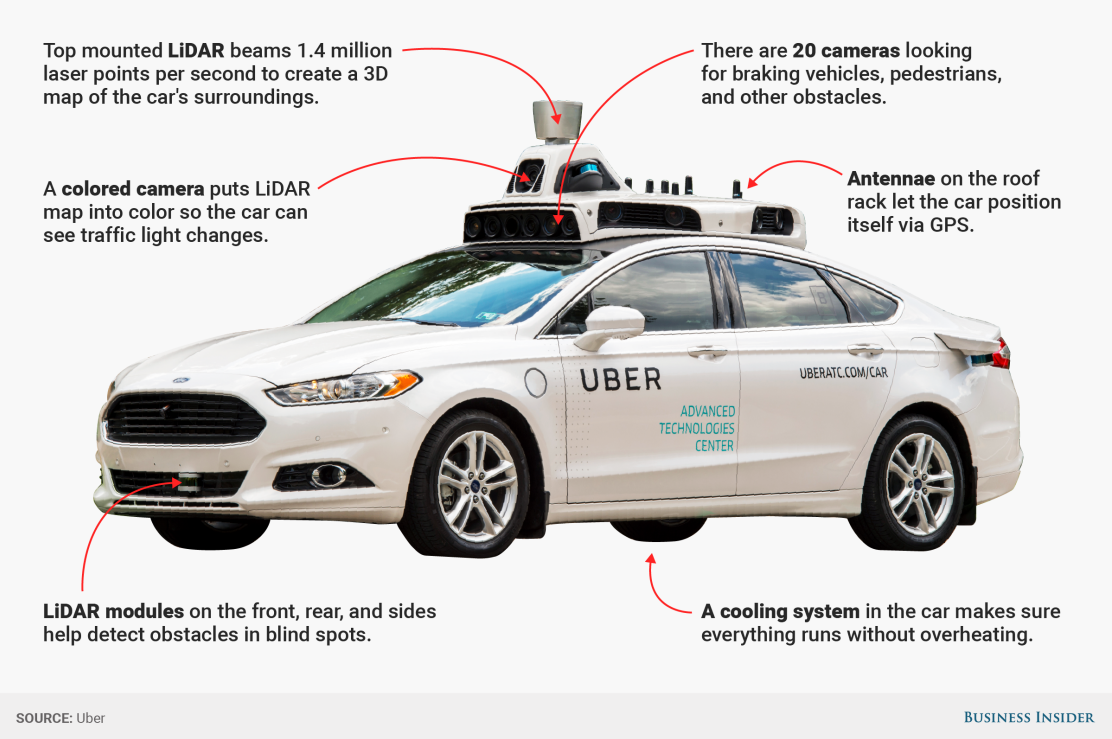


Figure 1.4: Ford Fusion Uber ATC car.

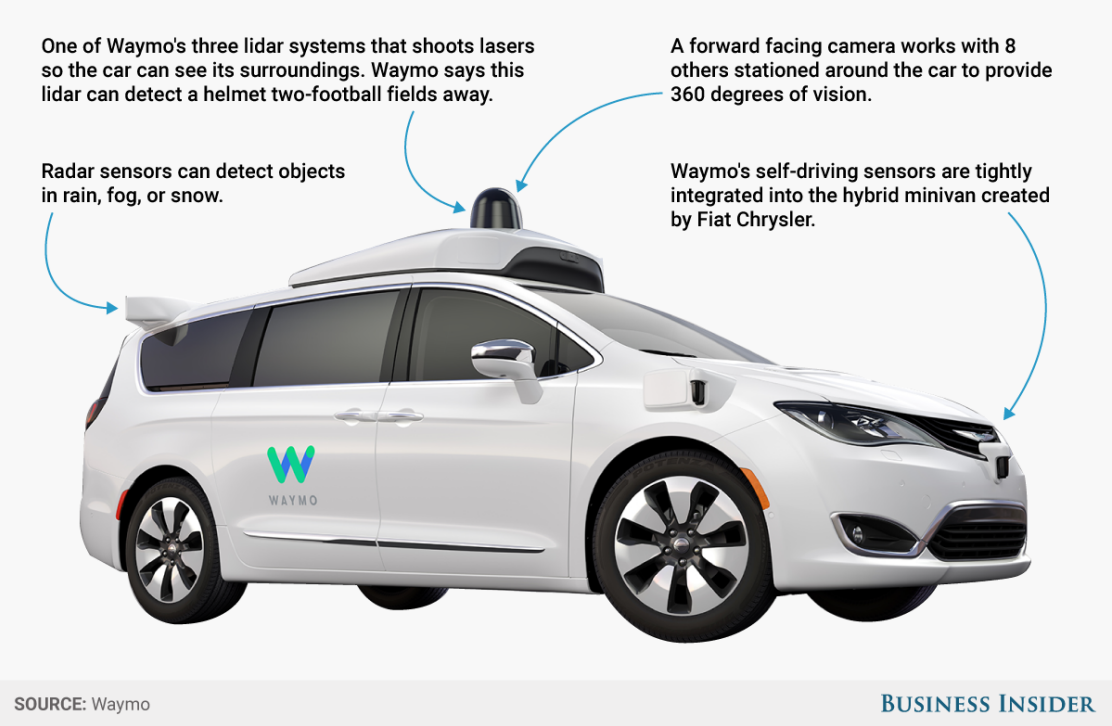


Figure 1.5: Waymo self-driving vehicle, based on a Chrysler Pacifica Hybrid minivan.

research in ADAS and AD. Another source of inspiration is Shared Computer-Operated Transit (SCOT), figure 1.6. It is a research platform from the Singapore-MIT Alliance for Research and Technology (SMART) research enterprise [26]. SCOT is a Mitsubishi i-MiEV equipped with several sensors of which a Velodyne Puck and two Sick LMS, somehow similar to ATLASCAR 2.

Figure 1.6: SCOT - Shared Computer-Operated Transit vehicle [20].

1.4.2 LIDAR data fusion

In order to properly merge data from multiple sensors they must be referred in a common coordinate system. For that, it is essential to have the extrinsic transformations between each sensor. In this matter, several solutions have been proposed along the years. In fact, at LAR, this subject has been the base for others thesis. The first proposed solution was presented by Almeida et al., in [7]. The author used a know object consisting in two cones connected by a plane. The conic geometry removes the height ambiguity and the second cone provides information to determine the sensors orientation around their vertical axis. The 2D scanners generate two ellipse sections connected by a line corresponding to the calibration object. Then, a fitting algorithm, from the Open Source Computer Vision Library (OpenCV) is used to find the ellipse that best fits the data extracted from the laser scans. By using the intersections of the major and minor ellipse axis the author fits the laser data to the calibration object and computes the rigid body transformation from the laser scans to the 3D data. This solution contemplated only 2D and 3D laser scanners and presented some problems with the calibration of the roll angle. According to the author this problem was caused by the lack of points in the 2D data to proper fit the

ellipse. Later on, Pereira [3] developed an automatic extrinsic calibration method using a ball with known dimensions. The ball is detected by the multiple sensors and for each sensor the ball's center is computed and store in a point cloud. After having obtained a certain number of centroids the point clouds of each sensor are aligned relative to a reference sensor and the transformations between each sensor and the reference frame are found. This method was tested in ATLASCAR 1 with two SICK LMS151, one SICK LD-MRS400001 and two Point Grey cameras in stereo configuration. It presented reliable results for all the sensors exept for the stereo cameras due to the error associated with the calculation of the disparity map and the 3D reconstruction. The last work on this subject was done by Silva [2]. The author integrated two more sensors — Point Grey FL3-GE-28S4-C camera and a Microsoft Kinect — in the calibration package and developed an easy to use GUI. This package will be used in this thesis to calibrate the sensors and will be described in more detail in chapter 2.

Calibration is an early and necessary step for merging the data from multiple sensors. On this topic, although there is little work at LAR, there are many articles on this matter. After an analysis to the available literature, it was possible to conclude that: there are two main approaches for data fusion. One is to analyse the data separately, draw conclusions, and then merge the conclusions, as used in [11]. An other is to merge the raw or pre-processed data and then draw conclusions from the merged data, method used by the "Boss" vehicle in the Defense Advanced Research Projects Agency (DARPA) Grand Challenge [10].

1.5 Document structure

This document is composed by six chapters including the present one. Chapter 2 first describes the main characteristics and most important features of the hardware used in this work. More specifically the vehicle where the sensors will be installed, the sensors and other significant devices like the processing unit and the switch. Next the software tools used in this thesis is described. Chapter 3 first explains the mechanical design and construction process, from the mechanical design of the fixtures for the sensors, passing through the sensor locations study and finishing with the assembly process. Next, this document describes how all sensors are connected to the processing unit and how they are powered. Chapter 4 is dedicated to the description of the software developed to connect the sensors, merge the data and compute the free space around the vehicle. Finally, chapter 5 presents the results of this work, namely the quality of the fixtures for the hardware; the result of the calibration process including the calibration of a 3D LIDAR; and the results of the free space detection and data merging nodes. In the last chapter the conclusions of this thesis are presented and some proposals for future works related with ATLASCAR 2 and ADAS are described.

Chapter 2

Experimental infrastructure

This chapter describes in more detail the test platform (ATLASCAR 2) where the sensors will be installed. It gives a detailed description of each LIDAR and camera used as well as a quick overview of the hardware required to establish the communication infrastructure between the sensors and the processing unit. Finally, the software, libraries and tool kits used are explained.

2.1 ATLASCAR 2

As mentioned in section 1.1, ATLASCAR 2 is a 2015 Mitsubishi i-MiEV, figure 2.1. It is a fully electric vehicle with a 49 kW engine connected to the rear wheels. ATLASCAR 2 has a 16 kWh Lithium-ion battery to power the traction engine, which will be also used to power the external devices — sensors and processing units — and a 12 V lead battery to power the vehicle's auxiliary systems and to connect the relays isolating the Lithium-ion battery from the power circuit of the vehicle when it is not in operation. When the Lithium-ion battery is connected, it powers the high voltage circuit and also charges the lead battery and powers the low voltage circuit. The Lithium-ion battery can be charged on a regular 230 VAC power plug (up to 10 hours for a full charge) or on a fast charging station (up to 30 minutes for a full charge). Table 2.1 describes the main characteristics of this vehicle.

Figure 2.1: Mitsubishi i-MiEV.

This thesis is the first intervention on ATLASCAR 2, since it had never suffered any kind of modification or had been used in any other work.

Characteristic	Unit	Value
Wheelbase	mm	2550
Track (Front/Rear)	mm	1310/1270
Vehicle weight	kg	1450
Engine	–	Electric
Electric energy consumption	Wh/km	135
Electric range (NEDC)	km	150
Maximum speed	km/h	130
Minimum turning radius	m	4.5
Max. Power output	kW	49
Max. torque	Nm	180
Traction battery type	–	Lithium-ion battery
Traction battery voltage	V	330
Traction battery energy	kWh	16
Regular charging (AC 230V 1 phase) 8A	hrs	10

Table 2.1: Mitsubishi i-MiEV technical specifications [18]

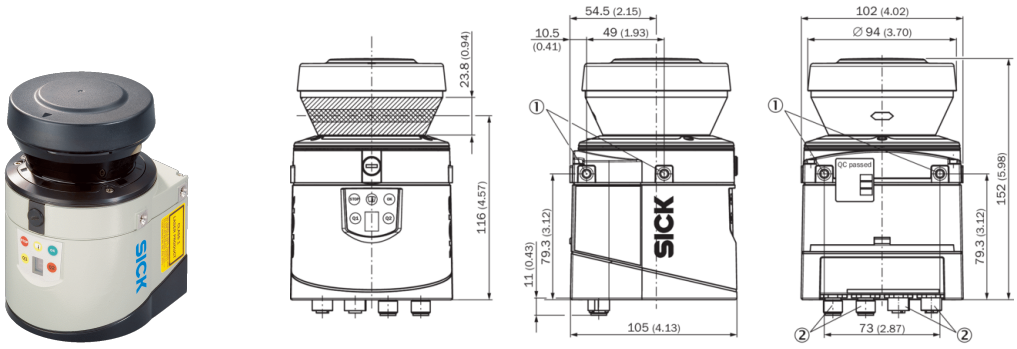
2.2 Sensors

LIDAR, or Light Detection And Ranging is the name of a technology that uses light to find the distance to a certain object. Usually, LIDAR sensors use a pulsed light or phase analyses from a laser. The distance to the object is found by measuring the time between the emission of the light pulse and the detection of its reflection. This technology is used in many areas such as topography and AD. Depending on the application and on the quality of the sensor, it can create a dense 3D point cloud or a planar scan with precisions that can go up to the millimetre.

This section describes the main characteristics of all the sensors (LIDAR and vision-based) to be installed in ATLASCAR 2.

2.2.1 Sick LMS151

The LMS151, figures 2.2a and 2.2b, is a robust 2D laser scanner (LIDAR) designed for both indoor and outdoor use, providing reliable data for navigation, detection and measurement [25]. It has a multi-echo technology and fog detection allowing it to measure distances through glass, fog and dust. Sick LMS151 has a high aperture angle, high scan frequency and long detection range making it suitable for use in AD and ADAS. At the begin of this work, there were two of these devices available to install in ATLASCAR 2. Table 2.2 details the most important specifications of this scanner.



(a) Sick LMS151.

(b) Sick LMS151 dimensional drawings.
1 - Mounting hole M5 x 7.5
2 - Connector M12

Figure 2.2: Sick LMS151 views.

Light source		Infrared (905 nm)
Laser class		1 (IEC 60825-1 (2007-3))
Aperture angle		270º
Scanning frequency		25 Hz / 50Hz
Angular resolution		0.25º / 0.5º
Operating range		0.5 a 50 m
Max. range with 10% reflectivity		18 m
Detectable object shape		Almost any
Systematic error		± 30 mm
Operating voltage		10.8 V DC to 30 V DC
Power consumption		Typ. 8 W, heating typ. 35 W
Weight		1.1 kg
Dimensions (L x W x H)		105 mm x 102 mm x 162 mm
Interfaces	Ethernet	TCP/IP 10/100 MBit/s
	Serial	RS-232 9.6 kBaud to 115.2 kBaud
	CAN	Extension of outputs
	IO's	4 Digital Inputs, 3 Outputs
	Optical indicators	7-segment display and 5 LEDs

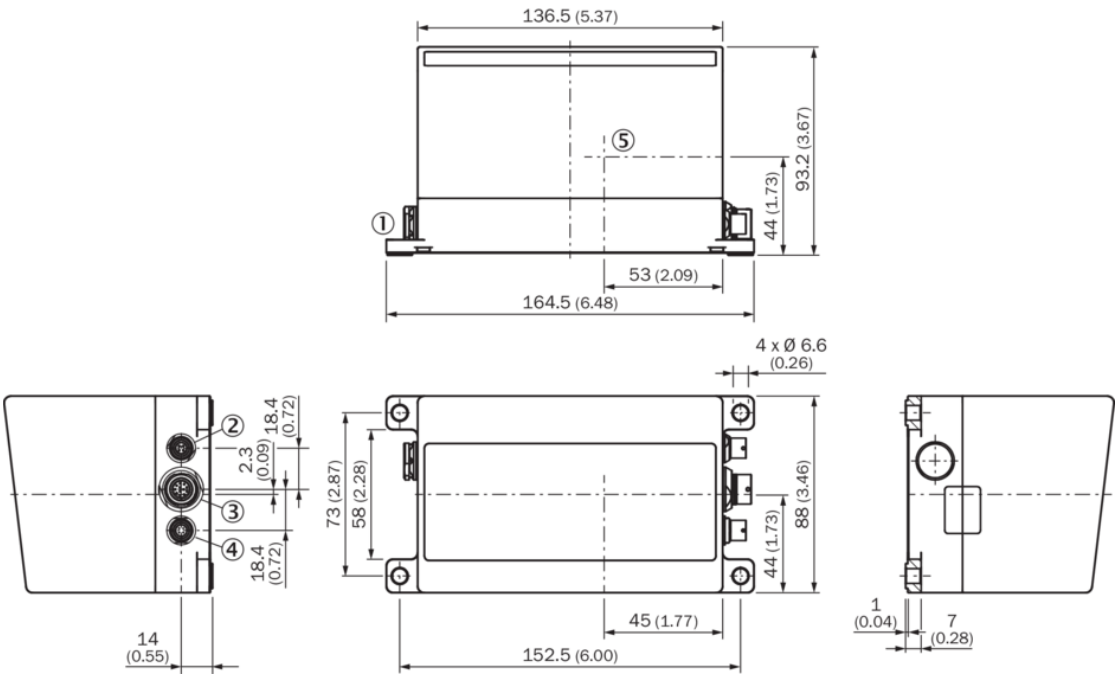
Table 2.2: Sick LMS151 technical specifications [25].

2.2.2 Sick LD-MRS400001

The Sick LD-MRS400001 is a 3D LIDAR, ideal for collision detection on automated vehicles or the scanning of objects [25]. With 4 scan planes with a vertical aperture of 0.8° between each plane, it can detect obstacles in the scan field, even when the vehicle is

accelerating or braking [25]. Its multi-echo technology ensures the objects are detected even with adverse weather conditions and, because different angular resolutions along the scanning range are available, better efficiency can be achieved improving the angular resolution in sensitive areas and reducing the number of point in the less important ones. Table 2.3 describes the most important specifications of this device.

(a) Sick LD-MRS400001.



(b) Sick LD-MRS400001 dimensional drawings.
1 - venting element; 2 - "Ethernet" connection
3 - connection data interface / synchronisation
4 - connection "Power"

Figure 2.3: Sick LD-MRS400001 views.

Light source		2 laser diodes, infrared light
Laser class		1 (IEC 60825-1 (2007-3))
Aperture angle		85° or 110°
Scanning frequency		12.5 Hz to 50Hz
Angular resolution		0.125°/ 0.25°/ 0.5°
Operating range		0.5 a 300 m
Max. range with 10% reflectivity		50 m
Detectable object shape		Almost any
Statical error		100 mm
Operating voltage		9 V DC to 27 V DC
Power consumption		Typ. 8 W
Weight		1 kg
Dimensions (L x W x H)		94 mm x 165 mm x 88 mm
Interfaces	Ethernet	TCP/IP 100 MBit/s
	Serial	RS-232 57,600 Baud
	CAN	Auxiliary interface

Table 2.3: Sick LD-MRS400001 technical specifications [25].

2.2.3 Point Grey ZBR2-PGEHD-20S4C

The Point Grey ZBR2-PGEHD-20S4C is an IP camera from Point Grey and is part of the Zebra2 family (sea figure 2.4). Zebra2 cameras combine the capabilities of a machine vision and an IP camera, and are used in applications that require real-time viewing in addition to image post processing. They have two main interfaces, and deliver uncompressed video over an HD-SD connection and also MJPEG compressed or uncompressed images over Gigabit Ethernet [15]. All Zebra2 cameras are "Power Over Ethernet" enabled.

In particular, this specific model has a 2.0 MP color CCD from Sony and a 8mm lens is used. On table 2.4 and in figure 2.5 the most significant specifications of this device are described.

Figure 2.4: Point Grey ZBR2-PGEHD-20S4C.

Resolution	1624 x 1224
Frame Rate	25 FPS with HD-SDI
Megapixels	2.0 MP
Chroma	Color
Sensor	Sony ICX274 CCD
Lens Mount	CS-mount
Serial Port	1 RS-485 (dedicated pins)
Interface	GigE PoE, HD-SDI
Operating voltage	8 V DC to 30 V DC
Power consumption	Max. 6 W
Weight	150 grams
Dimensions (H x W x L)	44 mm x 44 mm x 87.5 mm

Table 2.4: Point Grey ZBR2-PGEHD-20S4C technical specifications [15].

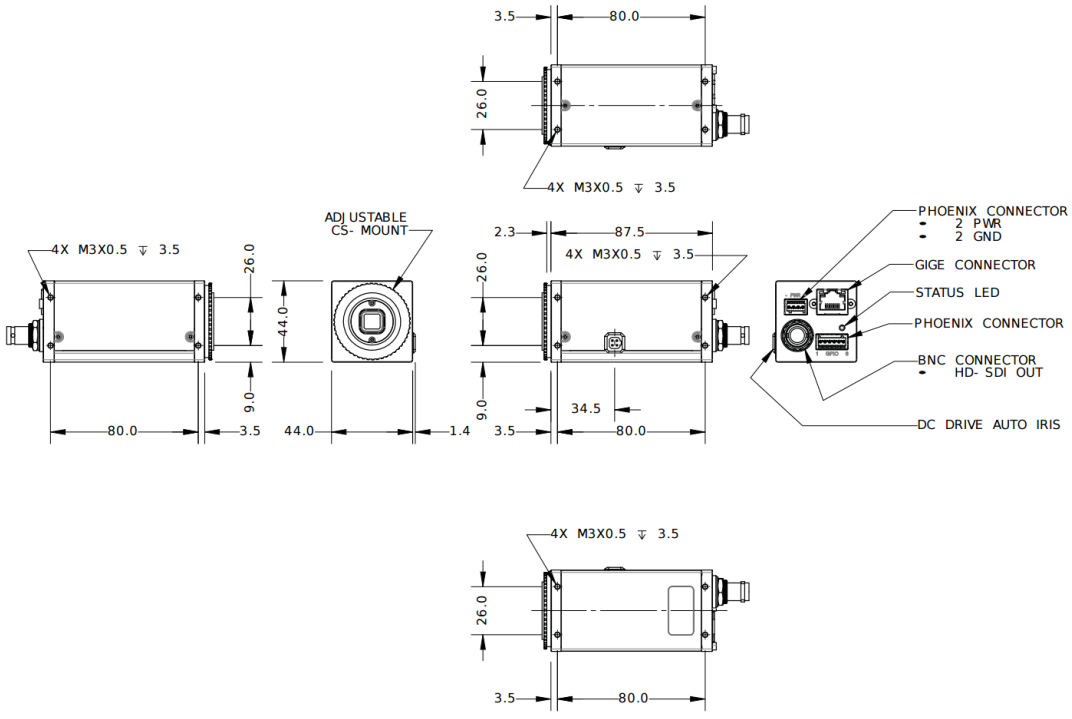
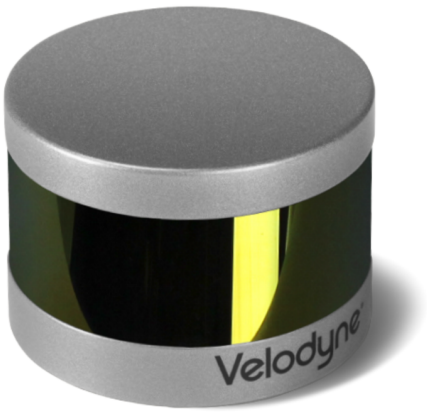


Figure 2.5: Point Grey ZBR2-PGEHD-20S4C dimensional drawings.

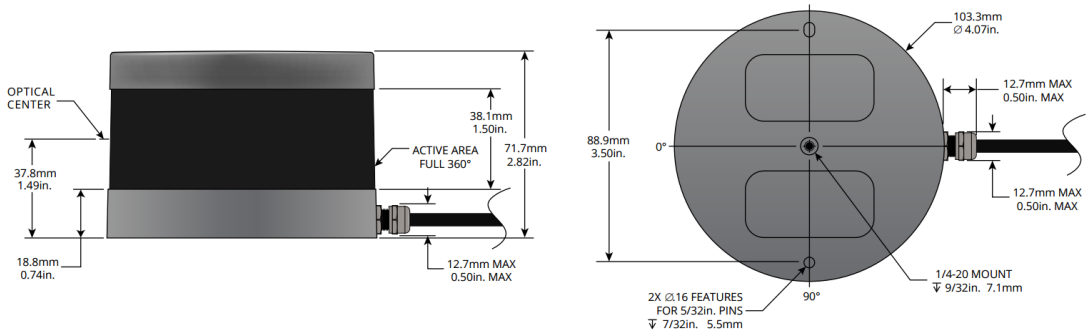
2.2.4 Velodyne Puck VLP-16

This is the most advanced LIDAR used. Its usage was made possible by the cooperation of Institute of Systems and Robotics (ISR), unfortunately is was only available in a later stage of this work. The Puck VLP-16 uses 16 channels, 300,000 points/second, 360° horizontal field of view and a 30° vertical field of view generating a 3D point cloud. This

device is very cost-effective, light weight, compact and designed for mass production which makes it suitable for autonomous vehicles, robotics and mobile terrestrial 3D mapping applications [28]. Figure 2.6 shows this device and its dimensions and table 2.5 discriminates its main specifications.



(a) Velodyne Puck VLP-16.



(b) Velodyne Puck VLP-16 dimensions.

Figure 2.6: Velodyne Puck VLP-16 views.

This device was only available in a latter stage of this work and its usage was made possible by the ISR and resulted from a collaboration between this institute and LAR.

2.3 Auxiliary equipment

2.3.1 Cisco SD205 Switch

Cisco SD205, figure 2.7, is a 5-port 10/100 Mbps switch. This switch provides non-blocking, wire-speed switching at a speed of 10 or 100 Mbps. All ports support auto-negotiation to connect the devices at the same network speed, and auto MDI/MDI-X crossover detection. Each port negotiates independently for best speed at either half- or full-duplex mode, for up to 100 Mbps of bandwidth per port. Fast store-and-forward switching prevents damaged packets from being passed into the network. Table 2.6 describes the most important specifications of this device.

Light source	infrared light (903 nm)
Laser class	1 (IEC 60825-1:2007 & 2014)
Field of View (Vertical)	+15.0° to -15.0°
Field of View (Horizontal)	360°
Rotation Rate	5 Hz to 20 Hz
Angular Resolution (Vertical)	2.0°
Angular Resolution (Horizontal/Azimuth)	0.1° to 0.4°
Operating range	Up to 100 m
Accuracy	±3 cm (Typical)
Operating voltage	9 V DC to 18 V DC
Power consumption	Typ. 8 W
Weight	830 grams
Dimensions	103 mm Diameter x 72 mm Height
Interface	100 Mbps Ethernet Connection

Table 2.5: Velodyne Puck VLP-16 technical specifications [28].

Figure 2.7: Cisco SD205 5-port 10/100 Switch. [13]

Ports	5 RJ-45 10/100 Mbps ports
Cabling Type	Cat5 Ethernet
Power	DC 12V/0.5A
Weight	0.23 kg
Dimensions (WxHxD)	93 mm x 30 mm x 90 mm

Table 2.6: Cisco SD205 technical specifications [14].

2.3.2 Nexus P-2308H4/HR4

The Nexus P-2308H4/HR4 (figure 2.8) will be used has the processing unit for ATLAS-CAR 2. It was two Intel XEON E5 processors, 32 Gb RAM memory, 6 Tb HDD storage capacity and a Nvidia Quadro graphics card. The main specifications of this equipment are described in table 2.7.

Figure 2.8: Nexus P-2308H4/HR4 server. [19]

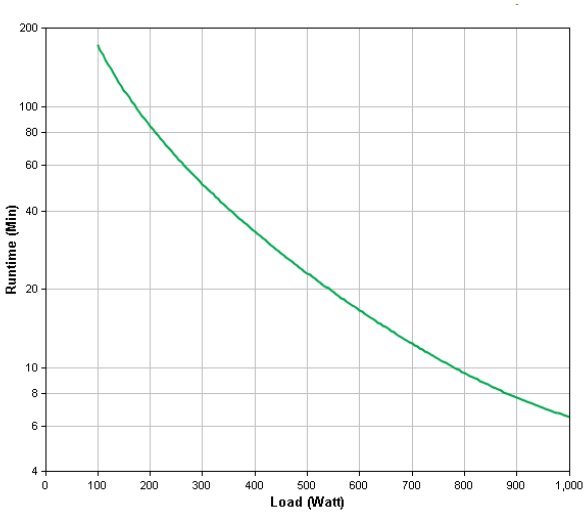
Processor	2x Intel XEON E5
Memory	32 Gb
Storage	6 Tb HDD
Network	2x Gigabit onboard
Graphics	Nvidia Quadro
Dimensions (mm)	178W x 437H x 648L

Table 2.7: Nexus P-2308H4/HR technical specifications [19].

2.3.3 APC Smart-UPS 1500 VA

The APC Smart-UPS 1500 VA (figure 2.9a) is the UPS that will be installed in ATLAS-CAR 2 temporarily to power the processing unit and the monitor. It has a maximum power output of 1500 VA, 230 V nominal output voltage, the figure 2.9b shows the runtime depending on the power output.

(a) APC Smart-UPS 1500 VA. [12]



(b) UPS runtime depending on the power output. [12]

Figure 2.9: Sick LMS151 views.

2.4 Software

In this section the software used in the present thesis is described. Since there is large experience at LAR using the Robot Operating System (ROS) framework because it has been used on Atlas and other major projects, and because the previous work this thesis is based on was also implemented using ROS, all the software developed in this work is based on a ROS architecture. The tools required to interact with the sensors use ROS packages that handle the communication and return the laser scans in a `Point Cloud` data formate or as a `Laser Scan` message. The data processing is done using Point Cloud Library (PCL) functions. Sensor calibration is accomplished using the multi-sensor calibration package developed by Silva [2].

2.4.1 ROS

ROS or Robot Operation System is a framework for robot software development. It is a collection of tools and libraries written in C++, Python, Octave, or LISP, [9], designed to create complex and robust robot behaviour across a wide variety of robotic platforms [24]. ROS has over 3000 published packages including industrial-quality drivers and capabilities to provide low-level device control, communication between processes, package management and more. It is open source and is supported by Ubuntu Linux, other operation systems like Fedora Linux, Windows and Android are supported by the community [9].

The fundamental concepts of the ROS implementation are nodes, messages, topics, and services [9]. In the next paragraphs these concepts will be briefly explained, as well as four packages, essentials in this work: `rviz`, `tf`, `roslaunch` and `rosbag`.

Nodes

Nodes are software modules that perform computation, typically a system is running many processes (nodes). Nodes communicate with each other by **messages** published on **topics**. If a node is interested in the messages of a certain kind of data it subscribes the respective topic. Usually publishers and subscribers are not aware of each others' existence.

Messages

Messages are a way for nodes to communicate with each other. A message is a data structure that can be composed by primitive types and constants, by arrays of primitive types by other messages or by arrays of other messages. Messages are identified by the topic were they are being published or subscribed.

Topics

Topics are used by ROS to identify a certain message. They are simply strings like "map". A node that sends messages to a topic is called publisher and a node that receives messages is called subscriber. A single node can both subscribe and publish to topics. To deal with the asynchronous nature of ROS, publishers and subscribers have message queues which store messages until the defined queue limit is reached.

Services

The publisher-subscriber method is a flexible communication solution between nodes but it is only in one direction and it is not appropriate for synchronous transactions. The solution is to use a service. Services are defined by a name (string) and a pair of strictly typed messages, one for the request and other for the response (bidirectional communication). Unlike topics, services can only have one node advertising it. This is analogous to web services defined by URIs.

Roslaunch

The roslaunch package allows to execute multiple nodes locally or remotely as well as to set parameters. It has tools to respawn processes that have died or kill all processes launched if a specific one dies [23]. This package reads one or more XML configuration files that specifies which nodes to launch and the parameters to set.

In this thesis, roslaunch is used to launch the hardware drives, `rosbag`, the free space detection node, and `rviz`.

Rviz

Rviz is a 3D visualization tool for ROS. It provides a GUI (sea figure 2.10) for the visualization of robotic platforms and of a wide range of sensor's data types. Rviz also includes a library that allows most of its functionalities to be used by other applications.

In this work, Rviz will be used to display the raw and the processed data from the LIDARs described in the section 2.2.

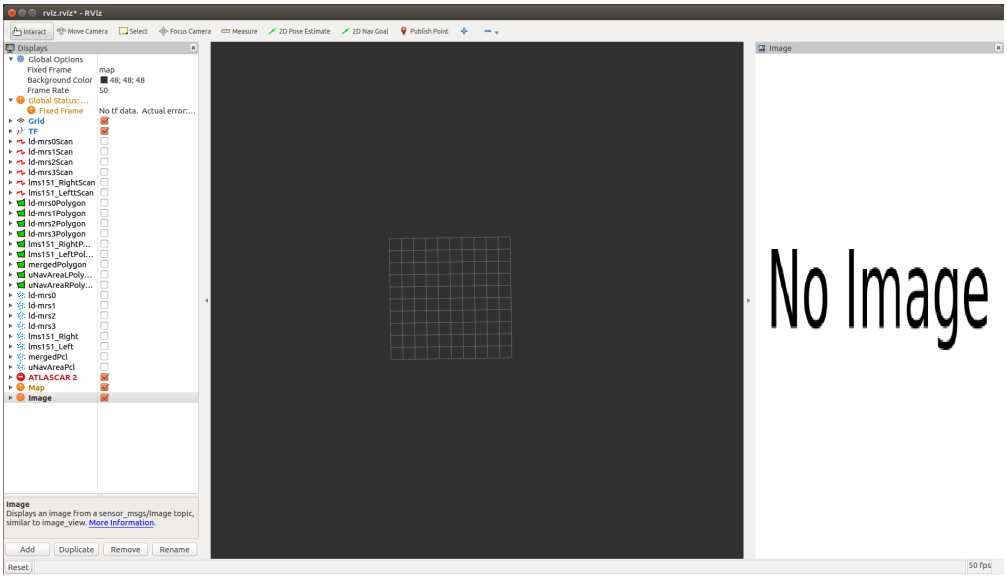


Figure 2.10: Rviz GUI. On the left is the panel to select the data to display. On the right there is an image from the camera and, on the center, the data from the LIDAR sensors is shown.

Tf

The tf library is a ROS package that lets the user keep track of multiple coordinate frames and geometric transformations over time. It has tools to know the relation between two different coordinate systems at a specific time, to broadcast the position of coordinate system, visualize all the frames [5]. In this work the tf package is used to publish the positions of each sensor's coordinate system relative to the reference coordinate system, named "map".

Rosbag

The rosbag package provides a set of tools for recording and playing back bags. A bag is a file format for storing ROS message data. Bags are widely used for data logging allowing the offline use of the data [21]. Rosbag provides command-line tools for working with bags as well as code APIs for reading/writing bags and it is very useful to create datasets that later can be visualized, labelled and stored for future use[22].

2.4.2 PCL

PCL is an open source library for n-D Point Clouds and 3D geometry processing. It is available in multiple platforms — Linux, MacOS, Windows, Android and iOS — and is fully integrated with ROS. PCL is written in C++ and with efficiency and performance in mind on modern CPUs [8].

PCL incorporates multiple 3D processing algorithms including: filtering, feature estimation, surface reconstruction, model fitting, segmentation, registration, etc. Each set of algorithms is defined via base classes that attempt to integrate all the common functionality used throughout the entire pipeline, thus keeping the implementations of the actual algorithms compact and clean. The basic interface for such a processing pipeline in PCL is: [8]

- Create the processing object (e.g., filter, feature estimator, segmentation);
- Use setInputCloud to pass the input point cloud dataset to the processing module;
- Set some parameters;
- Call compute (or filter, segment, etc) to get the output.

To simplify the development, PCL is split in multiple smaller libraries that can be compiled separately [8]:

- `libpcl_filters`;
- `libpcl_features`;
- `libpcl_io`;
- `libpcl_segmentation`;
- `libpcl_surface`;
- `libpcl_registration`;

- `libpcl_keypoints`;
- `libpcl_range_image`.

2.5 Software extensions from LAR

2.5.1 Multisensor calibration package

The multisensor calibration package was developed at LAR and is described in detail in [2]. It is a GUI, figure 2.11, that allows the calibration of multiple sensors, both LIDAR and vision-based. With two working modes, the user can choose between automatic or manual calibration and witch sensors to calibrate. By moving a ball in front of the sensors the transformation matrices between the sensors and a reference is found and saved to text files; the maximum error and the ball diameter can also be set by the user.

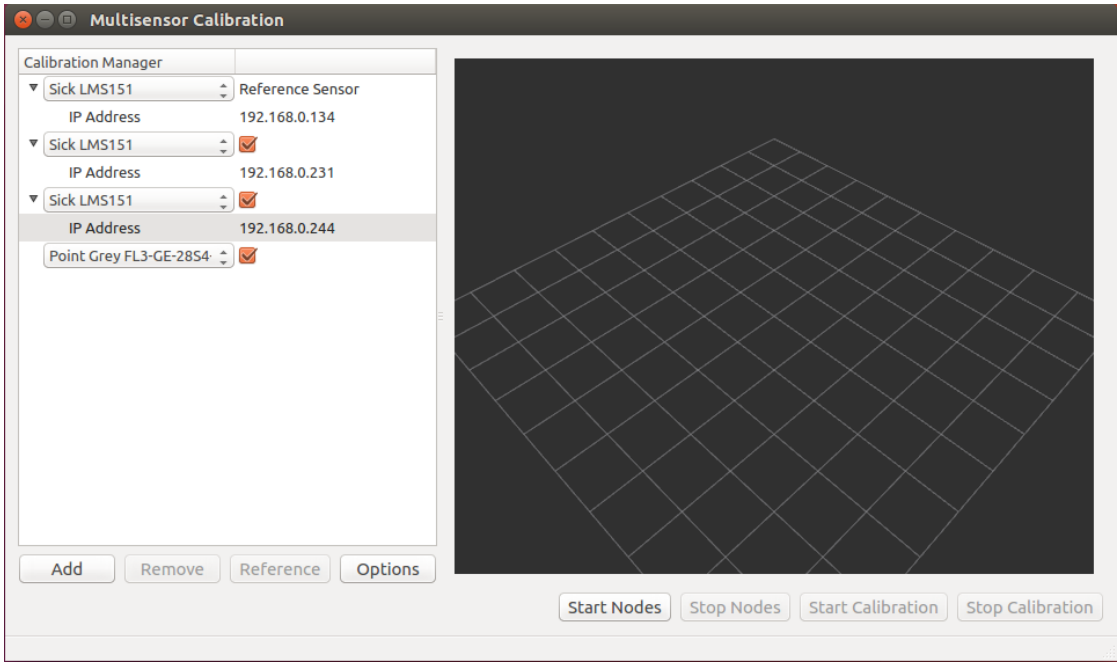


Figure 2.11: Multisensor calibration GUI [2].

The ball's center is detected in each sensor using custom made algorithms. For each sensor to calibrate, a set of nodes are launched that acquire data from the sensors, process it and publish the ball's center in the device coordinate system, see figure 2.12.

For the LIDAR sensors, the ball's center is found by segmenting the laser scan in clusters and search for a circle in each cluster, the z coordinate of the center is found knowing the ball's diameter and applying the following equation: $z = \pm\sqrt{\left(\frac{d}{2}\right)^2 - r^2}$ where d is the sphere diameter and r is the circle radius. To solve the $\pm$ ambiguity prior information about the sensor location relative to the ball's equator as to be supplied. When the LIDAR has multiple scan planes, as is the case of the Sick LD-MRS, the same algorithm is applied to each scan plane and the final ball's center is the mean of the centers found in each scan plane. In vision-based sensors the ball's center is found in

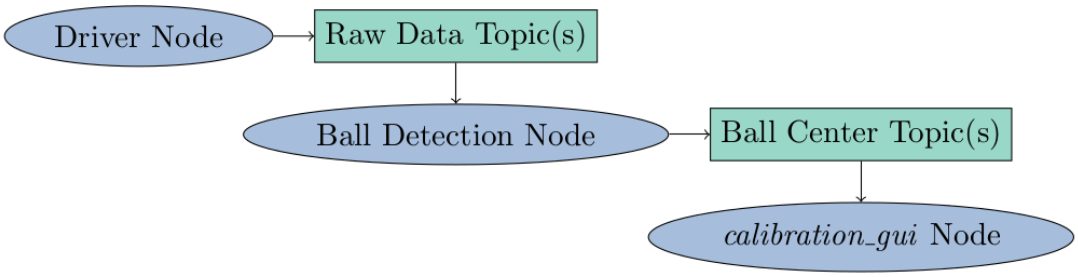
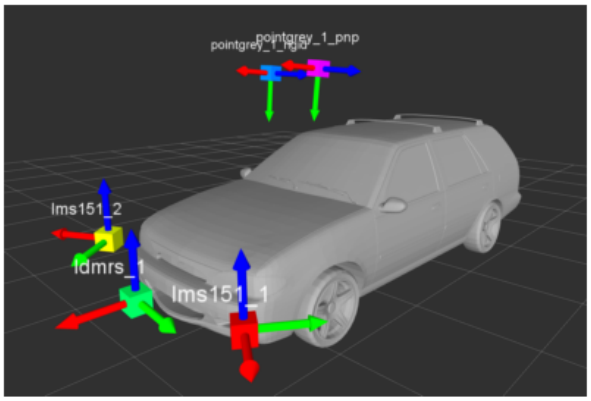


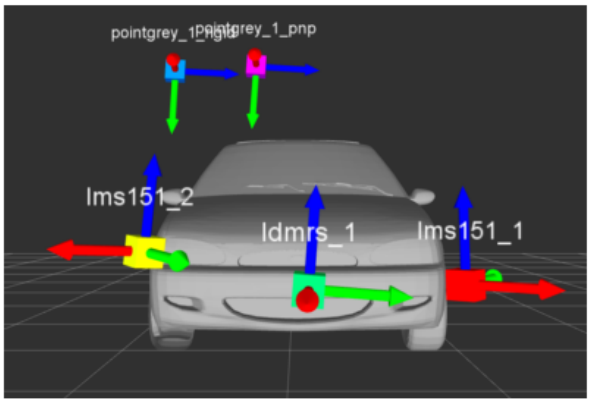
Figure 2.12: Generic ROS architecture implementation for each sensor. Ellipses represent nodes, and rectangles represent topics [2].

the image by using a Hough Circle Transform algorithm, implemented in OpenCV, and then is converted to real world coordinates using the intrinsic matrix of the camera. A separated node subscribes the sphere centers published by the sensor's nodes, checks if the point is valid (respects the minimum distance between calibration points and the maximum ball center displacement error) and, if it respects this conditions, adds it to a point cloud. For each calibrated sensor a point cloud is generated with the ball's center detected by the sensor in its coordinate system. The extrinsic transformation between each sensor's coordinate system if found using a 3D rigid body transformation algorithm from PCL library. The package structure was developed keeping in mind the variety of sensors used in AD so new sensors can be easily added to the package. Figure 2.13 shows the results of the calibration package used to calibrate two Sick LMS151, one Sick LD-MRS and a Point Grey Flea 3 camera in ATLASCAR 1.

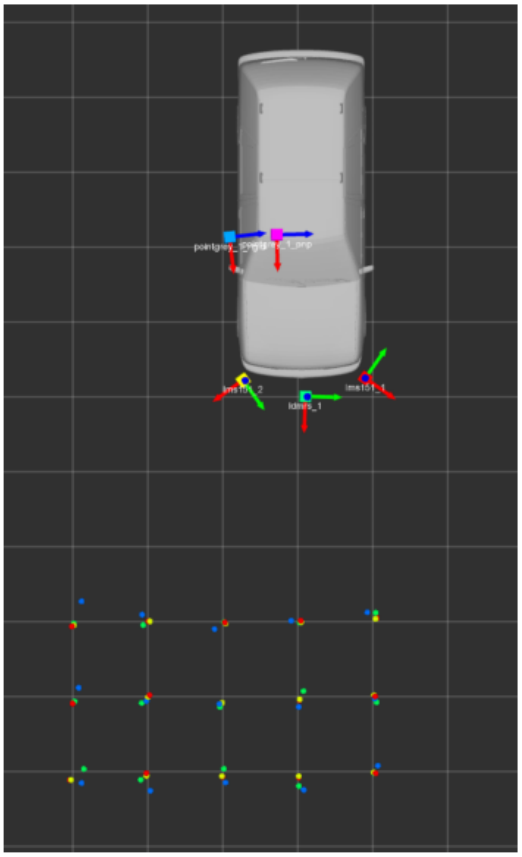
This work was published in the Robotics and Autonomous Systems journal in 2016 [1].



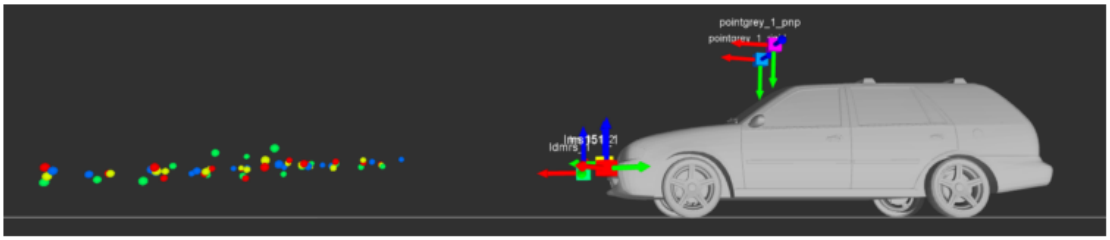
(a) 3D view.



(b) Front view.



(c) Top view with calibrated point clouds.



(d) Side view with calibrated point clouds.

Figure 2.13: Sensor calibration on ATLASCAR 1. SICK LMS151(A) is shown in red, SICK LD-MRS400001 in green, SICK LMS151 in yellow, Point Grey camera using the 3D rigid body transformation method in blue and Point Grey camera using the extrinsic camera calibration method in magenta [2].

Chapter 3

Hardware Setup on ATLASCAR 2

One of the main objectives of this thesis is to instrument ATLASCAR 2 with LIDAR and vision-based sensors. For that, it was necessary to design and install supports and cables had to be inserted in the vehicle. This chapter describes the process to select the sensors location, find mounting points on the vehicle's chassis, design the required supports, find passageways to run cables from the vehicle's front to its trunk as well as find a power source to power all devices.

The process to install all the necessary fixtures and supports where the camera and the LIDARs will be placed as well as the fixtures for an Ethernet switch are described. The way the devices are powered and the communication infrastructure are also explained.

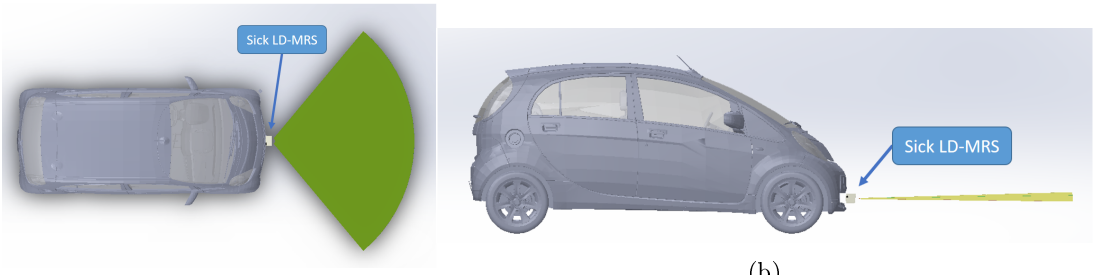
3.1 Mechanical design and assembly

This section explains how the hardware was installed in ATLASCAR 2. From the selection of the LIDAR sensors position up to the construction and installation of the fixtures, and covering also the research for mounting points on the chassis as well as the design of 3D printed parts to support the camera and the Ethernet switch.

3.1.1 Sensors location

Since the vehicle is moving forward most of the time, being aware of the free space in front and on the sides of ATLASCAR 2 is essential for AD and even for ADAS. Hence, two Sick LMS151 and a Sick LD-MRS were installed in the vehicle's front. The two Sick LMS151, have an aperture angle that will allow to monitor the front and the sides of the vehicle while the Sick LD-MRS will detect objects in front of the vehicle. In a latter stage of this work, a Velodyne Puck LIDAR was also tested in two different locations: mounted on the roof top, to take de most advantage of its 360° horizontal field-of-view and in the vehicle's front to take most advantage of its vertical field-of-view.

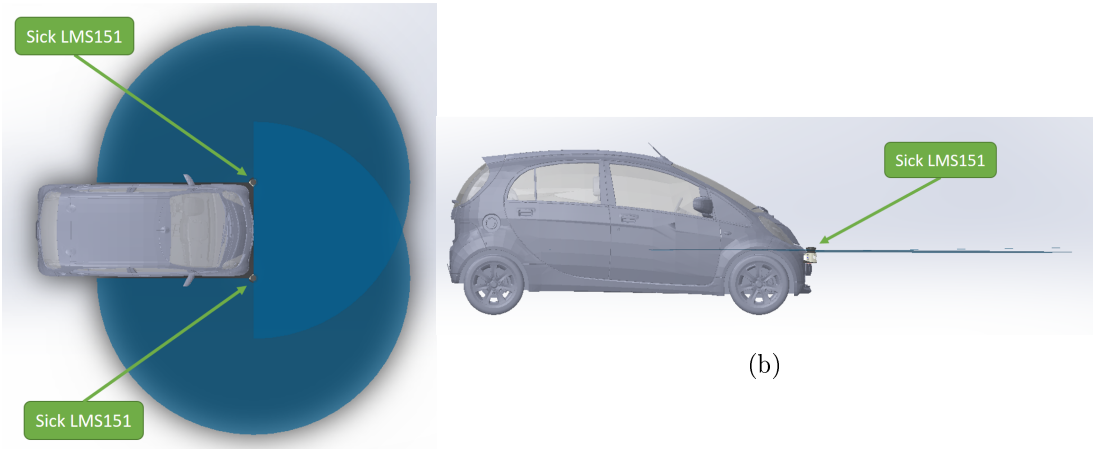
Sick LD-MRS, because of its long range and four scan planes, is ideal for obstacle detection and even to detect road inclinations such as ramps or bumps that can be seem as an obstacle by other planar LIDARs. Taking this into account, and because it has only an 85° aperture horizontal angle, the best place to install it is in the middle of the track width in the front of the vehicle, figure 3.1. Its vertical distance to the road will be defined by the mounting points on the vehicle's frame as described in subsection 3.1.2.



(a) (b)

Figure 3.1: Sick LD-MRS location on ATLASCAR 2.
(a) Sick LD-MRS location on ATLASCAR 2 - Top view.
(b) Sick LD-MRS location on ATLASCAR 2 - Side view.

The Sick LMS151 LIDARs have a 270° aperture angle. To take the most advantage of the 270° aperture angle of Sick LMS151 LIDARs they were installed in the vehicle's front left and front right ends at a certain distance from the vehicle's bumper to avoid scanning any part of the vehicle, figure 3.2a. Besides that, placing the lasers lower than the top part of the wheels might result in them being detected by the LIDARs when the vehicle is turning, reducing the laser field-of-view. Therefore the laser should be installed immediately above the front wheels, as showed in figure 3.2b.

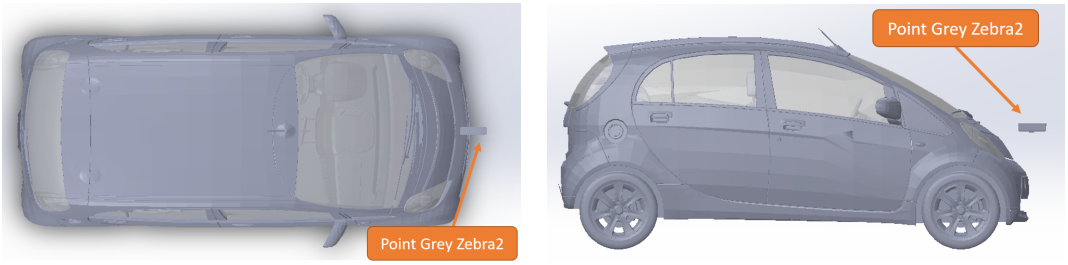


(a) (b)

Figure 3.2: Sick LMS151 location on ATLASCAR 2.
(a) Sick LMS151 location on ATLASCAR 2 - Top view.
(b) Sick LMS151 location on ATLASCAR 2 - Side view.

As already mentioned, Point Grey Zebra2 camera will only be used for visualization. Therefore, its location is not a major concern. Ideally, the camera should be installed inside the vehicle in order to protect the equipment from damage due to environmental agents like water, dust or even due to mosquitoes or other small flying animals that may hit the camera when the vehicle is moving. However its installation in an advantageous position inside the vehicle proved to be very difficult and time consuming. Therefore, the camera was installed outside the vehicle, on the fixtures developed for the Sick LMS151

sensors, further described in the subsection 3.1.3. Unfortunately this meant the camera had to be protected from the agents mentioned before. Furthermore, the camera has to be positioned high enough to get an interesting field-of-view. Figure 3.3 shows a proposed position for the equipment.



(a) (b)

Figure 3.3: Point Grey Zebra2 location on ATLASCAR 2.
(a) Point Grey Zebra2 location on ATLASCAR 2 - Top view.
(b) Point Grey Zebra2 location on ATLASCAR 2 - Side view.

3.1.2 Sick LD-MRS400001 fixtures

As mentioned before, the Sick LD-MRS400001 will be installed centred in the front of the vehicle. After an analysis of the vehicle's chassis it was possible to conclude that the best mounting points were the ones shown in figure 3.4. This position has the advantage of using an existing opening in the front bumper so there is no need to modify it to fix the laser. Fixing this scanner to those points required some custom parts to be developed and other acquired externally. Figure 3.5 shows all the parts required to fix this device.

This fixture system has the particularity of allowing the sensor to be adjusted in two angles — pitch and roll — by loosen the screws one and two it is possible to adjust the roll angle and by loosen the screws three and four adjusts the pitch angle, figure 3.5.

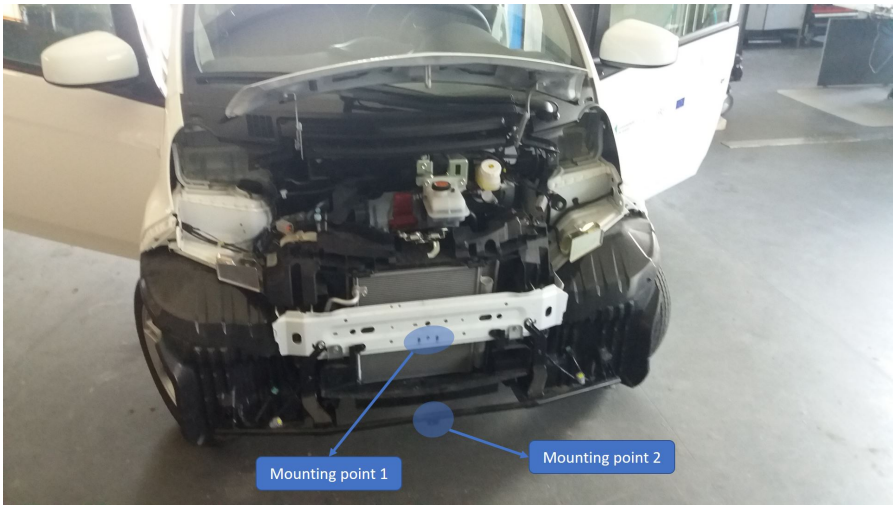
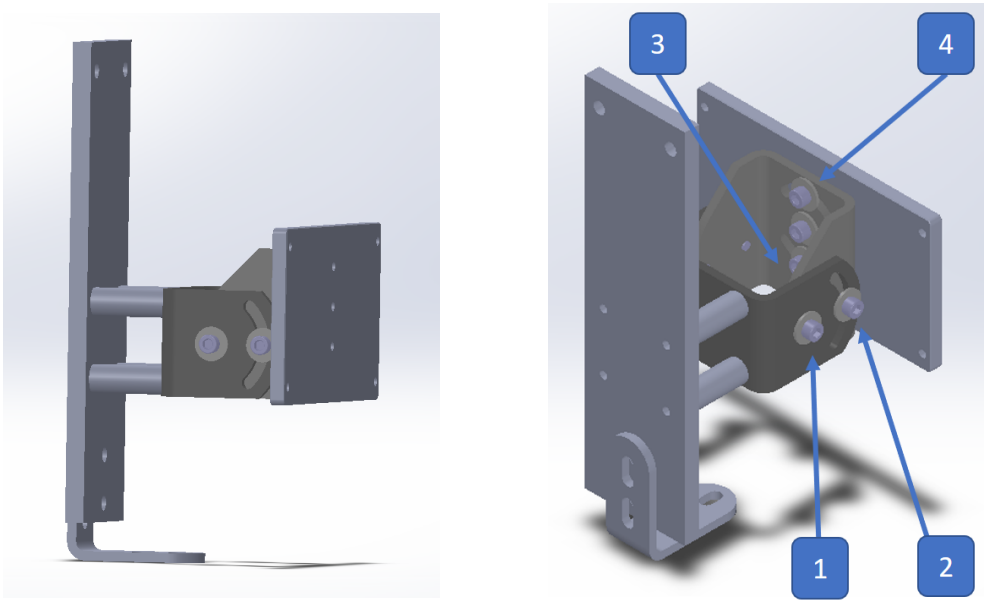


Figure 3.4: ATLASCAR 2 chassis mounting points.



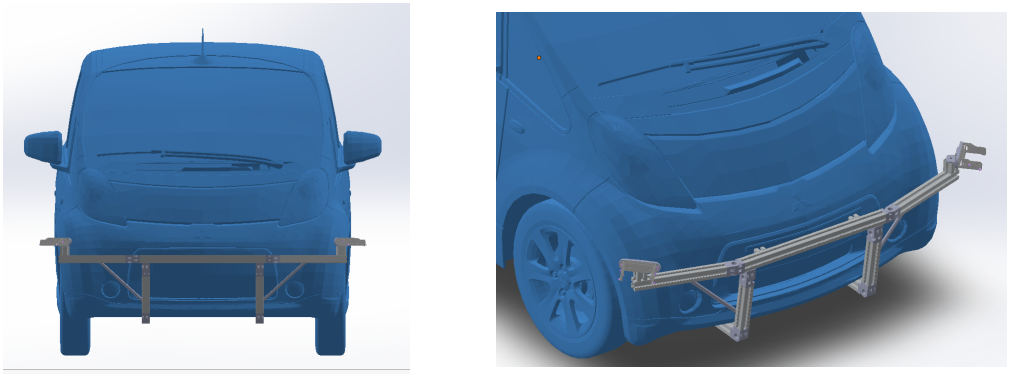
(a) Sick LD-MRS fixtures - View 1.

(b) Sick LD-MRS fixtures - View 2.

Figure 3.5: Sick LD-MRS fixtures.

3.1.3 Sick LMS151 fixtures

In order to fix the two Sick LMS151 in the desired positions, mentioned in subsection 3.1.1, several solutions were considered. One was to reinforce the front bumper with a steel sheet and fix the scanners directly to the bumper. This solution was discarded because it would not provide enough mechanical resistance to properly fix the sensors and would be very hard to materialize. The next explored solution was to fix the lasers directly to the chassis, as done in ATLASCAR 1. Unfortunately there were no mounting points near where the sensors must be fixed, so another solution had to be found. The final solution was to install a set of bars fixed directly to the vehicle's chassis (figure 3.6) providing a modular fix system that allows for an easy installation of any sensor.

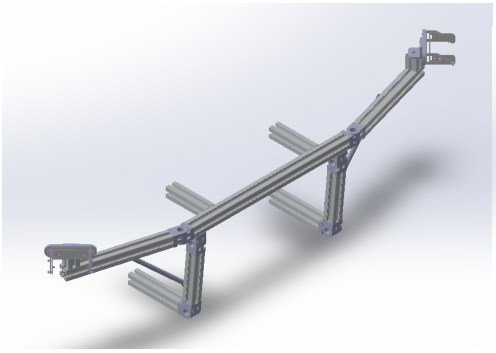


(a) Sick LMS151 fixtures - View 1.

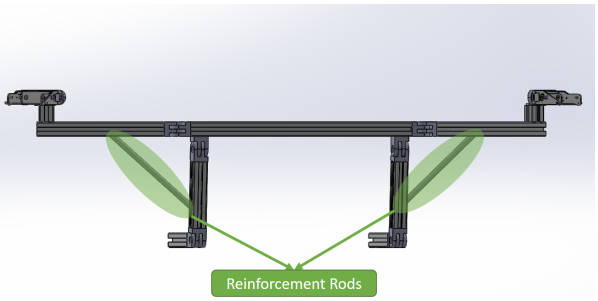
(b) Sick LMS151 fixtures - View 2.

Figure 3.6: Sick LMS151 fixtures on ATLASCAR 2.

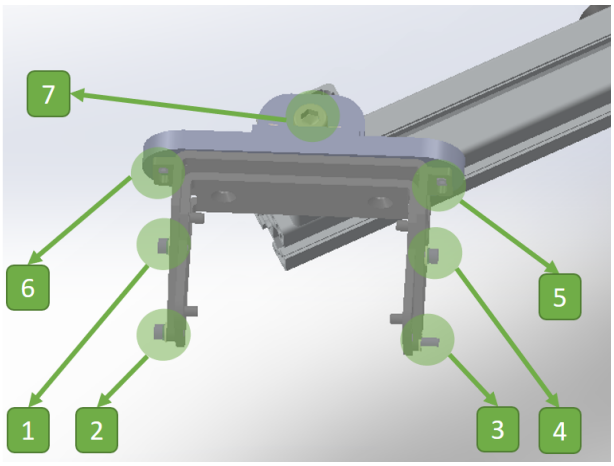
This solution provides a modular fixing system witch allows for the installation of many types of sensors in a large number of different positions. It is easy to build and install, the sensors can be easily placed or removed and it does not require many extra holes in the vehicle's chassis or bumper. The structure (figure 3.7) was built entirely from standard aluminium profile with a 40x40 mm section and standard linking accessories. It is fixed directly to the chassis in points one, two, three and four represented on figure 3.8, providing high mechanical resistance. Furthermore, to prevent possible vibrations due to the long length of the right and left bars two steel rods (marked in green in figure 3.7b) were added. This setup allows for both sensors to be adjusted along the three rotation angles — yaw, pitch e roll — independently. The screws one to four allow to adjust the laser's pitch angle, and the screws five and six adjust the roll angle. The yaw angle is adjusted with screw seven, showed in figure 3.7c.



(a) Sick LMS151 fix system - View 1.



(b) Sick LMS151 fix system - View 2.



(c) Sick LMS151 fix system - Close-up view of the angles adjusting mechanism.

Figure 3.7: Sick LMS151 fix system.

The rods used to reinforce the structure's right and left bars are made of steel tube and connected to the profile bars by a pair of rod-ends each. To connect the rod-ends to the tube two steel parts with an internal thread were welled. Figure 3.9 show the rods prior to the welding, and after finished.

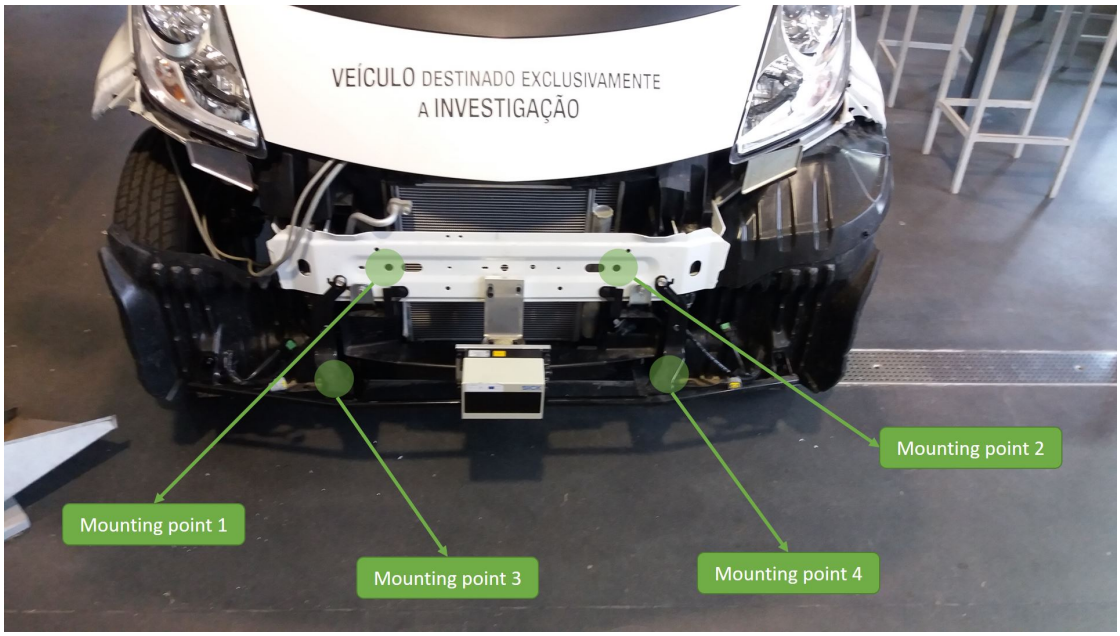


Figure 3.8: Sick LMS151 fixing system mounting points.

(a) Rods before begin welded. (b) Rods during the weld process.

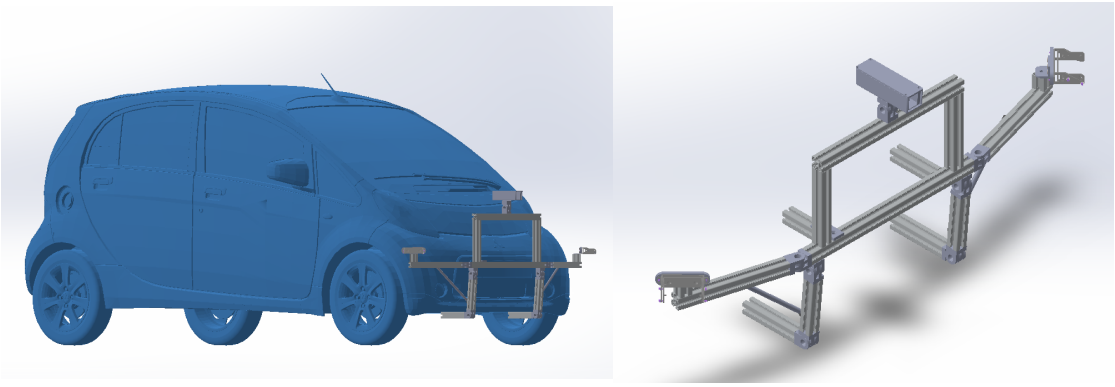
(c) Rod assembled before painting.

Figure 3.9: Rods manufacturing.

3.1.4 Point Grey Zebra2 camera fixtures

As mentioned in the previous subsection, the setup to mount the two Sick LMS151 is very versatile and can be used to mount any other sensor. This meant the camera could be fixed on this support as well. Because the support is in a lower position than the

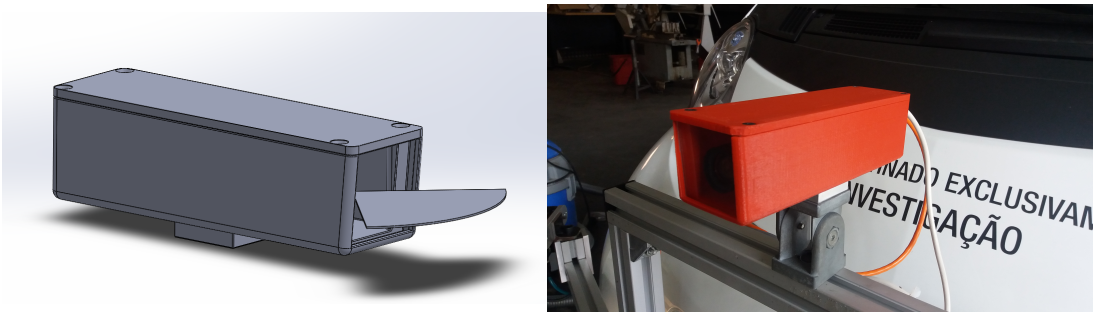
proposed position for the camera, three additional bars had to be added to the setup in order to fix the camera in a higher position. The final setup is shown in figure 3.10.



(a) (b)

Figure 3.10: Final fixtures setup for Sick LMS151 and Point Grey Zebra2.
(a) Sick LMS151 and Point Grey Zebra2 fixtures on ATLASCAR 2.
(b) Sick LMS151 and Point Grey Zebra2 fixtures.

Since the Point Grey Zebra2 camera is not ready for outdoor usage, a protecting case was also developed (figure 3.11). To have a versatile design, the case was projected keeping in mind that different cameras are used at LAR with different lenses. With this in mind, the case was projected for two different Point Grey camera models (Zebra2 used in this thesis and Flea3 also used in other works at LAR) with different types of lens.



(a) (b)

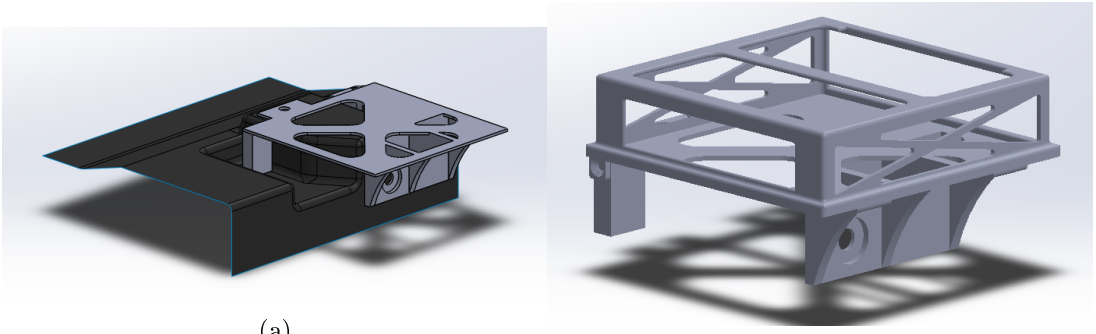
Figure 3.11: Point Grey Zebra 2 protective case.
(a) Point Grey Zebra 2 protective case - CAD model.
(b) 3D printed Point Grey Zebra 2 protective case.

3.1.5 Cisco SD205 switch fixture

Since all sensors used have an Ethernet interface, the Cisco SD205 switch is a key element in the communication infrastructure. It connects all the sensors to the processing unit. Because most of the sensors use non-standard connectors, the cables available have specific lengths. In this case, the cables available for some of the sensors did not have enough length to run from the vehicle's front to its trunk. Furthermore, passing 4 cables

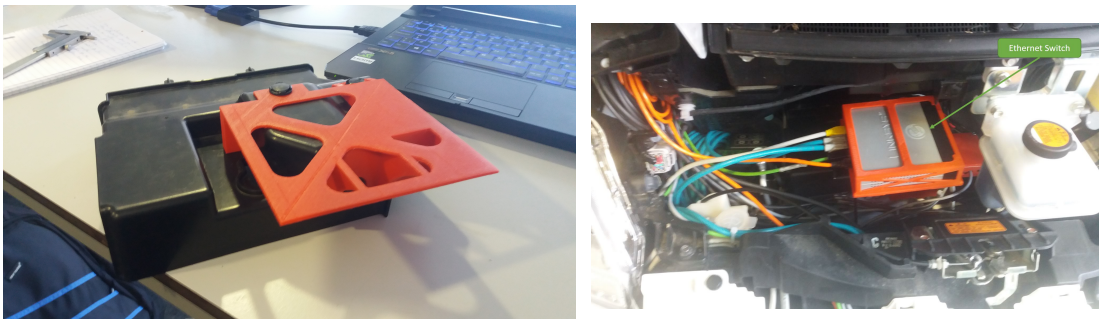
from the sensors to the trunk is more difficult than passing just one. As a result the switch was placed under the vehicle's hood, on the battery plastic cover. This meant only one Ethernet cable needs to run from the trunk to the hood compartment, and no longer cables have to be acquired.

Unfortunately, the battery plastic cover was not flat and there was not enough base area to properly fix the switch. This problem was solved with a custom made 3D printed case, figure 3.12 and 3.13. The box base was designed to fit on the battery case and is fixed by three quick fasteners removed from the ATLASCAR 2 front bumper (not necessary after the installation of the front bars). Then, a top cover ensures the switch remains in a fixed position.



(a) (b)

Figure 3.12: Switch fix system - CAD model.
(a) Switch custom made base on the batterry plasctic cover.
(b) Switch custom made box.



(a) (b)

Figure 3.13: Switch fix system - 3D printed.
(a) Switch custom made base on the batterry plasctic cover.
(b) Switch custom made box.

3.1.6 Roof bars

The last modification on ATLASCAR 2 was the installation of roof bars to allow for other sensors, like the Velodyne Puck VLP-16 to be installed. Because the vehicle does not have any longitudinal bars in the roof it was necessary to install two crossbars, like

the ones used to transport bicycles, and on top of those two longitudinal bars made of standard aluminium profile with a 40x40 mm section (same as the one used on the front bars) were placed. Because there were no standard crossbars for the specific vehicle model the job of installing the cross bars was done in a specialised auto-shop and the longitudinal bars were installed in house. Figure 3.14 shows the final result.

Figure 3.14: ATLASCAR 2 rooftop bars.

3.1.7 Processing unit, monitor and UPS installation

The processing unit and the UPS were installed in the vehicle's trunk, see figure 3.15. The monitor was placed on the back of the front passenger seat fixed to the seat's headrest, figure 3.16.

3.1.8 Final assembly

The final project is shown in figure 3.17. It is a modular and very versatile solution made of standard aluminium profile and fixing accessories. That does not require significant modifications on the vehicle and most of them are fully reversible. It provides quick and easy sensor installation and results in a reliable platform for future works.

After the project was finalized, the required materials were ordered and some parts started to be produced. As soon as the materials were available and all the parts machined, the assembly process started. Assembling all the components was straight forward, since no significant changes were made in the vehicle's body. First the Sick LD-

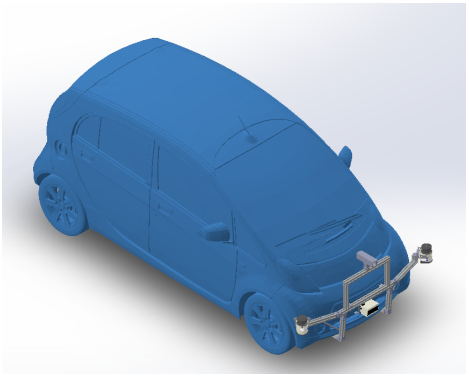
Figure 3.15: ATLASCAR 2 trunk with UPS and the processing unit.

(a) (b)

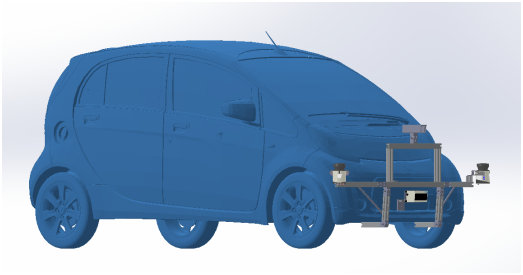
Figure 3.16: Monitor installation on ATLASCAR 2.
(a) Monitor installed on ATLASCAR 2.
(b) Monitor fixture to the front passenger seat of ATLASCAR 2.

MRS fix system was assembled because it has to be installed in the car without its front bumper. Next, the bars that connect the Sick LMS support system to the front of the chassis were installed. Next, the bumper can be installed, but first, two holes had to be drilled on it in order to give space for the bars to connect from the chassis to the Sick LMS support. The next step was to install the front bars that support the sensors. Finally the two bars that fix to the bottom of the chassis were installed and connected by other set of two bars to the front bars and the two rods were fixed, finalizing the support system for the two Sick LMS151. Lastly, the set of bars that fixes the camera were installed on the Sick LMS support, and the camera placed inside its protective case. Figure 3.18 shows the final result.

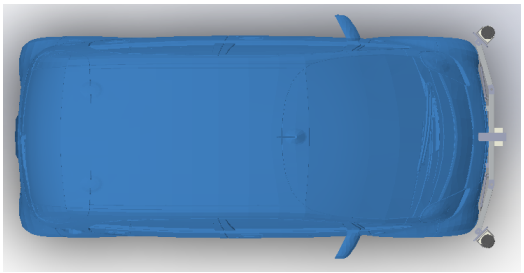
(a) Sensors final setup - Front view.



(b) Sensors final setup - Perspective view.



(c) Sensors final setup - Side view.



(d) Sensors final setup - Top view.

Figure 3.17: Sensors final setup.

(a) LIDAR's final setup.

(b) Hardware setup - View 1.

(c) Hardware setup - View 2.

(d) Hardware setup - View 3.

Figure 3.18: Hardware final setup.

3.2 Electrical Project

This section describes the installation process of the power and communication cables and the power distribution board. At this stage, the sensors will be powered by the 12 V DC car's circuit through a connection found in the engine compartment. The processing unit and the monitor are powered by an external UPS placed in the trunk that gives the the processing unit about 20 minutes autonomy. Simultaneously to this thesis, other projects are being developed with the goal of powering all the devices from the car's high voltage circuit using the traction batteries as power source.

3.2.1 Power distribution circuit

As was already mentioned, the sensors are powered by the 12 V DC car's circuit. The power comes from the engine compartment and enters the power distribution board. There, a 16A DC circuit breaker protects the circuit and serves as a ON/OFF switch. The circuit breaker is connect in series with a relay to ensure the sensors are only powered on when the car's Lithium-ion battery is connected, otherwise the lead battery would discharge very quickly. The relay is connected to a power line on the vehicle's hood compartment that is only active when the car is powered from the Lithium-ion battery, see figure 3.19.

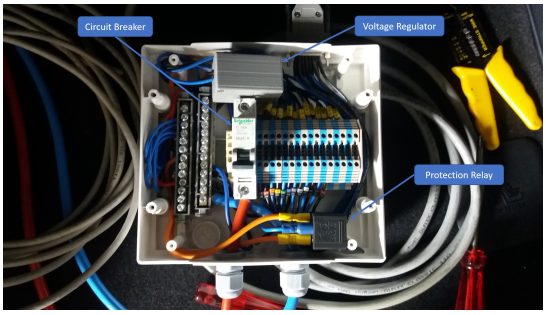
(a) Relay coil connection on ATLASCAR 2. (b) Relay coil connection close-up.

Figure 3.19: Protection relay coil connection to the vehicle.

The power distribution board (figure 3.20) has two output connectors: a DB 15 female connector that connects to the cable that runs to the vehicle's front and a pair of female banana connectors to power other devices.

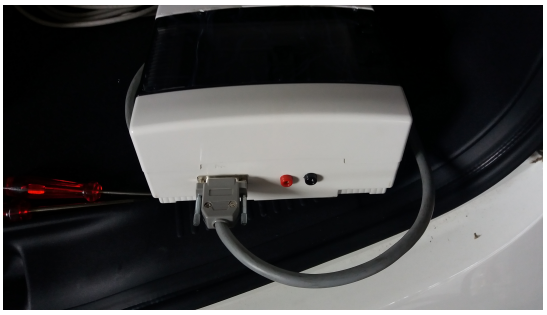
Although the car's low voltage circuit was referred as a 12 V DC circuit, in order to be able to fully charge the lead battery, its real voltage is 14.5 V DC . This is no problem for the sensors because it is within their power supply range, the same does not apply to the switch because it requires a fixed 12 V DC power source. To solve this problem a MC7812SCT 12 V/1 A voltage regulator (figure 3.21a) was added to the input of the switch. To protect the circuit board of coming in contact with other terminals, a custom case was 3D printed (figure 3.21b).

A cable with 12 connectors runs from the trunk to the hood compartment along the car's right stringer and gives power to the sensors, to the switch and caries the signal to activate the relay in the power distribution board. The schematics in figures 3.23 and 3.22 represents the implemented circuit.



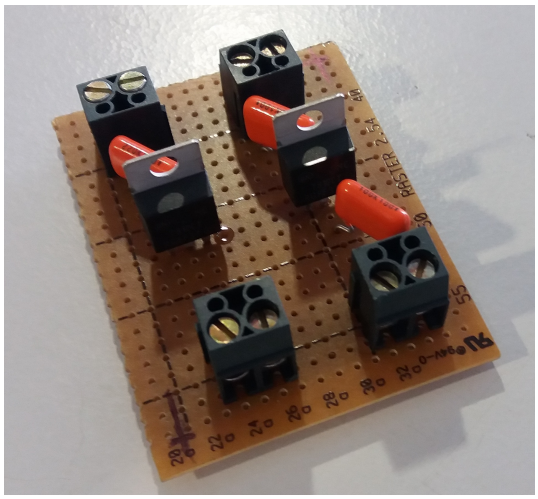
(a) Power distribution board inside view.

(b) Power distribution board outside view.



(c) Power distribution board connectors.

Figure 3.20: Power distribution board.



(a) Two MC7812SCT 12 V 1 A voltage regulator.

(b) Voltage regulator circuit's case.

Figure 3.21: Switch voltage regulator and case.

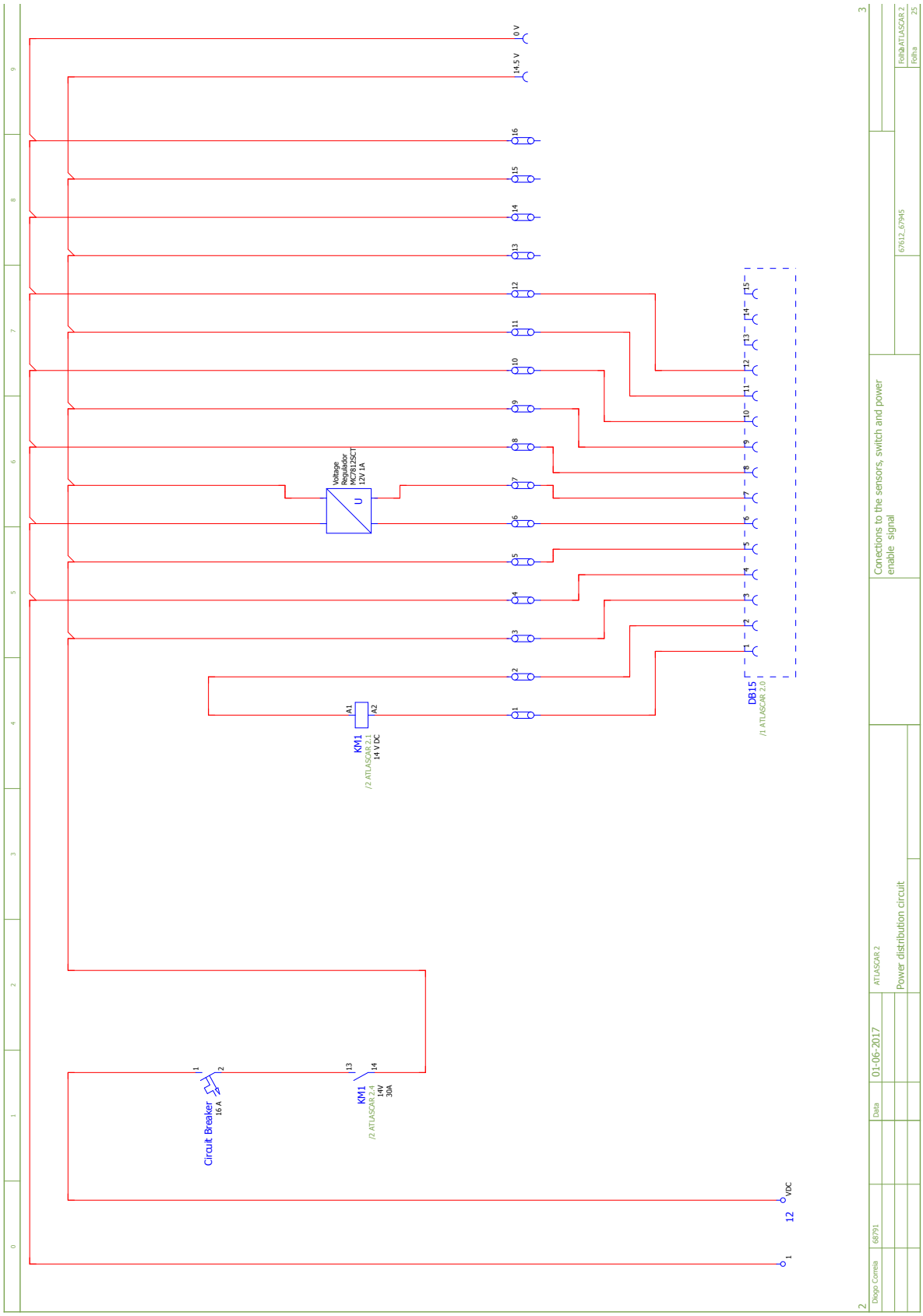


Figure 3.22: Power distribution board.

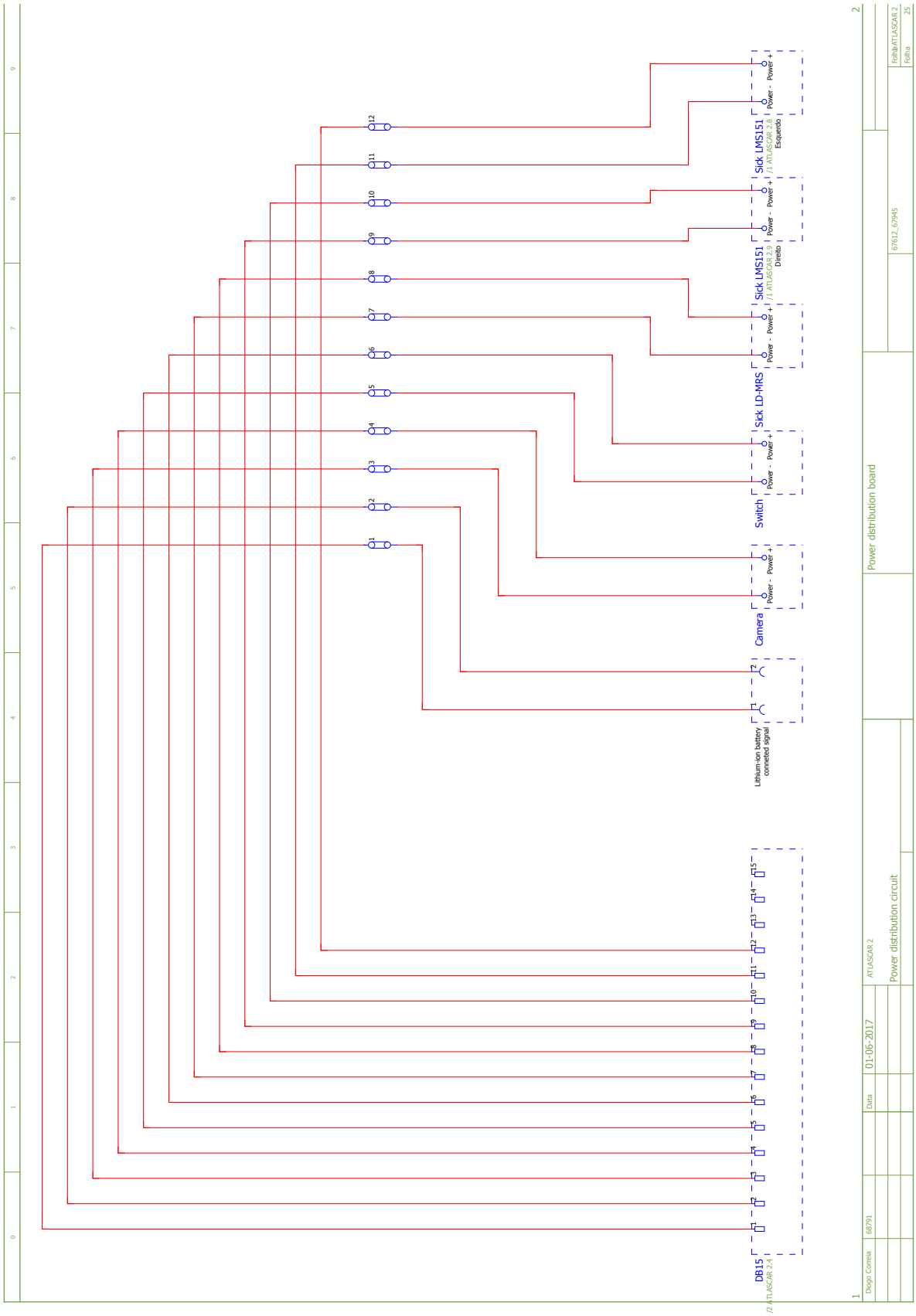


Figure 3.23: Power connections.

3.2.2 Communication infrastructure

Since all devices have an Ethernet interface, the communication infrastructure was very easy to setup. All the devices are connected to the switch (figure 3.24) that distributes the TPC-IP messages to the respective devices. An Ethernet cable Cat.6 runs alongside with the power cable from the hood to the trunk connecting the switch to the processing unit. Figures 3.24 and 3.25 represent the communication infrastructure.

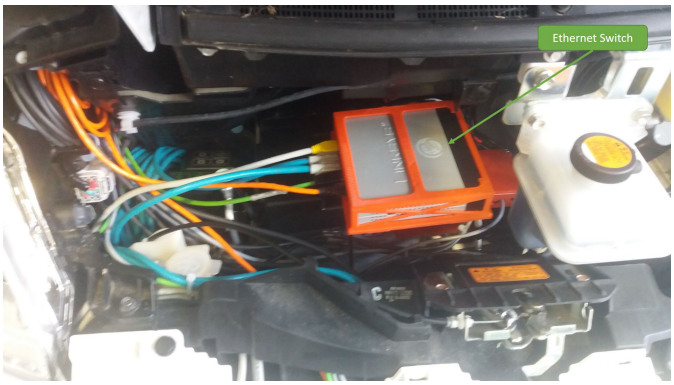


Figure 3.24: Switch connections.

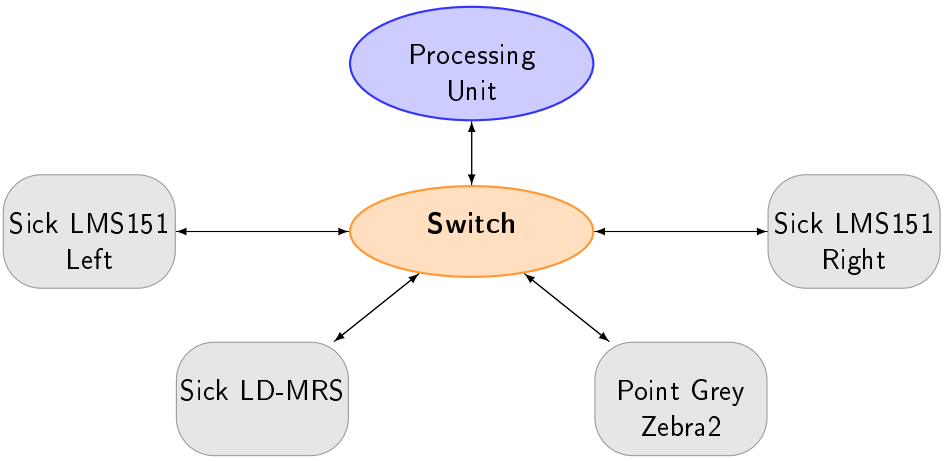


Figure 3.25: Communications diagram.

3.2.3 Wiring Installation

The installation of the power and the communication cables required some effort since the cables need to go from the sensors to the power distribution board and processing unit with the minimal modifications on the vehicle and keeping a good aesthetic look, so finding a place to run the cables through was not a trivial task.

The first alternative was to find a hole on the vehicle's chassis that gave access to the vehicle's interior from the hood compartment, from there the cables would pass under the carpet to the trunk, a solution similar to the one adopted in ATLASCAR 1. To find

the mentioned hole, part of the dashboard had to be removed (figure 3.26) and even then the only hole found was covered by refrigeration ducts. Accessing it made this solution impracticable. An alternative could be to drill another more accessible hole, but this was a last resource solution once it could damage the vehicle's integrity. After a few more attempts another alternative was found: there is space for the cables between the vehicle's chassis and the vehicle body along its stringers. From there the cable runs above the rear wheels, protected by plastic covers, and enters the engine compartment directly in the trunk. The other ends of the cables run above the front wheel and enter in the hood compartment by a space between the front bumper and the chassis.

(a) (b)

Figure 3.26: Attempts to find passage ways to run the cables. (Vehicles interior without parts of the dashboard).
(a) Vehicle's interior without the glove compartment.
(b) Car radio compartment.

On the other hand, the cables with power and signal to the monitor were easily installed. The power cable goes directly from the UPS to the monitor through a space between the vehicle's chassis and the plastic covers of the rear door opening, enters under the front passenger seat and goes up to the monitor. The VGA cable goes from the processing unit to the monitor alongside the power cable (sea figure 3.27).

Figure 3.27: Space between the chassis and the plastic covers for the read door opening to install the power and signal cables for the monitor.

Chapter 4

Sensor calibration and free space detection

This chapter describes the software developed to acquire and merge data from the LIDAR sensors and compute the free space around the vehicle, as well as the work done to add a new sensor to the calibration package developed by Silva [2].

As a tool to test the reliability of the developed hardware setup on ATLASCAR 2, and to make way for new projects and thesis related with AD and ADAS, some software was developed. Based on a ROS architecture, the developed package contains nodes that publish the frames for each sensor, subscribe the data from the LIDARs and computes the space available for navigation around the vehicle. Making use of `roslaunch` and launch files, the package allows for the user to acquire, visualize and store data from all the sensors as well as acquire data and, in real time, process it and visualize the space available for navigation around the vehicle using `Rviz`.

Apart from this package, this section also describes how the Velodyne Puck VLP_16 was added to the calibration package.

The next sections describe how the sensor frames are obtained and published, how the calibration target is detected in the Velodyne data, how the data is visualized and merged, and how the free space is represented.

4.1 Package architecture

The developed package has two main functions: first of, to acquire, visualize and optionally store the data, and secondly to compute the navigable space around the car. For each functionality there is a launch file that starts the required nodes which can be parametrised to enable or disable the data recording option. This option is enabled by default in the "get_data.launch" file and disabled by default in the "free_space.launch" file. The diagram in figure 4.1 shows the package launched nodes for each option. The orange and the blue rectangles represent launch files and the gray ellipses represent nodes.

Each launched driver connects to its respective device and starts publishing its data to the corresponding topic. The `device_frame` node reads the extrinsic calibration files and publishes a frame for each device. If the `get_data` file was launched, `Rviz` subscribes the raw data and displays it in the respective coordinate system. Otherwise, if the launched file was the `free_space`, the `free_space` node subscribes the data from the LIDARs

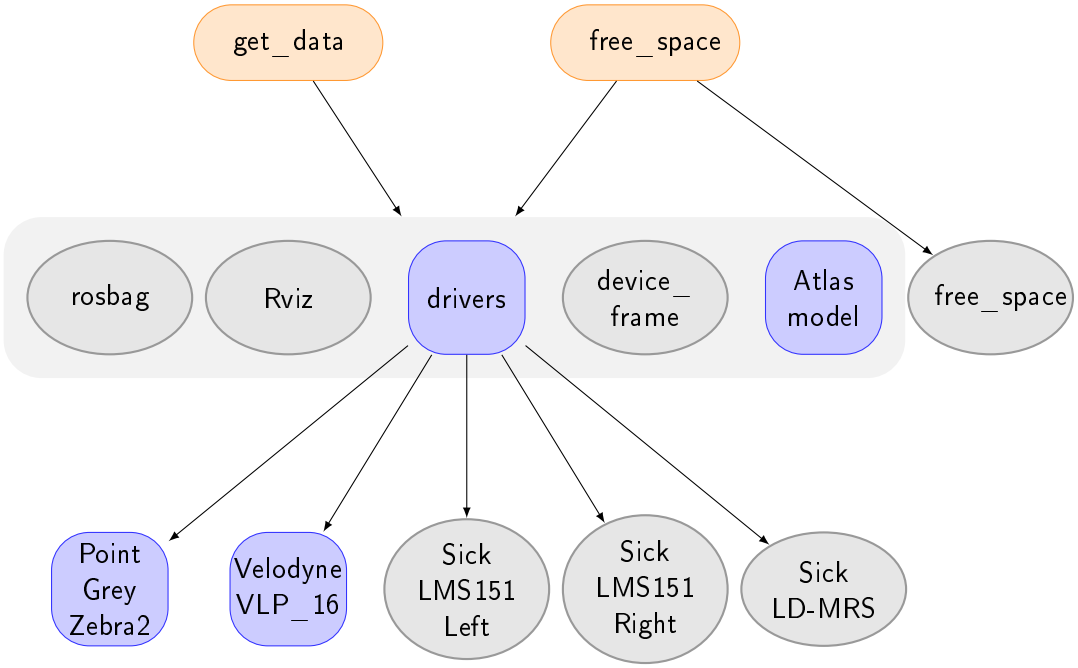


Figure 4.1: Launched nodes. The ellipses represent nodes and the parallelograms represent launch files.

processes it and then publishes it. `Rviz` will subscribe the processed data and displays it. If the recording option is activated, the `rosbag` node is launched and records the raw data from all sensors. Figure 4.2 illustrates the described architecture when the `free_space` file is launched. The next sections describe in more detail the most important nodes and launch files.

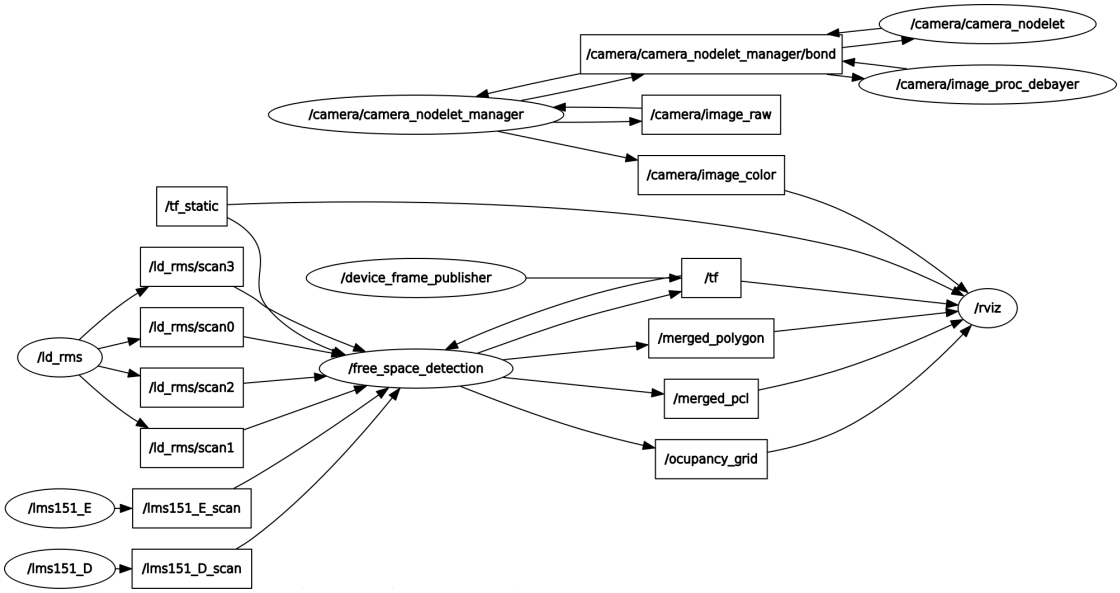


Figure 4.2: Active topics and nodes when the `free_space` launch file is launched.

4.2 Data aquisition

For all the sensors used there is a ROS package already developed that connects to the devices and starts data acquisition and publishes it to a specific topic. Using the launch file in the listing 4.1 allows to launch the drivers for all the sensors, specify the name of each launched node and, in the case of the Sick LMS151 driver, remap the name of the topic where the data is published so the two launched nodes do not publish the data of each LIDAR in the same topic.

```
<?xml version="1.0"?>
<!-- -->
<launch>
  <arg name="lms151_E" default="true"/>
  <arg name="lms151_D" default="true"/>
  <arg name="ld_rms" default="true"/>
  <arg name="zebra2" default="true"/>
  <arg name="VLP16" default="true"/>
  <!--Launch the drivers-->
  <node name="lms151_E" pkg="lms1xx" type="lms1xx" required="true" output
    ="screen" if="$(arg lms151_E)">
     <param name="host" value="192.168.0.134"/>
     <param name="frame_id" value="lms151_E"/>
     <remap from="scan" to="lms151_E_scan"/>
  </node>

  <node name="lms151_D" pkg="lms1xx" type="lms1xx" required="true" output
    ="screen" if="$(arg lms151_D)">
     <param name="host" value="192.168.0.231"/>
     <param name="frame_id" value="lms151_D"/>
     <remap from="scan" to="lms151_D_scan"/>
  </node>

  <node pkg="sick_ldmrs" type="sickldmrs.py" name="ld_rms" required="true
    " output="screen" if="$(arg ld_rms)">
     <param name="host" value="192.168.0.244"/>
     <param name="port" value="12002"/>
  </node>

  <include file="$(find pointgrey_camera_driver)/launch/camera.launch" if
    ="$(arg zebra2)"/>

  <include file="$(find velodyne_pointcloud)/launch/VLP16_points.launch"
    if="$(arg VLP16)"/>
</launch>
```

Listing 4.1: Launch file for device drivers - drivers.launch.

This launch file can also be parametrized to launch only a specific set of device drivers. By default all the drivers are launched. The following example would launch the drivers for all the sensor with the exception of the Velodyne Puck VLP16 scaner and the Point Grey Zebra2 camera:

```
roslaunch free_space_detection drivers.launch VLP16:=false
  zebra2:=false
```

4.3 Ball dection using Velodyne Puck VLP-16

In a latter stage of this work a Velodyne Puck VLP-16 LIDAR was available to test due to a cooperation with ISR. Since this device had never been used at LAR the calibration package could not calibrate this sensor. Therefore, in this work, we focused on adding it to the calibration packages so it can be easily used in future works.

To calibrate the sensor, first an algorithm must be developed to detect the calibration target in the point cloud data. Since the sensor returns a 3D point cloud, two approaches were analysed to detect the ball. One is to use a RANSAC algorithm from PCL similar to the one used to detect the ball in the Kinectic and the Swiss Ranger sensors. The other is to separate the point cloud data into multiple planar scans and apply the same algorithm used in Sick LMS-151 LIDARs to detect the sphere. Its centroid is calculated as the mean of all spheres detected in the planar scans.

The RANSAC method was pretty straightforward to implement, however it does not present reliable results since the ball is not detected in most of the cases (figures 4.3 and 4.4), most likely due to the reduced number of points available to fit the sphere model.

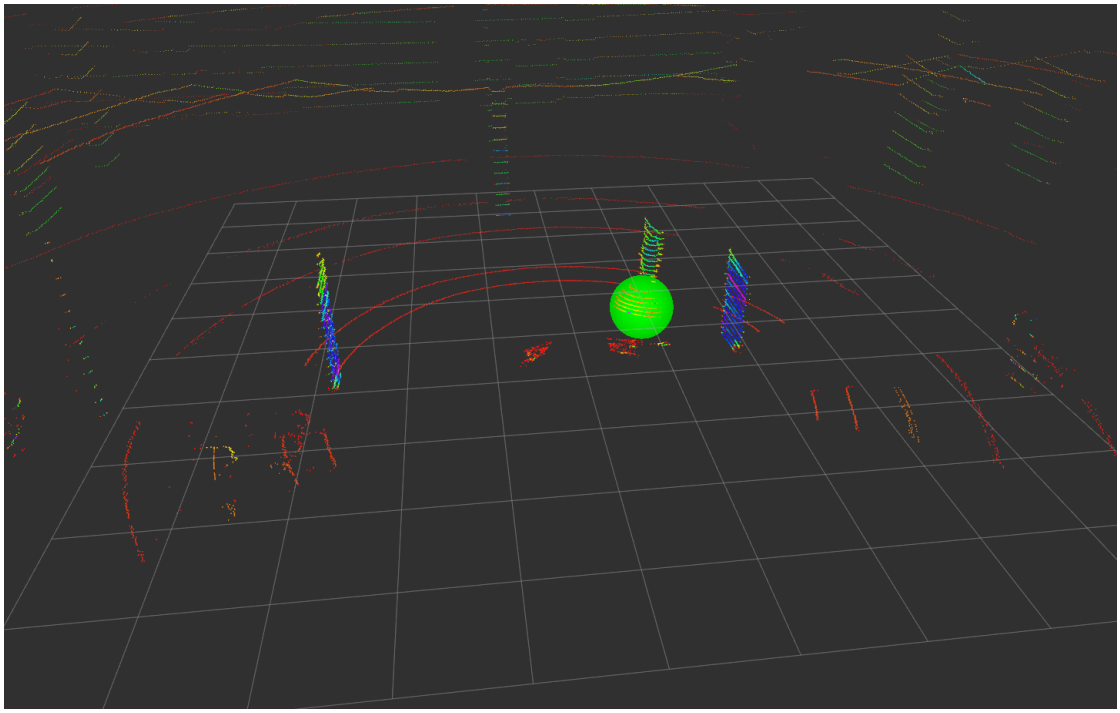


Figure 4.3: Ball detected by the RANSAC algorithm.

On the other hand, the implementation of the second approach was a hard task, but presented much better results. Since the device's driver publishes the data as a point cloud message, it has to, firstly, be separated into 16 scan planes corresponding to the 16 laser beans spread along the device's vertical aperture angle. This separation is only possible because the point cloud points have an extra parameter that indicates the number of the scan plane it belongs to, called ring number. The ring number varies from 0 to 15 were 0 correspond to the lowest laser beam with a -15° inclination and 15 correspond to the higher laser beam with a +15° inclination.

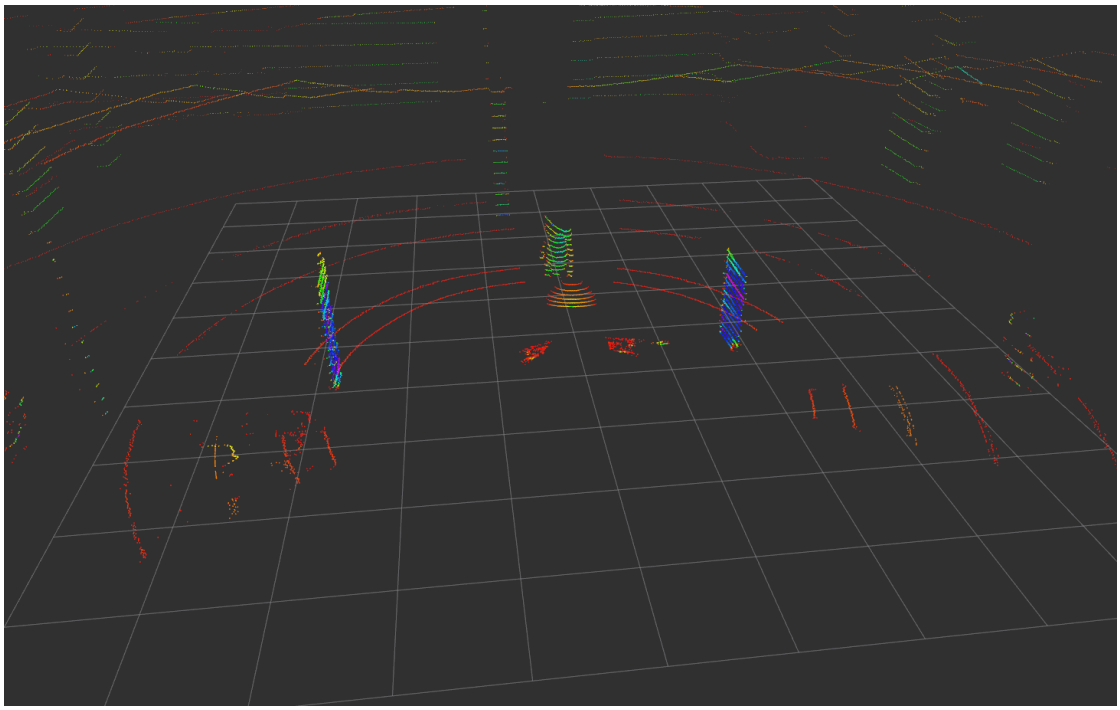


Figure 4.4: Ball not detected by the RANSAC algorithm.

Now, the same processing as the one applied to the planar LIDARs can be used. However, this processing requires some heavy computational algorithms like clustering, resulting in this process being too slow due to the high amount of data to process. To reduce the processed data, some assumptions were made. Firstly, the 8 scan planes with positive angle were discarded meaning the sensor has to be higher than the ball. Next, points with a negative x coordinate are also removed so the ball can only be in the front of the sensor. Finally, points too close to the sensor (less then 20 cm) are also removed resulting in much less data to process.

Next, the algorithm used to find the circle in the planar scans is applied to each remaining scans and the circle centers are computed and filtered to remove outliers. As described in chapter 2, this method required prior information about the ball's equator position relative to the scan plane to calculate the ball's center. In this case, there are some scan planes above the balls equator and others below. To solve this ambiguity, firstly the largest radius is found. Next the scans above the one with the largest radius are considered to be above the ball's equator an the others are considered bellow. Each computed centroid is then saved in a point cloud. After all the scans are processed, the algorithm computes two different means using the ball centers, one considering the largest radius as being above the ball's equator and other considering it is bellow the ball's equator. The mean that results in the smallest standard deviation is return as the ball's center. Figure 4.7 represents this algorithm and figures 4.5 and 4.6 show its results.

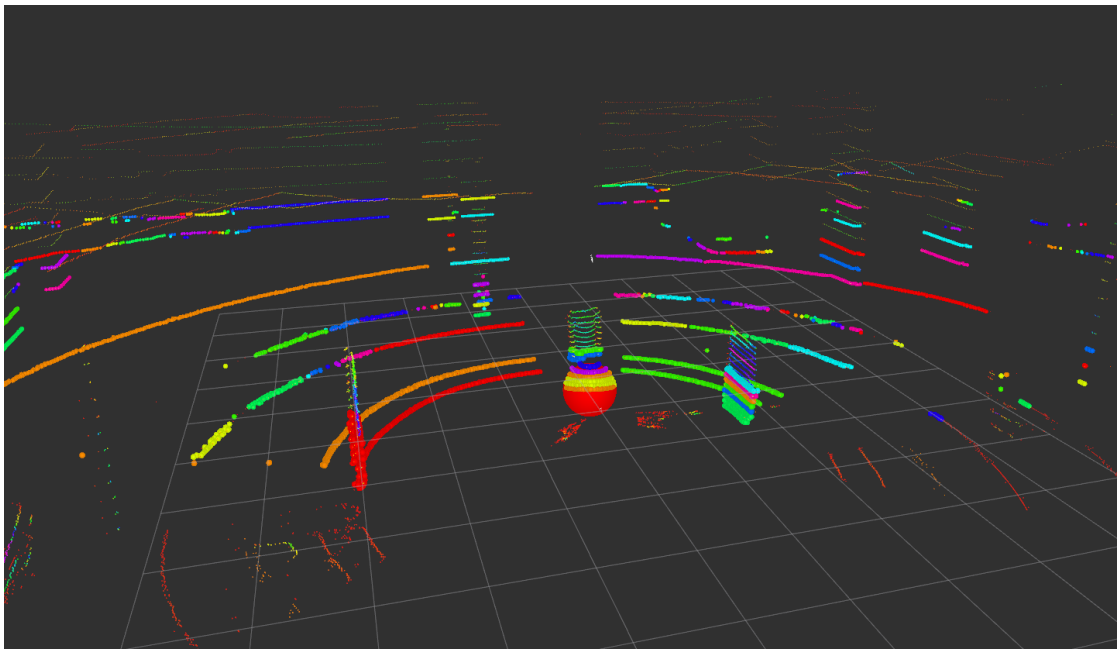


Figure 4.5: Ball close to the sensor detected by the planar scans algorithm.

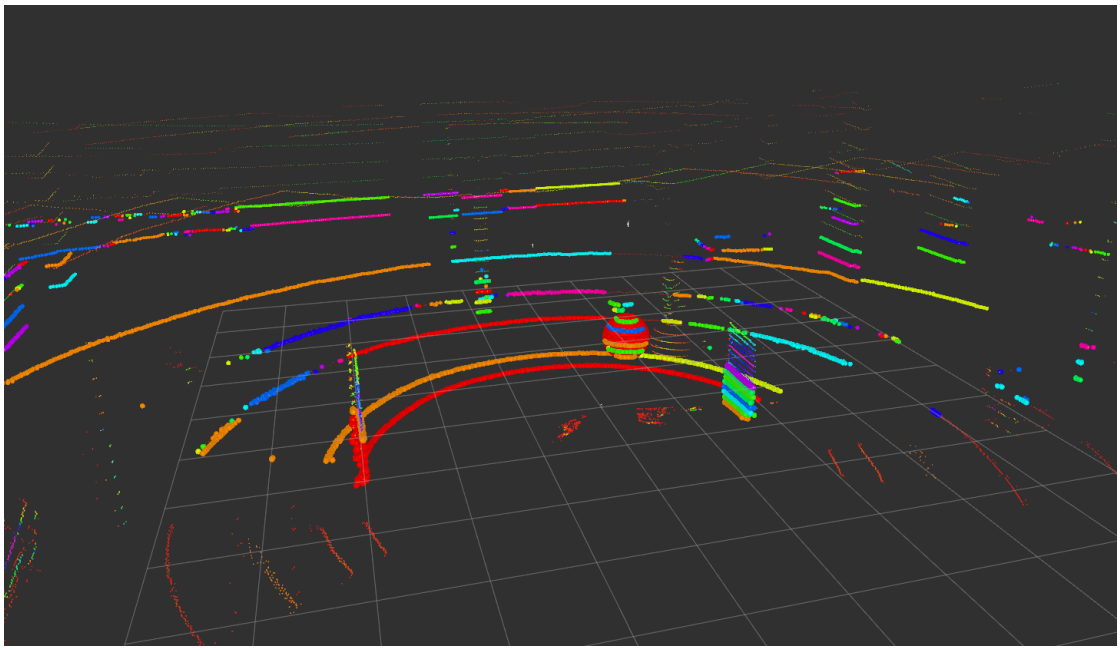


Figure 4.6: Ball far from the sensor detected by the planar scans algorithm.

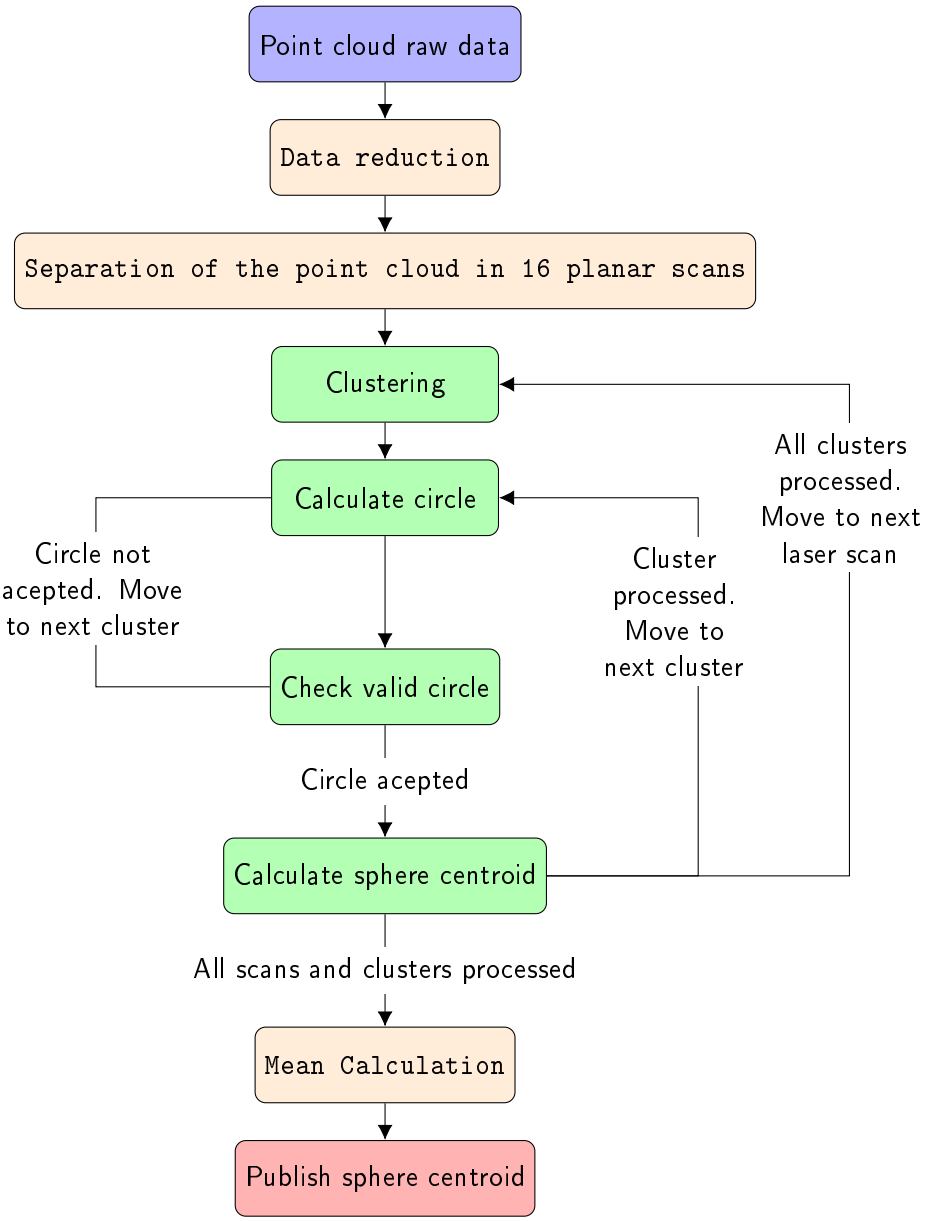


Figure 4.7: Ball detection algorithm using the separation of the point cloud in multiple planar scans.

4.4 Sensor calibration

4.4.1 Sick LIDARs calibration

The calibration package described in chapter 2 makes it possible to calibrate the two Sick LMS151 and the Sick LD-MRS LIDARs.

The calibration of the LIDARs was relatively easy using the automatic option of the package. Figure 4.8 shows the points used to calibrate the scanners. Figure 4.9a represents the sensors positions; each coordinate system represents one LIDAR, while the actual sensor position is represented in the CAD model. By analysing this image it

is possible to see that the coordinate systems of the sensors is coincident with the sensor actual position.

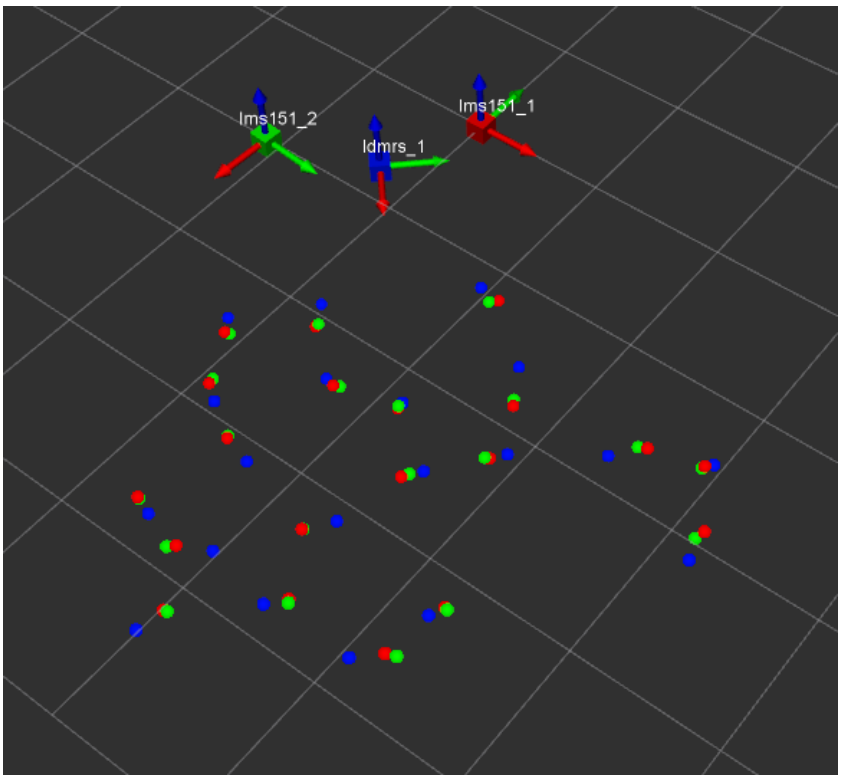
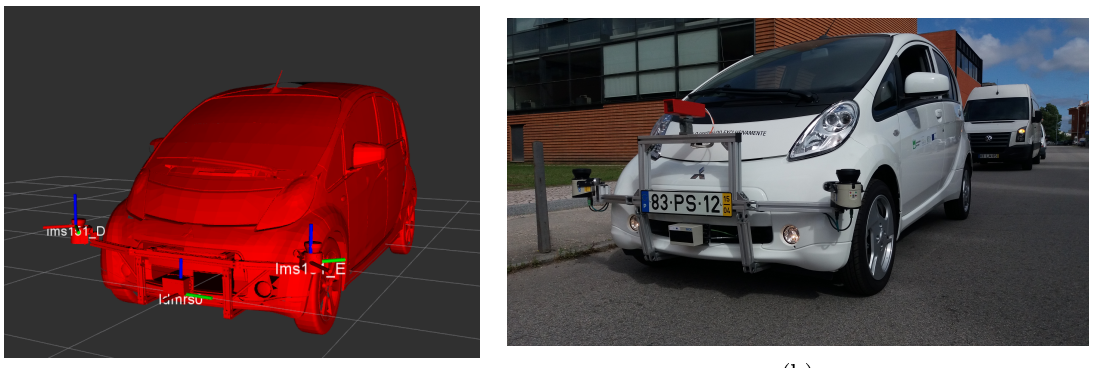


Figure 4.8: Points used in the LIDAR calibration.



(a) (b)

Figure 4.9: LIDAR calibration results.
(a) LIDARs positions estimation on ATLASCAR 2.
(b) LIDARs real positions on ATLASCAR 2.

4.4.2 Camera calibration

The calibration of the camera proved to be more difficult, due to two main reasons. Firstly, the ball's center on the image is hard to obtain because the ball is difficult to binarize, which causes the ball's center detection to be very unstable. Second, the camera's frame rate is much lower than the acquisition rate of the LIDARs. This is a problem when using the automatic calibration option because the calibration package compares the ball's displacement from all the sensors to evaluate if it was properly detected in all of them (although the sensors are not calibrated yet, the distance between two points must be the same in all sensors if the ball is properly detected) and because the frame rate of the camera is lower than the frame rate of the LIDARs, the ball is detected in one position in the LIDARs and when is detected in the camera it is already in a different position. To try solving this problems the camera was calibrated indoors with controlled illumination and using the user prompt calibration option. Nevertheless, the ball detection in the image was very unstable and the camera could not be properly calibrated.

4.4.3 Velodyne VLP_16 calibration

After the ball detection node was developed, the option to calibrate the Velodyne VLP_16 sensor was easy to add to the calibration package. Because this LIDAR was available for a short period of time the calibration was accomplished using previously recorded `bags` with data from the Sick LD-MRS and the Velodyne VLP_16 detecting the ball. The output of this calibration is represented in figure 4.10. The calibration results are shown in figure 4.11 where the image on the left represents the coordinate systems of the calibrated sensors, and figure on the right shows the real position of the sensors.

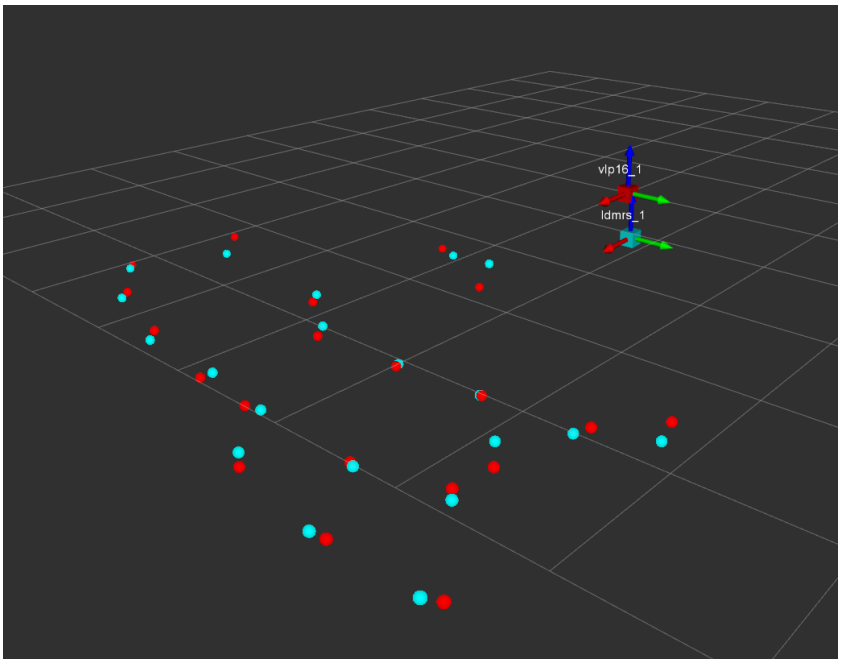
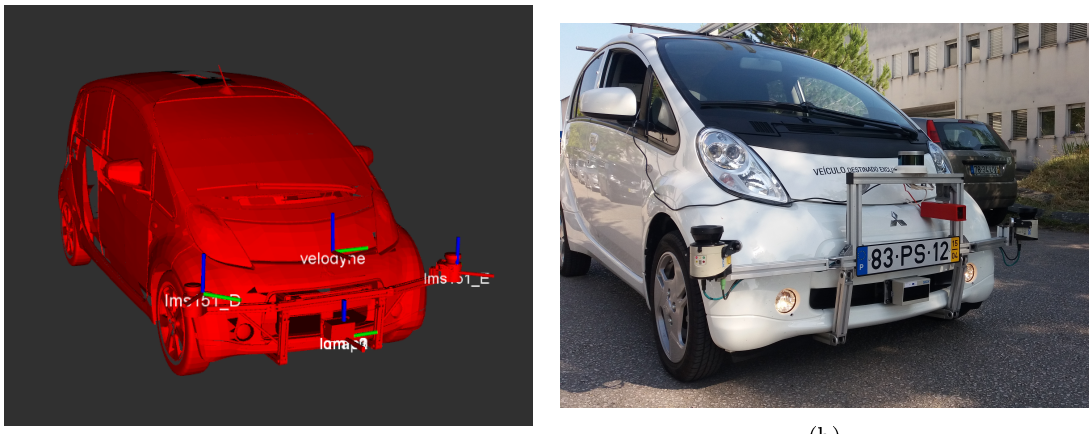


Figure 4.10: Points used to calibrate the Velodyne VLP_16.



(a) (b)

Figure 4.11: Velodyne calibration results.
(a) LIDARs positions estimation on ATLASCAR 2.
(b) LIDARs real positions on ATLASCAR 2.

4.5 LIDAR data fusion

This section describes how the extrinsic transformation between each sensor is published, how the data is represented and visualized, and how it is merged.

4.5.1 Sensor frames

The `device_frame_publisher_node` is the node responsible for publishing the extrinsic transformations from the reference frame — map — and the coordinate system of each sensor. Rviz will then subscribe to this ROS Transform (TF) messages to know the position of each sensor and represent its data accordantly.

The result of the calibration process using the calibration package described in chapter 2 are several text files with the extrinsic transformations of each calibrated sensor to the reference sensor. This means the calibration package outputs $n-1$ text files, were n is the number of calibrated sensors.

This node will read all text files from a specific folder designated "calibration_data", that must be nested on the package folder. Then creates a TF message for each transformation matrix read. These transformation represent the transformation from the reference sensor to the sensor whose name is the name of the text file where the matrix was read from. Any sensor can be used as a reference sensor so, when launching the node, an additional parameter must be passed indicating witch sensor is the reference sensor.

In order to have a data representation independent from the reference sensor, it was decided that the Sick LD-MRS was always coincident with the reference frame "map". That being said, an additional TF message corresponding to the transformation between the reference sensor and the world reference frame must be created. This transformation coincides with the inverse transformation of the reference sensor to the Sick LD-MRS coordinate system so it can be coincident with the world reference frame "map". Finally, all the frames are published and subscribed by `Rviz`.

4.5.2 LIDAR data merging algorithm

In order to have a representation of the navigable space around the ATLASCAR2 using data from multiple LIDAR sensors it is necessary to first merge the data.

The `free_space_detection_node` is responsible for subscribing the raw data from the LIDARs and process it.

The drivers for the LIDAR publish the laser data in a standard ROS message format named "LaserScan". This message contains a header with the message timestamp and the scan data in polar coordinates (the scan angle and the distance to the obstacle) in the laser coordinate system.

The first step of the process is to convert the LaserScan messages to a PCL format in the coordinate system of each device. Next, the point clouds are transformed to a same coordinate system and merged into it. The merged laser scans are selected through a parameter passed to the node in the launch file. In the case of the laser scans from the planar sensors, the point clouds are merged without any pre-processing; the same does not apply to the scans from the LD-MRS LIDAR. In the current configuration of this device on ATLASCAR 2, and because it has four scan planes with 0.8° between each scan plane, it is expectable that the two laser beams pointing downwards catch the road plane around 7.2 and 14.3 meters ahead (see figure 4.12), respectively, assuming the car's bottom plane is parallel to the road. This means that, if the laser lowest beam hits an obstacle further than this distance, most likely the car is accelerating or approaching a descent; this analysis will not be explored in this work. This means that the scans from the two lower planes are pre-filtered to remove points with distances, from the sensor, further than 6 meters for the lowest scan, and 13 meters for the second lowest scan. At this stage the point cloud contains the data from all the selected laser scans.

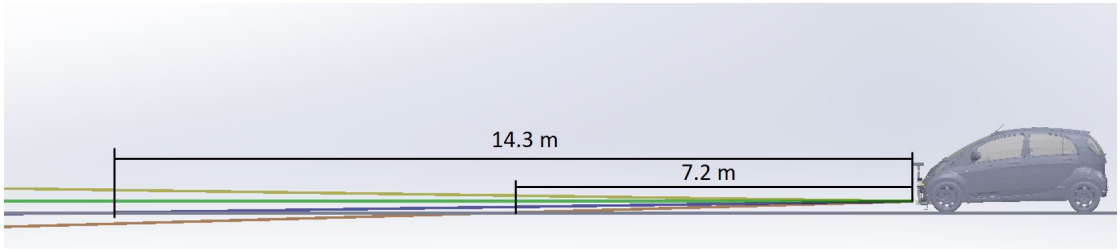


Figure 4.12: ATLCASCAR 2 side view with the scan planes from the Sick LD-MRS represented.

Since the representation of the free space will be in 2D, all the points will be projected to the XY plane, coincident with the road plane (z = 0). This means that there is an area in front of the vehicle were there are points from multiple LIDARs detecting the same obstacle. Although the calibration procedure returns accurate results, the extrinsic transformations between the sensors have some error, meaning that those points are not coincident; some are closer to the vehicle than others. Moreover, some objects may be detected by one scan and not detected by others, see figure 4.13. Since the merged data will be used to compute the free space around ATLASCAR 2, it is important to retain the objects closer to the vehicle, but obstacle behind those objects are less important since the space is no longer free in front of that object. Hereupon, removing the points with close azimuth angle results in a much less complex point cloud with all the relevant

information to compute the navigable space.

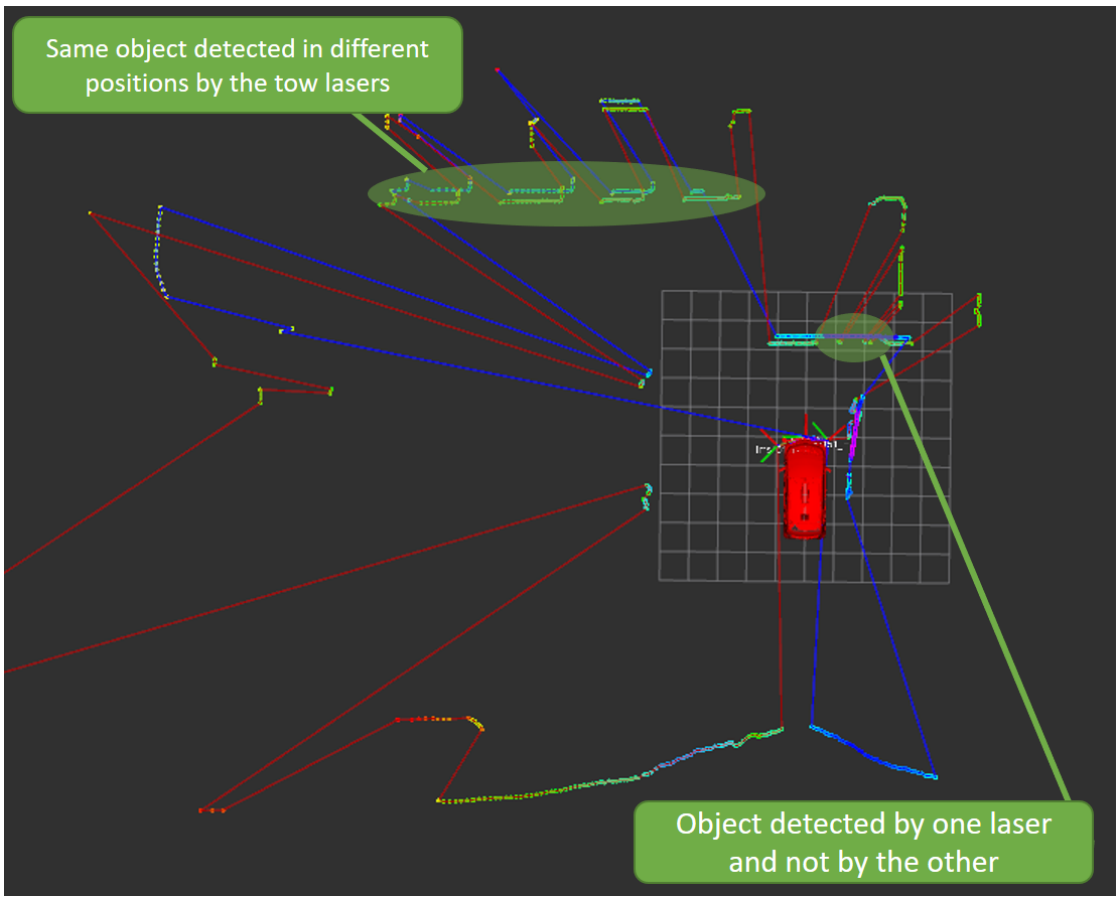


Figure 4.13: Top view of the laser data from the tow Sick LMS151.

To filter the mentioned points, firstly the point cloud is sorted by the point's azimuth angle; this operation will be also useful for the representation of the free space, as described further in this chapter. For the sorting process, the points are firstly converted to a polar coordinate system and then sorted. Next, a filter is applied on the sorted point cloud to eliminate redundant points. If a point is less then 0.5° apart from its neighbour then the algorithm will only keep the point closest to the vehicle. Finally, the points are reconverted to the original Cartesian coordinate system. This algorithm is represented in figure 4.14.

4.6 Free Space Representation

After the data is properly merged, the resulting point cloud contains a set of points that represent the obstacles around the vehicle which means that the space between the vehicle and those points is free to be navigated. Unfortunately, the vehicle has a typical Ackerman steering type which means it has a minimum turning radius. As mentioned in chapter 2, the vehicle's minimum turning radius is 4.5 meters which means the area where the vehicle can navigate, if there are not any obstacles, is the one shown in figure

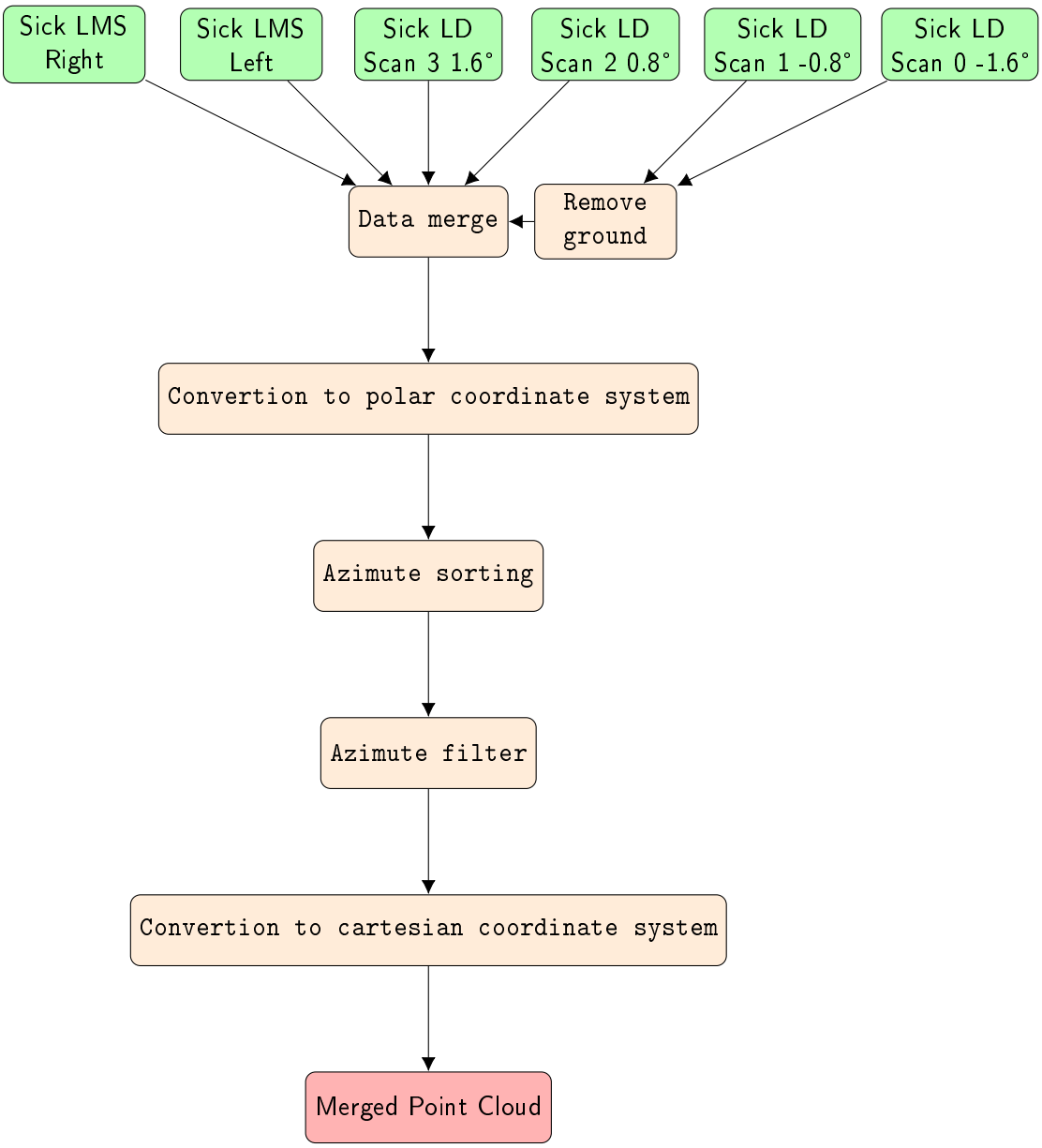


Figure 4.14: Data merging algorithm.

4.15. By intersecting this area with the one provided by the merged LIDARs data it is possible to determinate where the vehicle can actually navigate.

The free space obtained from the LIDAR data and vehicle's physical restrictions is represented with two different approaches. One is to connect the points from the merged point cloud by a line creating a polygon. The interior of that polygon represents the free space, sea figure 4.17a. To be able to create this representation, the points from the point cloud need to be sorted by their azimuth angle. Fortunately, the merging algorithm already returns the point cloud points sorted, so the implementation of this representation is made easier. This approach is fast to compute, however it does not provide an intuitive

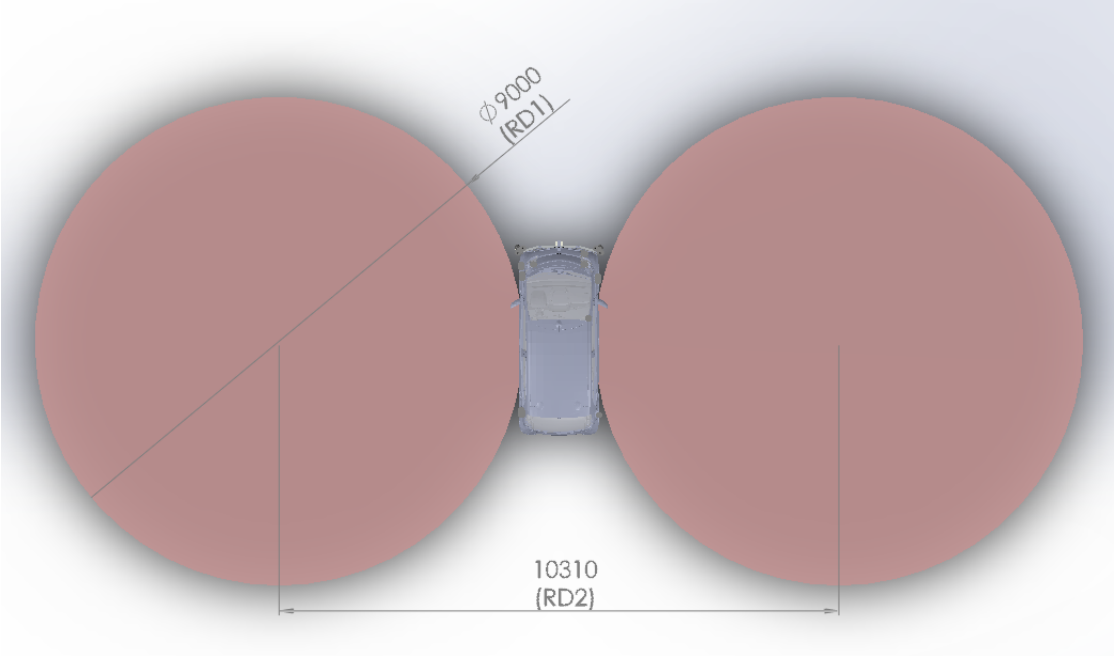


Figure 4.15: Unreachable area of ATLASCAR 2 due to steering limitations.

representation and path paling algorithms are more difficult to implement with this representation.

The second approach is to use occupation grids. To convert the point cloud data into an occupation grid, the first thing is to have the data points sorted, next the points need to be uniformly spaced so virtual points are added between each consecutive point along a line defined by those points so the final point cloud has all consecutive points with a space between them equal to the resolution of the grid. With this point cloud, the algorithm assigns each cell of the grid its value. If a point, from the point cloud, is inside a cell, that cell assumes the color black and the cells along the line between the center and the point assume the color green, see figure 4.16. This is done for every cell of the grid with the exception of the cells where the car can not go due to the steering limitation. These areas are marked in red if there is no information about the occupancy of that space and marked in yellow if that area is free of obstacles. The rest of the grid is represented in grey (unknown space). Figure 4.17b shows an example of this representation. Although this approach is slower to compute, it is much more intuitive and it is easier to apply path planing algorithms using this representation. Listing 4.2 shows the piece of code used to assign the values to the grid from the point cloud input.

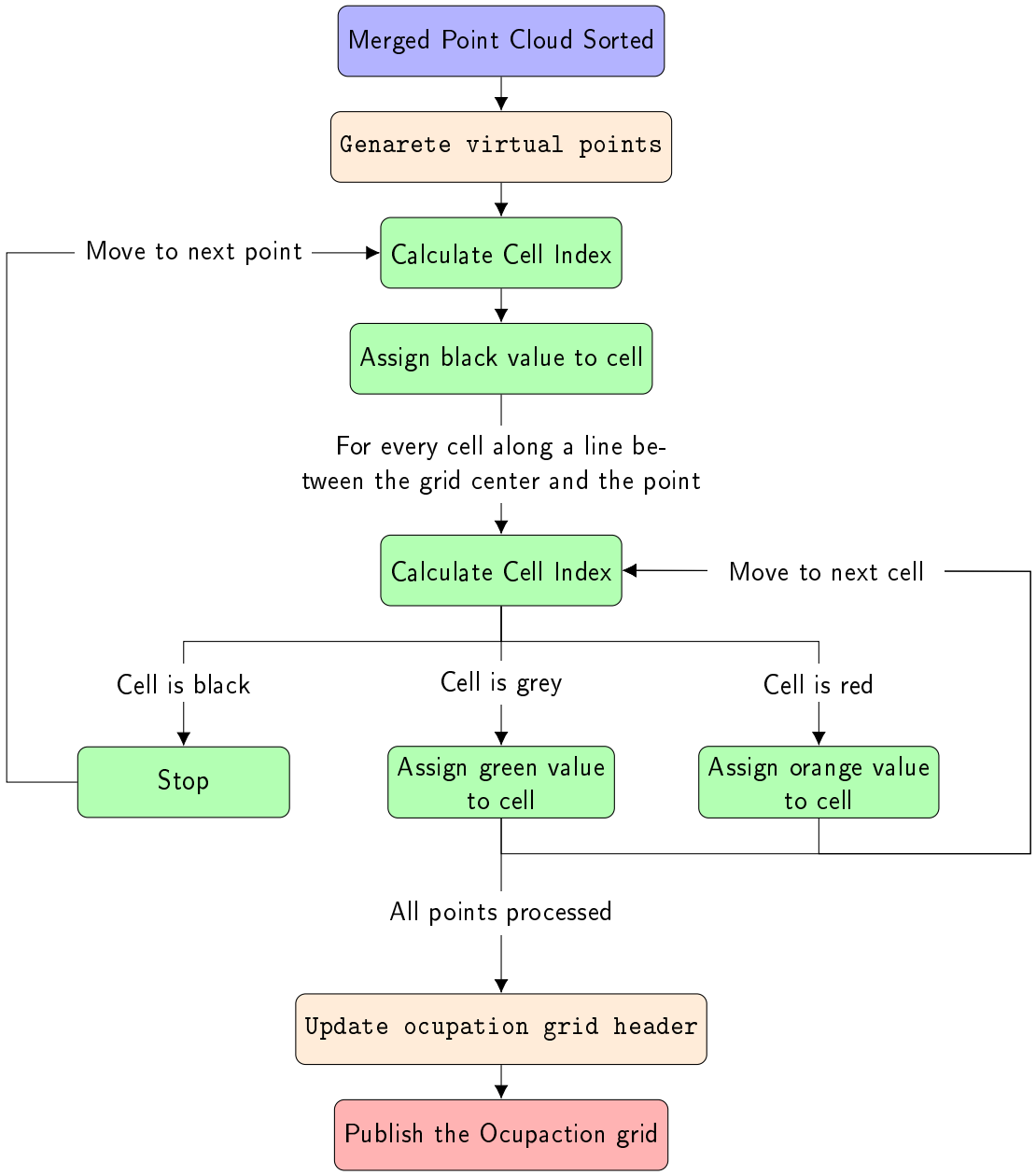
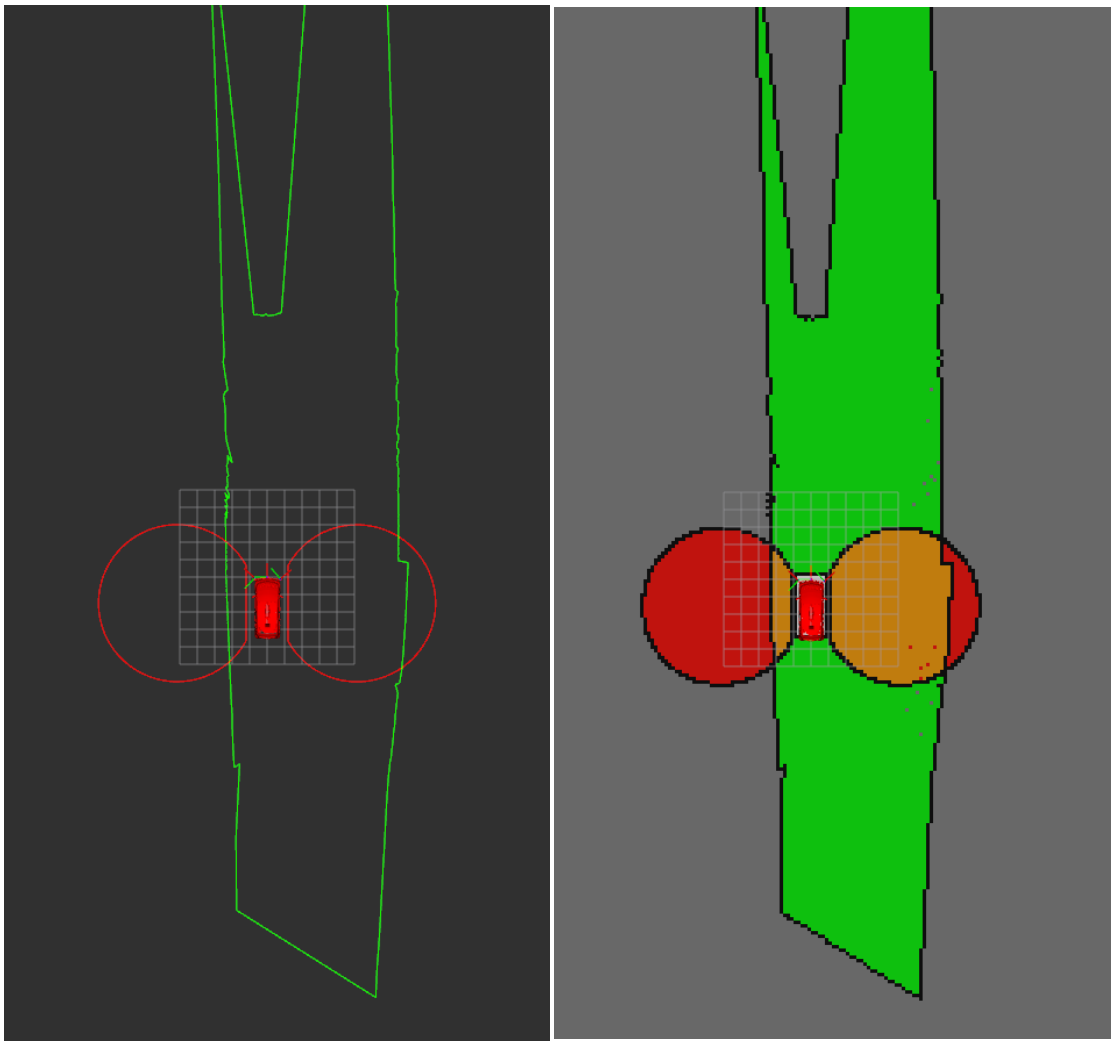


Figure 4.16: Ball detection algorithm using the separation of the point cloud in multiple planar scans.



(a) (b)

Figure 4.17: Free space represented using two different approaches.
(a) Free space represented by a polygon.
(b) Free space represented by an occupation grid.

```
1  void ocupGrid::populateMap(pclPtr inPcl, int color, double
    xOrigin, double yOrigin){
     for(int i = 0; i<inPcl->points.size(); i++){
3
       geometry_msgs::Point point;
5      point.x = inPcl->points[i].x; double x = point.x; x+=
    xOrigin;
       point.y = inPcl->points[i].y; double y = point.y; y+=
    yOrigin;
7      point.z = 0;

9      if(x<xMax && x>xMin && y>yMin && y<yMax){
         int xCell = (int) ((x-xMin)/cellResolution);
11       int yCell = (int) ((y-yMin)/cellResolution);

13       int idx = yCell*xCells + xCell;
         ocGrid[idx] = 100;
15     }

17     geometry_msgs::Point point_s = xyzTortp(point);

19     for(double k = cellResolution; k<point_s.x; k+=
    cellResolution){
         geometry_msgs::Point point;
21       point.x = k;
         point.y = point_s.y;
23       point.z = point_s.z;
         geometry_msgs::Point point_c = rtpToxyz(point);
25
         double x = point_c.x; x+=xOrigin;
27       double y = point_c.y; y+=yOrigin;

29       if(x<xMax && x>xMin && y>yMin && y<yMax){
           int xCell = (int) ((x-xMin)/cellResolution);
31         int yCell = (int) ((y-yMin)/cellResolution);

33         int idx = yCell*xCells + xCell;
           if(ocGrid[idx]==UNKWON){
35          ocGrid[idx] = color;
           }else if(ocGrid[idx] == RED && color != RED){
37           ocGrid[idx] = YELLOW;
           }
39       }
       }
41   }
  }
```

Listing 4.2: Function used to convert the point cloud data into an occupation grid.

Chapter 5

Results

This chapter presents the results of the work developed in this thesis.

The mechanical fixations for the sensors were evaluated to ensure that the scanners are kept stable during vehicle motion in any kind of paved roads. Next the quality of the calibration results are also analysed. Finally, the LIDAR data merging algorithm and the free space detection node are also evaluated.

5.1 Sensor fixtures and calibration

The sensor mounting setup in ATLASCAR 2 must fullfill two main requirements:

- Ensure a clear field-of-view along the LIDARs aperture angles (vertical and horizontal aperture angles in the case of the Sick LD-MRS and horizontal aperture angle only in the case of the two Sick LMS151);
- Provide stable and reliable mounting system even when the vehicle is in motion.

The first requirement was easy to evaluate. After the sensors are installed, visualizing its data using Rviz shows that none of the scan points represent any part of the car, meaning the laser beams from the scanners do not hit any part of the car, figure 5.1.

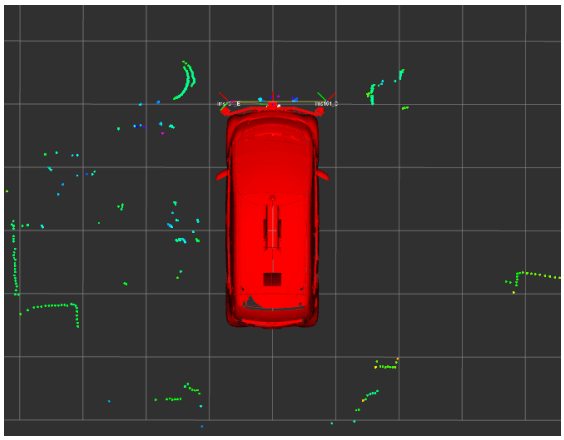
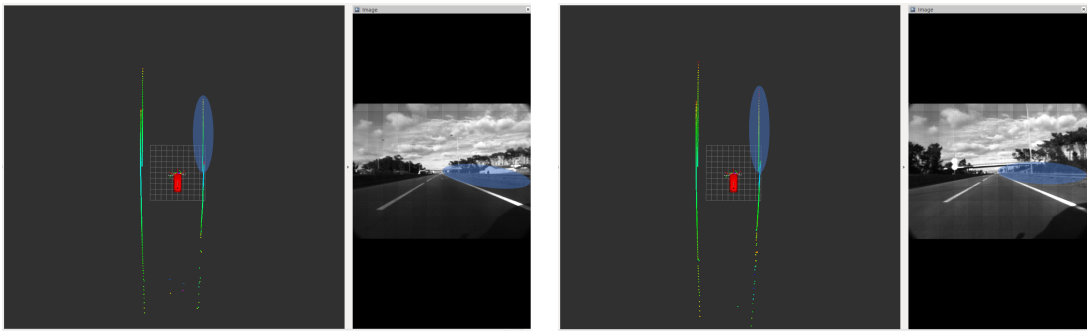


Figure 5.1: LIDAR data and ATLASCAR 2 model.

The second requirement is harder to quantify, however there are some indicators that point to the quality of the solution. Figure 5.2 represents two screenshots of the vehicle moving in a highway approximately at 100 km/h. The highlighted points represent the road rails caught by the planar scanners. These road rails are found by the scanners even at distances far from the car and they are continuously detected during the vehicle's route. This means the lasers are stable otherwise slight variations on the laser tilt angle would result in the road rails being detected intermittently witch is not verified.



(a) Screenshot 1. (b) Screenshot 2 - a few meters ahead.

Figure 5.2: Screenshots of data acquired from the sensors with the vehicle in motion in two different positions.

Regarding the calibration of the Sick LIDAR, figure 5.3 represents the calibrated data from the scanners in the same coordinate system. As is possible to see, when the same object is detected by all the LIDARs as is the case of the highlighted points the measurements are similar in all the three LIDARs.

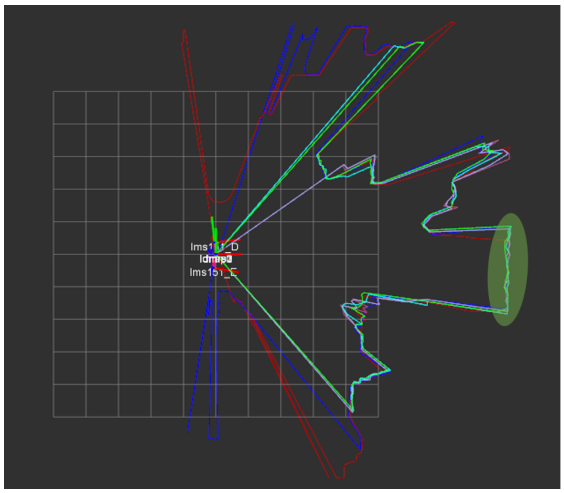


Figure 5.3: LIDARs calibrated data.

5.2 Velodyne Puck VLP-16 calibration

After the development of the ball detection algorithm (represented in figure 4.7) to find a sphere in the VLP_16 data, this sensor was successfully added to the calibration package. Figure 5.4 represents the data from the Sick LD-MSR (represented in white dots) and from the Velodyne VLP_16 (represented in color dots) in the same coordinate system. As is possible to see, the positions of the detected objects coincide, with a small deviation, in the data from booth sensors. Since the Sick LD-MSR has a maximum error of about 10 cm the calibration output presented satisfying results.

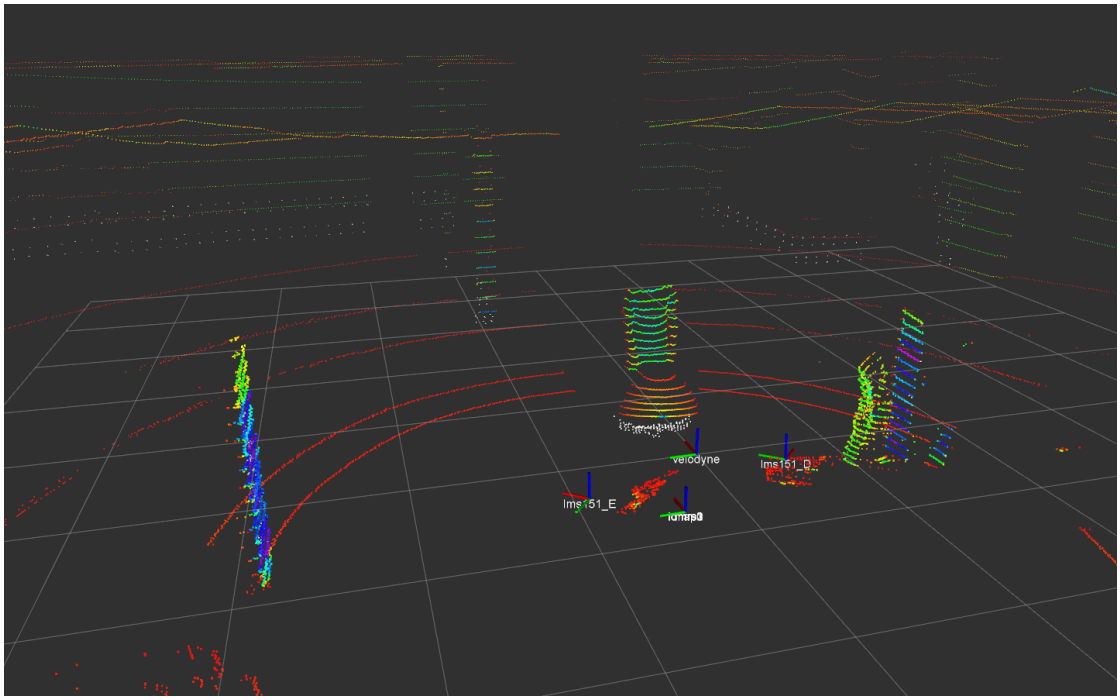
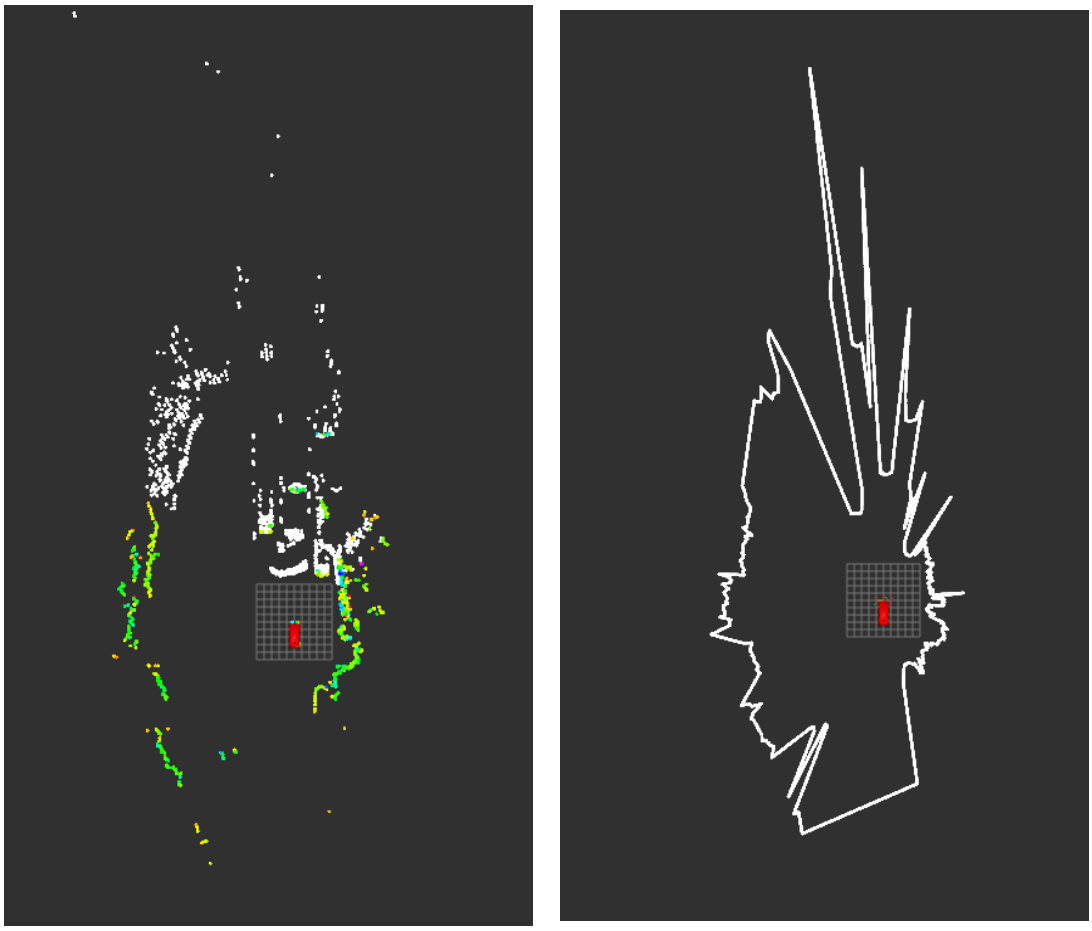


Figure 5.4: VLP_16 and Sick LD-MRS calibrated data.

5.3 Sensor data fusion

The data fusion algorithm merges the data from the LIDAR sensors and treats it providing a point cloud suitable to be used to represent, in 2D, the free space around ATLASCAR2. The main function of there data merging algorithm (figure 4.14) is to remove ground detections from the Sick LD-MRS scan planes, remove points detecting objects behind other object or points from the same object but represented in different positions due to calibration errors and finally sort the points by their azimuth angle. As figures 5.5 and 5.6 show, the algorithm works in two different environments, high way and urban.

As is possible to see, in both cases, the algorithm removes all the ground points from de Sick LD-MRS, and filters the data resulting in a much more clean point cloud, nevertheless maintains all relevant information to compute the free space.



(a) (b)

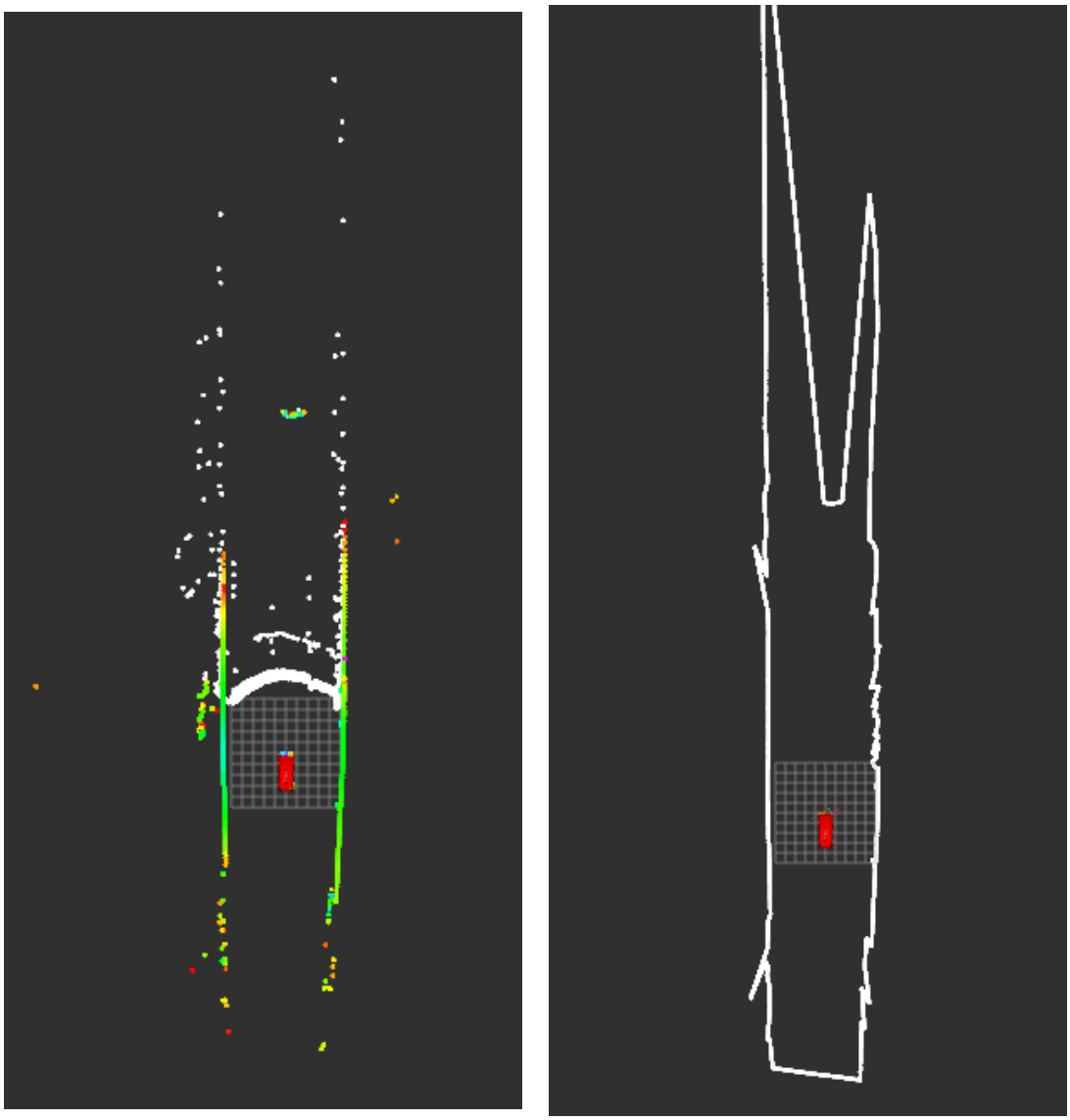
Figure 5.5: Data gathered in a high way, represented before (left image) and after (right image) being merged.
(a) All scan points represented in the same coordinate system before they are merged.
(b) Point cloud resultant from merging the scans showed in figure (a).

5.4 Free space detection

The algorithm developed to calculate the free space around the vehicle using occupation grids present good results even when the objects in front of the vehicle are lower then the two Sick LMS151 scan planes.

Figure 5.7 represents a scene with multiple obstacles like a paper box and a ball, figure 5.8 shows the computed free space using the tow representations developed in this thesis. As is possible to see the objects are well represented even in the case of the box that is lower than the two Sick LMS scan planes but is identified by the Sick LD-MRS.

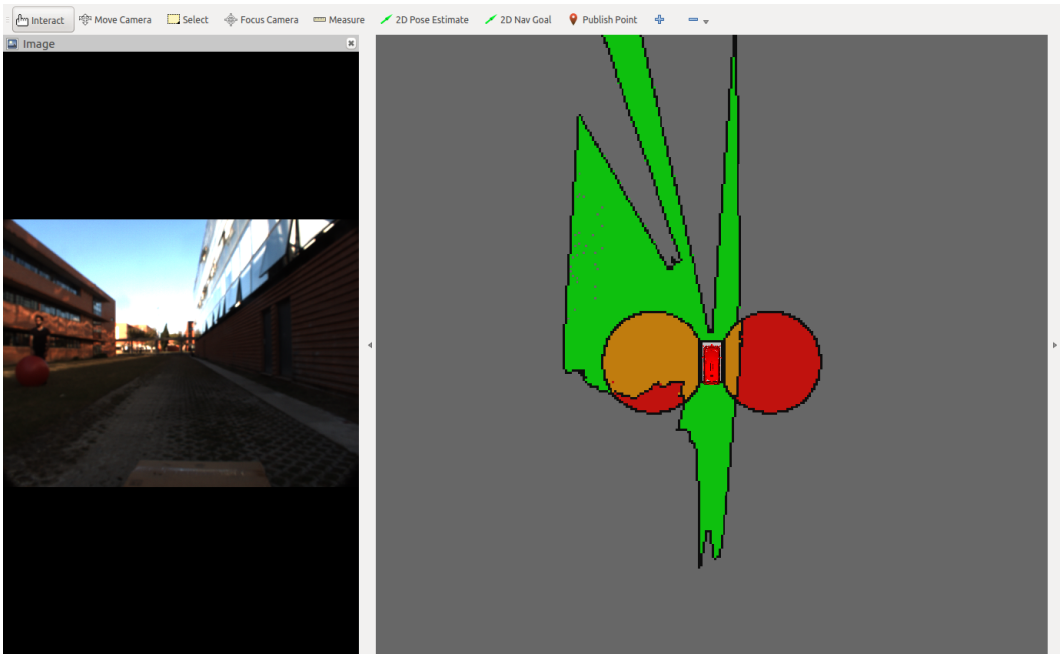
Figure 5.9 represents the same scene with some changes in the objects positions and figure 5.10 shows the representaions of the free space. In this images we can see that all the objects are defined in the representations including an open door at the building on the left and the grey obstacle on the right also present in the previous scenario.



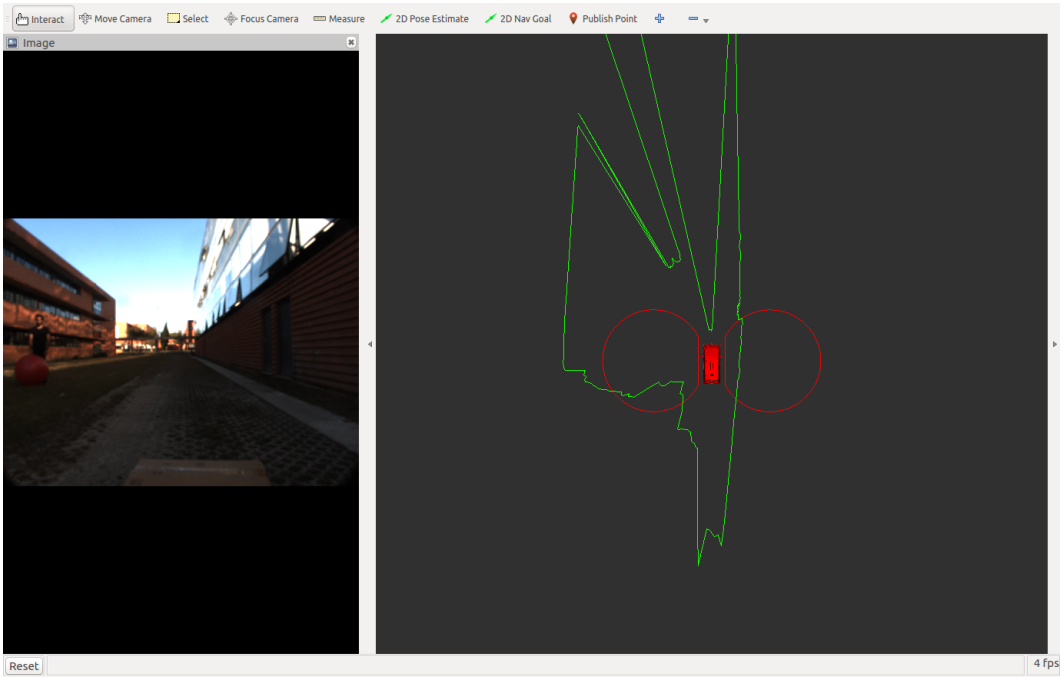
(a) (b)

Figure 5.6: Data gathered in a urban environment, represented before (left image) and after (right image) being merged.
(a) All scan points represented in the same coordinate system before they are merged.
(b) Point cloud resultant from merging the scans showed in figure (a).

Figure 5.7: Obstacles arround ATLASCAR 2.



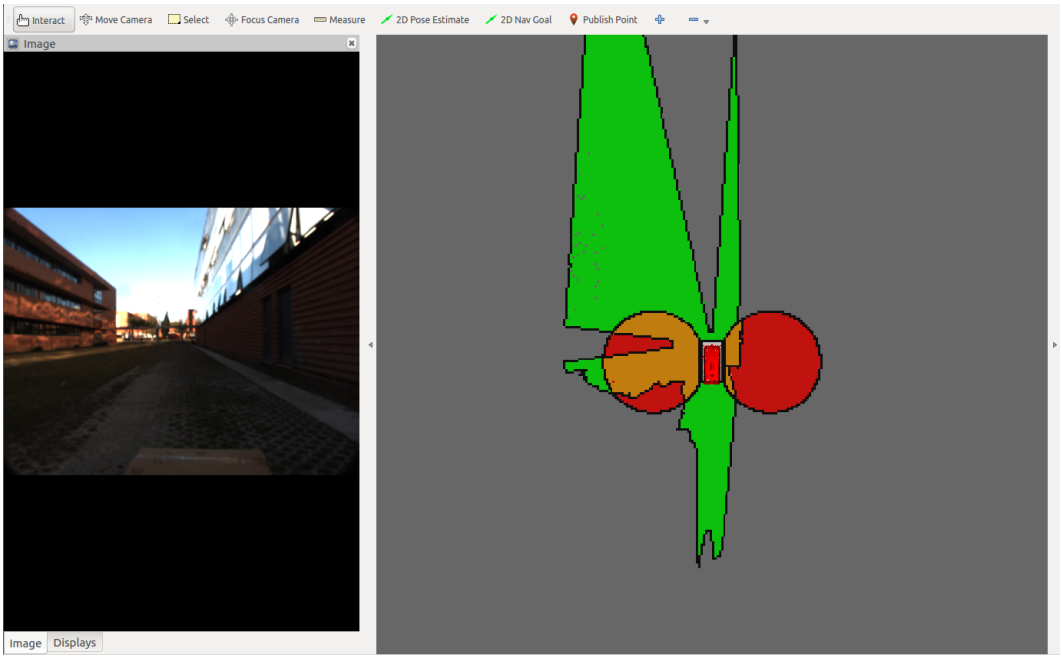
(a)



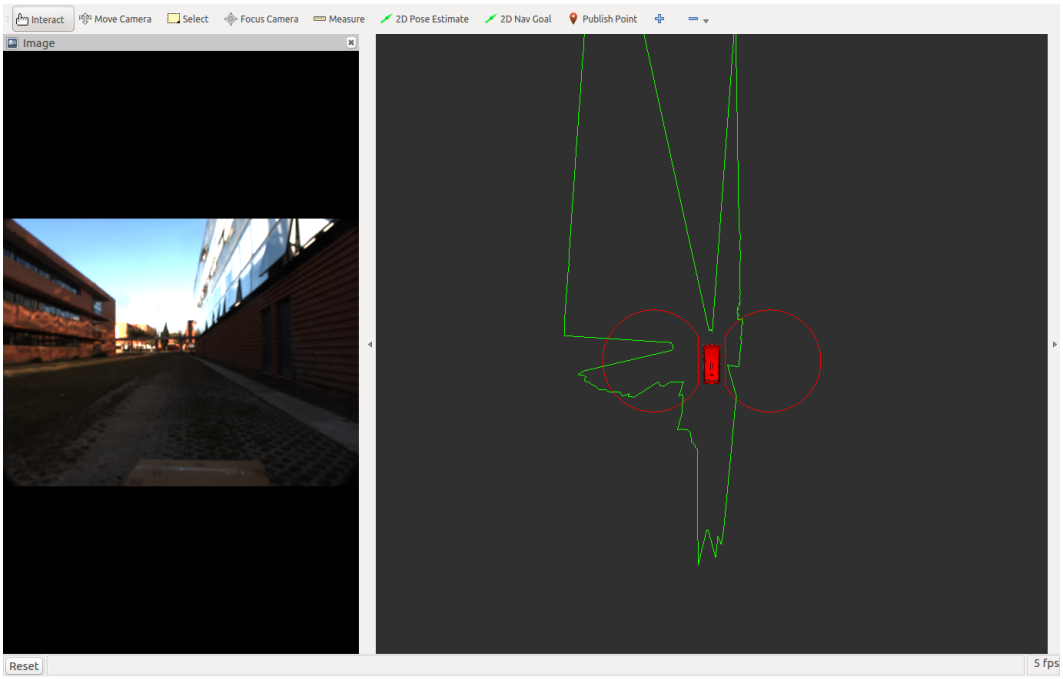
(b)

Figure 5.8: Free space representation with test objects in the environment.
(a) Free space represented using an occupation grid.
(b) Free space represented using a polygon.

Figure 5.9: Obstacles arround ATLASCAR 2.



(a)



(b)

Figure 5.10: Free space representation with test objects in the environment.
(a) Free space represented using an occupation grid.
(b) Free space represented using a polygon.

Chapter 6

Conclusions and Future work

This chapter is a summary of the work developed in the matter of the hardware setup and regarding the free space detection and the calibration packages. Based on this conclusions a few proposals for future works using ATLSACAR 2 are presented.

6.1 Conclusions

Research related to ADAS and AD is a priority for many car manufactures, research institutes and universities. Having a test and data acquisition platform it is essential to move the research forward. This work provides a prototype equipped with several LIDAR and vision-based sensors combined with the required hardware and software to acquire data in real road scenarios and test navigation and perception algorithms on a full scale prototype circulating in real uncontrolled environments.

The position of the two Sick LMS151 allows for a 360° field of view around the car and the vehicle does not cause any type of occlusion to the scanners. The Sick LD-MRS position allows for a clear horizontal field of view. However, its lower position causes the two lower scan planes to be useless for obstacle detection at distances larger than 6 and 12 meters respectively for the lowest and the second lowest scan planes. This issue can be solved by moving this laser to a higher position or change its tilt angle. However, having two scan planes points to the road may be very useful for other applications like estimate the road inclination or side-walks detection, therefore the laser was kept in the same position. The mounting system installed in the front bumper of ATLASCAR 2 was proven to be rigid enough and can be used to fix any sensor with many possibilities for its position and orientation and the rooftop bars provide even more options for sensor installation. The power for the sensors is being drawn from the car's low voltage circuit allowing the vehicle to acquire data in motion during a long period of time. The switchboard is able to power all the installed sensors and is prepared to power even more that may be added in the future. In order to protect the car's lead battery from discharging, the devices can only be powered on when the vehicle's traction batteries are connected (the vehicle's engine is on or the vehicle is charging).

The calibration of the installed LIDARs was accomplished with success and the resulting transformations matrices presented with satisfying quality the extrinsic relations between the sensors. However, the camera calibration still needs to be improved. The ball is hard to binarize and as result the centroid detection is very unstable. Combing it

with a low frame rate results in being very difficult to get a valid point in all the sensors at the same time compromising the calibration process. Regarding VLP_16, this sensor was successfully added to the calibration package. The ball detection algorithm is able to detect the ball wether if is close or far from the sensor, and does it with satisfying precision resulting in a reliable calibration.

Regarding the free space detection, two representations were implemented, each with its advantages and disadvantages. Representing the free space by a polygon has the advantage of being fast to process and compute, but it is difficult to use for other applications like path planing. On the other hand, the occupation grids proved a more suitable representation for further developments in path planning, but the algorithm is slower and computational heavier.

6.2 Future Work

The amount of work that can be done on this test platform is very large. However, some are more prioritary. Of these, is worth mentioning the development of a path planning algorithm to compute the best possible trajectory given the free space around the vehicle and an end goal. To develop a data acquisition and automatic labelling package in order to create a repository of data sets that can be used to test perception algorithms, is also an important extension of this work. The acess to the vehicle's information about speed, odometer, steering wheel position and throttle and brake pedals positions, available from the vehicle's On-Board Diagnostics (ODB-II) port, would also be useful to provide information for driver profiling and for navigation.

Regarding the hardware setup on ATLASCAR 2, some improvements can be made. Those may include the installation of a switchboard connected to the car's Lithium-ion battery to power the processing unit and other hardware that may be installed in the future, eliminating the need for the UPS. The installation of a GPS system could also be useful for navigation and path planing algorithms. A small touch display could also be installed in the front passenger compartment for data visualization or even for the driver to interact with an ADAS application like parking assistance.

In the calibration package, the ball detection on the camera's image needs to be improved. For example, using an object tracking library to track the ball and reduce the ball's center detection algorithm to a smaller region of interest instead of using the hole image. Being a smaller region, the ball would be easier to binarize which would result in faster processing and more stable centroid detection.

The free space detection package can also suffer some minor improvements, namely regarding the usage of the two lowest scan planes of the Sick LD-MRS. To know for sure at what distance the scans will hit the road it is necessary to know the vehicle inclination relative to the road. With this information it is possible to better evaluate if the scan is hitting an obstacle or the road improving the quality of the merged data.

Bibliography

[1] Marcelo Pereira et al. "Self calibration of multiple LIDARs and cameras on autonomous vehicles". In: *Robotics and Autonomous Systems* 83 (2016), pp. 326–337. ISSN: 09218890. DOI: 10.1016/j.robot.2016.05.010.

[2] David Silva. "Multisensor Calibration and Data Fusion Using LIDAR and Vision". MA thesis. Universidade de Aveiro, 2016.

[3] Marcelo Silva Pereira. "Automated calibration of multiple LIDARs and cameras using a moving sphere". MA thesis. Universidade de Aveiro, 2015.

[4] Rui Filipe Azevedo. "Sensor Fusion of LASER and Vision in Active Pedestrian Detection". MA thesis. Universidade de Aveiro, 2014.

[5] Tully Foote. "tf: The transform library". In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*. Open-Source Software workshop. 2013, pp. 1–6. DOI: 10.1109/TePRA.2013.6556373.

[6] Miguel Oliveira. "Automatic Information and Safety Systems for Driving Assistance". PhD thesis. Universidade de Aveiro, 2013.

[7] Miguel Almeida. "Reconstrução 3D e calibração de lasers no AtlasCar". MA thesis. Universidade de Aveiro, 2011.

[8] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4. ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5980567.

[9] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[10] Chris Urmson et al. "Autonomous Driving in Urban Environments: Boss and the Urban Challenge". In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466. DOI: 10.1002/rob.20255.

[11] Romuald Aufrère et al. "Perception for collision avoidance and autonomous driving". In: *Mechatronics* 13.10 (2003), pp. 1149–1161. ISSN: 09574158. DOI: 10.1016/S0957-4158(03)00047-3.

[12] APC. *APC Smart-UPS 1500VA LCD 230VÂ -Â APCÂ -Â Macedonia*. URL: http://www.apc.com/shop/mk/en/products/APC-Smart-UPS-1500VA-LCD-230V/P-SMT1500I (visited on 07/02/2017).

[13] Cisco Systems. *Cisco SD205 5-port 10 100 Switch - Cisco*. URL: http://www.cisco.com/c/en/us/support/switches/sd205-5-port-10-100-switch/model.html (visited on 06/08/2017).

[14] Cisco Systems. *Quick Start Guide*. URL: http://www.cisco.com/c/dam/en/us/td/docs/switches/lan/csbus/SD205_SD208_SD216/quick_start/guide/SD205_208_216_QuickStartv1.pdf.

[15] FLIR. *Zebra2 GigE PoE HD-SDI Cameras for traffic and security applications*. URL: https://www.ptgrey.com/zebra2-gige-vision-poe-hd-sdi-cameras (visited on 05/23/2017).

[16] LARlabs. *ATLAS project*. URL: http://atlas.web.ua.pt/ (visited on 02/22/2017).

[17] Leddartech. *Solid-State LiDAR for Automotive, Autonomous Driving, ADAS*. URL: http://leddartech.com/automotive/ (visited on 05/23/2017).

[18] S.A. MBP Automóveis Portugal. *Mitsubishi i-MiEV Especificações*. URL: https://www.mitsubishi-motors.pt/imiev/#!especificacoes/tech-spec/electrico (visited on 05/22/2017).

[19] NEXUS. *Servidores NEXUS | P-2308HR3*. URL: http://nexus-solutions.pt/sistemas/servidores-3/p-2308h4-hr4/#! (visited on 07/02/2017).

[20] Open Gov. *An Inside Look: Trials of the SMART Autonomous Vehicle in One-North | 2016-01-13 | OpenGovAsia*. URL: http://www.opengovasia.com/articles/6897-an-inside-look-trials-of-the-smart-autonomous-vehicle-in-one-north (visited on 05/29/2017).

[21] Open Source Robotics Foundation. *Bags - ROS Wiki*. URL: http://wiki.ros.org/Bags (visited on 05/25/2017).

[22] Open Source Robotics Foundation. *rosbag - ROS Wiki*. URL: http://wiki.ros.org/rosbag (visited on 05/25/2017).

[23] Open Source Robotics Foundation. *roslaunch - ROS Wiki*. URL: http://wiki.ros.org/roslaunch (visited on 05/25/2017).

[24] Open Source Robotics Foundation. *ROS.org | About ROS*. URL: http://www.ros.org/about-ros/ (visited on 05/24/2017).

[25] SICK. *SICK United Kingdom | SICK*. URL: https://www.sick.com/gb/en/ (visited on 05/23/2017).

[26] Singapore-MIT Alliance for Research and Technology. *SMART: Singapore-MIT Alliance for Research and Technology*. URL: http://smart.mit.edu/ (visited on 05/29/2017).

[27] UBER Blog. *Steel City's New Wheels | Uber Blog*. URL: https://www.uber.com/blog/pittsburgh/new-wheels/ (visited on 05/29/2017).

[28] Velodyne. *VLP-16*. URL: http://velodynelidar.com/vlp-16.html (visited on 05/24/2017).

[29] Waymo. *Waymo*. URL: https://waymo.com/ (visited on 05/29/2017).

[30] Wired. *Waymo and Lyft Join Forces in the Obsessive Pursuit of Data | WIRED*. URL: https://www.wired.com/2017/05/waymo-lyft-join-forces-obsessive-pursuit-data/ (visited on 05/29/2017).

Appendix A

Apendices

A.1 Power on the sensors and the processing unit (computer)

To power the sensors first you need to start the car, next go to power distribution box in the vehicle's trunk and turn on the circuit breaker in the power distribution board, figure A.1. The sensors and the switch are now powered.

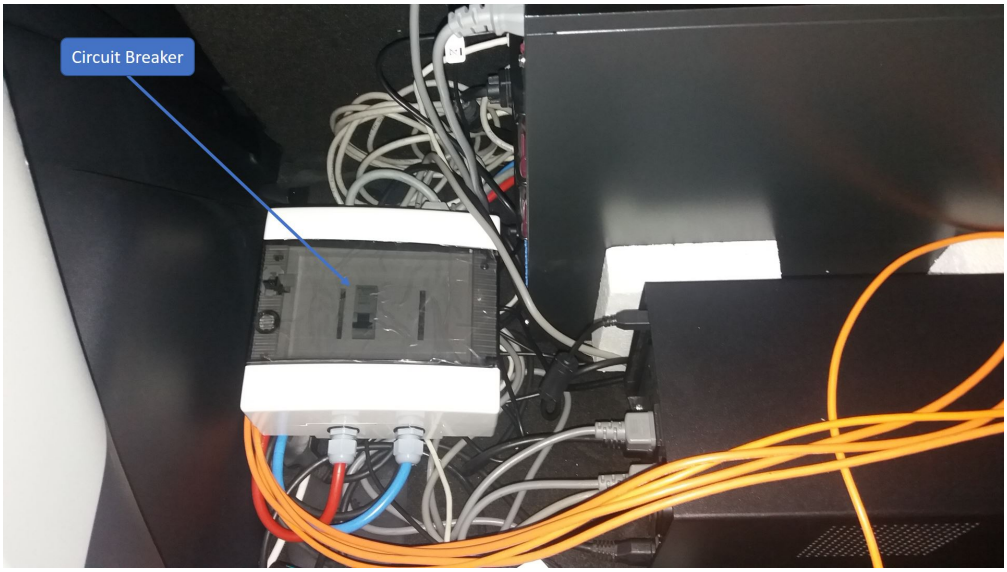
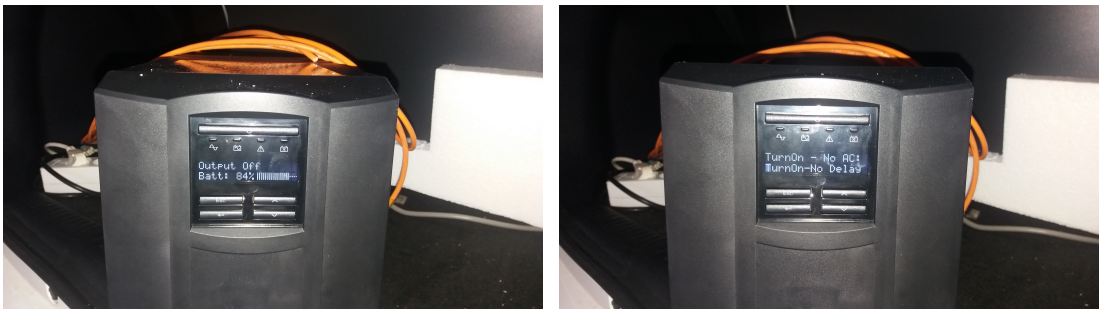


Figure A.1: Power distribution board and circuit breaker on ATLASCAR 2.

To turn on the processing unit first turn on the UPS. Press the UPS power button once, wait for info to appear on the display (see figure A.2a) and then press the power button again, use the arrows to select "No delay" (see figure A.2b) and them press the enter key. The processing unit and the monitor are now powered. The processing unit should start booting automatically, if not, press its power button and be patient. The processing unit takes some time to start.



(a) (b)

Figure A.2: Procedure to activate the power from the UPS.
(a) UPS after pressing the power button once.
(b) UPS after pressing the power button the second time and selection "No delay".

A.2 Point Gray camera connection

For the Point Grey camera to be discoverable in the network, firstly the `flycap` software must be used to set the camera's IP adress. Open a new terminal and type:

```
sudo flycap
```

A window like figure A.3 should appear. Press the button "Auto Force IP" and then press "Yes" in the pup-up window. Wait a few seconds and the camera should be listed in the cameras list of the window. Now you can press the button "Cancel" and launch the driver.

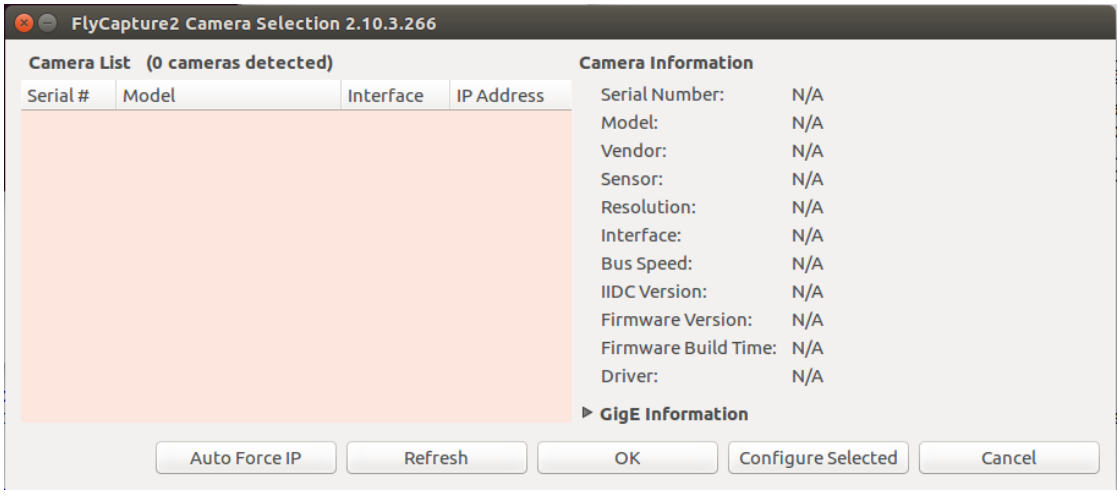


Figure A.3: Flycapture interface.

This procedure must be done before launching the drivers.launch file and every time the camera is pluged-on.

A.3 Free space detection package dependencies

The free space detection package depends on the following packages and libraries:

- csiro-asl-ros-pkg – provides a ROS driver node for the SICK LD-MRS400001 laser scanner;
- RCPRG laser drivers – provides a ROS driver node for the SICK LMS151 laser scanner;
- FlyCapture 2.x Software Development Kit (SDK) – provides a complete software Application Programming Interface (API) library for Point Grey cameras;
- colormap – the colormap ROS package provides similar functionality as MATLAB colormap. Included on Laboratory for Automation and Robotics Toolkit (LARtk);
- lidar segmentation – provides 2D LIDAR segmentation algorithms. Included on LARtk;
- pointgrey_camera_driver – provides a ROS driver node for the Point Grey Zebra2 device;
- velodyne_pointcloud – provides a ROS driver node for the Velodyne Puck VLP_16 LIDAR.

A.4 Free space detection package usage

The free space detection package is used with `roslaunch`. Is has 3 launch files:

1. `drivers.launch`;
2. `get_data.launch`;
3. `free_space.launch`;

The first one is launched to connect to the sensors. It can connect with five sensors:

- One Sick LD-MRS;
- Two Sick LMS-151;
- One Point Grey Camera;
- One Velodyne Puck VLP_16.

By default it connects to all of them, but it can be set to launch only a specific set of drivers. Each one of the sensors mentioned above is associated with a boolean argument that can be passed to the launch file, if that argument is set to "true" then the driver is launched, if is set to "false" the driver is not launched. By default they are all set to "true". The following example would launch the drivers only for booth Sick LMS-151:

```
1  roslaunch free_space_detection drivers.launch VLP16:=false
   zebra2:=false lms151_E:=true lms151_D:=true ld_mrs:=false
```

The second launch file is used to visualize and/or record data from the sensors. This launch file will also launch the drivers.launch so it accepts the same arguments plus an extra one to active or deactivate the recording option. By default it is activated. When launched it will open `Rviz` and subscribe to the raw data topics of the sensors, if the recording option is activated it will also launch `rosbag` that will record all raw data topics in a folder called "laserData" nested on the package folder. The following example will launch all the drives for the sensors, display it on `Rviz` and start recoding it.

```
roslaunch free_space_detection get_data.launch record:=true
```

Finally the third launch file is used to start the data processing. Opposite to what happens with the "get_data.launch" this launch file does not launch the drivers for the hardware so, to use this launch file first is necessary to launch the drivers for the sensors using launch file number one or having the raw data topics of the sensors being publish by other node like `rosbag`. When launched it will publish the processed data and open `Rviz` were it can be visualised. It also as the option to record the raw data which can be selected thorough the same parameter as launch file number two. This launch file also publish three parameters, the reference sensor used in the calibration procedure, indication if is to include the physical limits of the vehicle's steering in the free space representation and other to select which scans to merge. The following example would launch the processing nodes with recording the data:

```
roslaunch free_space_detection free_space.launch
```