Universidade de Aveiro Departamento de Engenharia Mecânica
2018

Tiago Simões Marques

Deteção da Navegabilidade da Estrada para o ATLASCAR2 usando LIDAR e Inclinometria

Detection of Road Navigability for ATLASCAR2 using LIDAR and Inclinometer Data

Universidade de Aveiro Departamento de Engenharia Mecânica
2018

Tiago Simões Marques

Deteção da Navegabilidade da Estrada para o ATLASCAR2 usando LIDAR e Inclinometria

Detection of Road Navigability for ATLASCAR2 using LIDAR and Inclinometer Data

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 com Agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Miguel Armando Riem de Oliveira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President

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

Vogais / Committee

Doutor Cristiano Premebida
Investigador da Universidade de Coimbra - Faculdade de Ciências e Tecnologia (arguente)

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

Agradecimentos / Acknowledgements

Quero agradecer especialmente ao professor Vítor Santos por todo o acompanhamento feito ao longo deste projeto como orientador. Durante todo este periodo sempre demonstrou uma exemplar dedicação e envolvimento em todas as etapas do processo, principalmente através das discussões importantes e enriquecedoras que tivemos. É com muito gosto que entrego esta dissertação, tendo tido o professor como orientador, pela enorme paixão e entuasiamo que demontra pelo seu trabalho que se torna contagiante e motivadora.

À minha família, quero agradecer a oportunidade que me deram de poder realizar o curso de Engenharia Mecânica e por todo apoio dado ao longo deste percurso académico.

Por fim, queria também agradecer aos colegas do LAR por todos os momentos que passámos juntos, pelo bom ambiente criado e pelas discussões e troca de ideias que tivemos entre nós que contribuiram para a realização deste trabalho.

Palavras-chave

Point density; Point Cloud; Curb detection; ATLASCAR2; LIDAR; Laser Projection; ROS

Resumo

Este trabalho está inserido no projeto *AtlasCar2*, e pretende criar mecanismos para a deteção da área navegável pelo veículo, delimitada pelos passeios ou barreiras nas extremidades da estrada, e por outros pequenos obstáculos como pinos verticais ou buracos no asfalto, que em conjunto condicionam a navegação do veículo. Assim este trabalho prevê a obtenção de uma nuvem de pontos que delimita estes obstáculos, para que esta depois seja utilizada em futuros algoritmos de navegação. A abordagem passou primeiro lugar, pela criação de uma reconstrução local da zona em frente ao veículo, utilizando para o efeito, leituras sucessivas efetuadas pelo sensor LIDAR SICK LD-MRS400001. Para conseguir fazer essa reconstrução foram desenvolvidos sistemas para estabelecer: o posicionamento global do veículo, recorrendo ao sistema de GPS do*AtlasCar2*; e a sua orientação relativa ao plano da estrada, obtida através do sistema de inclinometria, também este presente no veículo. Foi desenvolvido um método de filtragem que permite extrair da reconstrução criada, informação correspondente aos obstáculos à navegação presentes na estrada. Esta filtragem e extração é feita através de uma nova abordagem que se baseia no estudo da densidade de acumulação dos feixes laser consoante a topologia do terreno em que estes incidem. Tendo também sido desenvolvido um simulador que permite estudar estes comportamentos num ambiente controlado afim de avaliar o desempenho da acumulação em várias situações, incluindo diversas configurações do sensor e dos obstáculos na estrada. Por fim são também apresentados resultados que avaliam individualmente os sistemas de aquisição de posição e orientação desenvolvidos, bem como os resultados da sua integração na criação da reconstrução da estrada, e os resultados das simulações que detalham o comportamento da accumulação em várias situações. São também apresentados os resultados dos algoritmos de filtragem; numa primeira fase em que foram utilizados parâmetros constantes, e depois, utilizando parâmetros dinâmicos que variam em função da velocidade do veículo, por forma a combater algumas das limitações do método estático. A metodologia para a deteção da área navegável desenvolvida neste trabalho, embora ainda com algumas limitações, mostra ser capaz de corretamente identificar em tempo real os limites da estrada e outros obstáculos, com um custo computacional relativamente baixo, gerando uma nuvem de pontos pronta para ser usada em algoritmos de navegação.

Keywords Point density; Point Cloud; Curb detection; ATLASCAR2; LIDAR; Laser Projection; ROS

Abstract This work is developed as part of the *AtlasCar2* project, and intends to create mechanisms for the detection of area traversable by the vehicle, delimited by the side-walks or road barriers and other small obstacles, such as road delineators, or small holes on the road surface, that together condition the vehicle's navigation. As such this work aims to obtain from the lasers a point cloud that delimits these obstacles, so that it can be used in future navigation algorithms. The approach was to first, create a reconstruction of the environment ahead of the vehicle, using for this purpose the SICK LD-MRS400001 LIDAR sensor. To achieve this reconstruction, systems were developed to establish: the global position of the vehicle, using the GPS system of the *AtlasCar2*; and its orientation relative to the road plane, obtained through the inclinometer module also present in the vehicle. A filtering method was developed that allows to extract, from the created road reconstruction cloud, the information regarding the obstacles to the navigation, present in the road. This filtering method is performed through a new proposed approach based on the study of the accumulated density of the laser beams, dependent on the topology of the terrain they hit. A simulator was also built to study these accumulation behaviours in a controlled environment, in order to evaluate the performance of the accumulation in various situations, including several configurations of the sensor and the road obstacles. Lastly, some results testing the individual performance of the position and orientation systems, are presented, as well as the results of their integration in the road reconstruction system. Results of the performed simulations detailing the accumulation behaviour in various situations, are also presented, along with the results of the filtering algorithm: first using static filtering parameters, and then using dynamic ones, that change based on vehicle velocity, aimed at solving some of the limitations of the static method. The methodology for delimiting the road area accessible by the vehicle, despite some limitations, shows capable of accurately identifying, in real time, the road boundaries with relatively low computational cost, generating a point cloud ready to be used in navigation algorithms.

Contents

List of Tables

List of Figures

Acronyms

AD Autonomous Driving. 23

APCD Accumulated Point Cloud Density. 37

ECU Electronic Control Unit. 2

GNSS Global Navigation Satellite System. 18

GPS Global Positioning System. 18

IMU Inertial Measurement Unit. 18

LAR Laboratory for Automation and Robotics. 1

MLS Mobile LIDAR System. 4

PCL *Point Cloud Library*. i, 21

ROS *Robot Operating System*. i, 20

SPAN Synchronous Position, Attitude and Navigation. 18

Chapter 1

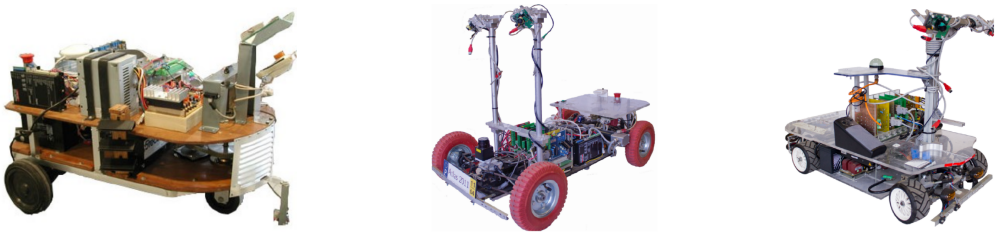
Introduction

During the last decade there has been a huge interest in autonomous vehicle (AV) technology from both the academic and industry world. This interest derives mostly from the possibility for this technology to fundamentally change transportation. Cars built with this technology will likely reduce crashes, energy consumption, and pollution[1] solving some of the major problem with vehicles today. Consequentially, there is a valuable need to develop and/or improve mechanisms that allow the vehicle to "see" the surrounding environment so that it can make the right decisions on what action to take.

In this context, this dissertation provides a method for the positioning of the autonomous vehicle, both on a local and global frame, as well as a new approach for the road edge and obstacle detection, based on the density of data gathered by a LIDAR sensor positioned on the lower half of the vehicle, that allows to create the region of road navigability, delimited by these obstacles.

1.1 The ATLAS Project

This dissertation is part of the ATLAS project, established in 2003 by the Group of Automation and Robotics at the Laboratory for Automation and Robotics (LAR), on the Department of Mechanical Engineering of the University of Aveiro. The mission from the very beginning was to create and develop advanced sensing and active systems designed for implementation in autonomous vehicle applications[2]. The first projects of the LAR team involved developing scaled prototypes (figure 1.1), which awarded the various teams along the years several prizes in autonomous driving competitions at the Portuguese National Robotics Festival.



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

Figure 1.1: Atlas autonomous robots.

The project later migrated to a full-sized vehicle, named *AtlasCar* (figure 1.2) where the knowledge and experience gathered from the development of these small prototypes was used create a vehicle equipped with state of the art equipment, that was able to perceive its surroundings [3, 4] and react to them using a plethora of actuators to move its mechanical components[5].

Figure 1.2: *AtlasCar* Vehicle

After years of modifications, hardware expansions and with the car approaching the end of its life cycle, the need arose to once again upgrade the platform to a more modern one. In 2015 the University of Aveiro acquired a Mitsubishi i-MiEV that would become the new iteration of the *AtlasCar* (figure 1.3). This more modern and electric vehicle has several important advantages, since power can be directly gathered from its batteries. The presence of a Electronic Control Unit (ECU) means that data about the state of vehicle can be directly gathered from the sensors already in the car eliminating the need for additional ones, and that it will be much easier to control, hopefully without the need for mechanical actuators.

Figure 1.3: AtlasCar2 Electric Vehicle

1.2 Problem Context

With the continuous development of the self-driving technologies in the ATLAS projects, more varied and sophisticated methods to perceive the environment surrounding the vehicle are necessary, so that the system can have reliable information on which to base its decision making process.

In the context of autonomous driving applications, there is an important need to identify obstacles at ground level that condition the position of the vehicle in the road. These objects can either completely constrain the car's movements to a certain limit, hard obstacles, such as road curbs and side-walks, or can simply indicate that small corrections to the vehicle's course need to be applied, soft obstacles, such as small holes or random items on the road.

The planned work focusses on designing a solution which will allow the *AtlasCar 2* to map and identify the various obstacles on the ground plane, primarily focussing on hard obstacles since in the early stages of the project these have more relevance towards achieving the self-driving goal. This feature extraction will be based on the point cloud data gathered by a four beam LIDAR sensor positioned on the lower half of the vehicle. However, given the nature of the sensor and its placement, it is natural that these measurements will be affected by the movement of the car so a solution to correct the sensor measurements will also need to be devised.

Additionally, given the need to perform a reconstruction of the road ahead of the vehicle, the need arises to have a method for locating the vehicle relative to a static global frame so that the environment can be correctly represented relative to that frame. Since the *AtlasCar 2* is a relatively new project, there is not a solution already in place to do so, therefore a method for determining the vehicle's location will also be put in place.

1.3 Objectives

The main goal of this project is to develop a methodology that will allow for a real-time reconstruction of the road ahead of the vehicle and from that reconstruction to extract the information relative to the road boundaries and other obstacles, so that this information could later be used in a trajectory planning algorithm.

As mentioned briefly in section 1.2, to achieve this, a module for determining vehicle global localization will also need to be developed as well as the integration of the orientation module already implemented in the *AtlasCar 2*, to correctly interpret the data gathered by the LIDAR sensor. This data will be assembled into a point cloud representing a reconstruction of the section of the road perceived by the sensor, that will serve as the data set from which the road features will be extracted.

1.4 Document Structure

This thesis is composed of six chapters, including the present chapter.

Chapter 2 provides a description of the main hardware and software tools used in this project.

Chapter 3 presents the proposed solution for the road reconstruction problem. In it, the integration of the orientation module is explained and the method for vehicle local-

ization is presented, also in this chapter an overview of the infrastructure is performed showing its reference frame structure and the system used to accumulate the scan data is explained.

Chapter 4 addresses the road edge detection problem and describes the methodology proposed to solve it, and the fundamentals behind the approach.

Chapter 5 is dedicated to the analysis of the method described in 5 and showing the results in terms of capabilities and limitations of the overall road edge detection system that derived from merging the systems described in the two previous chapters.

Chapter 6 presents the conclusions and final considerations for the developed work, along with some suggestions for future work.

Finally, Appendix A.1 contains instructions to install and use the developed package.

1.5 Related work

Given the fundamental necessity for detecting the road boundaries in autonomous driving applications, many researchers along the years have propose solutions to tackle this problem. In this section some of these techniques will be presented and discussed in terms of their results and applicability in the *AtlasCar 2* project.

Additionally a brief presentation will be made about de work already developed by former LAR students for the *AtlasCar 1* on the theme of road reconstruction and feature extraction.

1.5.1 Road edge detection techniques

The problem of road or lane perception is a crucial enabler for advanced driver assistance systems. As such, it has been an active field of research for the past two decades with considerable progress made in the past few years. The problem was faced under various scenarios, with different task definitions, leading to usage of diverse sensing modalities and techniques [6].

The two main approaches to road detection usually applied in autonomous driving applications are vision based or laser based. Despite the difference in the hardware used, inspection of the lane and road detection literature reveals that both these approaches share the main functional modules, though these modules are of course implemented differently in different systems. Figure 1.4 represents the general methodology used to tackle the road detection problem.

Despite the clear advantage of vision based systems in lane mark detection, this approach still has major difficulties when facing less then ideal conditions. Situations where the lane markings are not present, poor visibility conditions and the high reliability demands of AD applications [6], pose serious difficulties for vision based systems.

Due to these limitations, laser based applications are becoming increasingly popular. Mobile LIDAR System (MLS) is a newly emerging technology which collects 3D information of objects while vehicles drive at a posted speed. It becomes more and more popular in analysing 3D point clouds because of its high density, efficiency and cost-effectiveness and provides the possibility to extract the micro-objects such as road curbs [7].

A straightforward method for road curb detection usually makes use of elevation information. For example, most common algorithms such as [8] focus on detecting

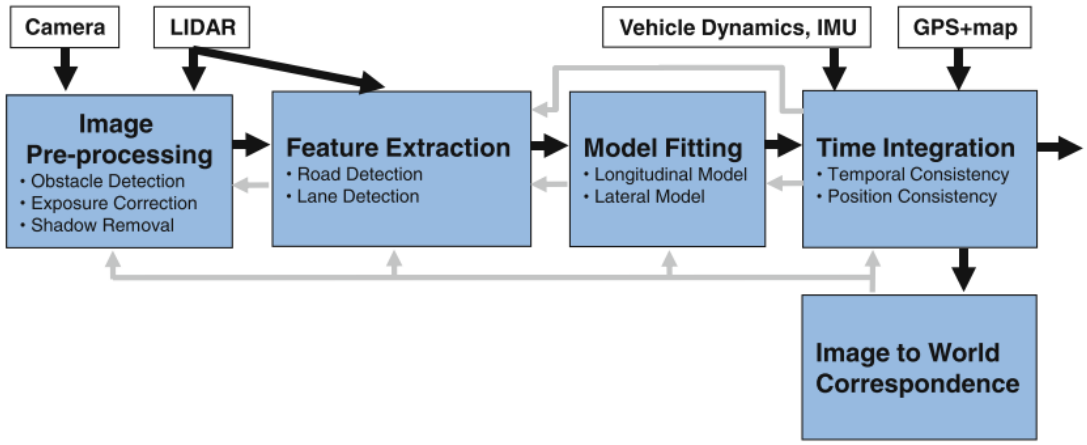
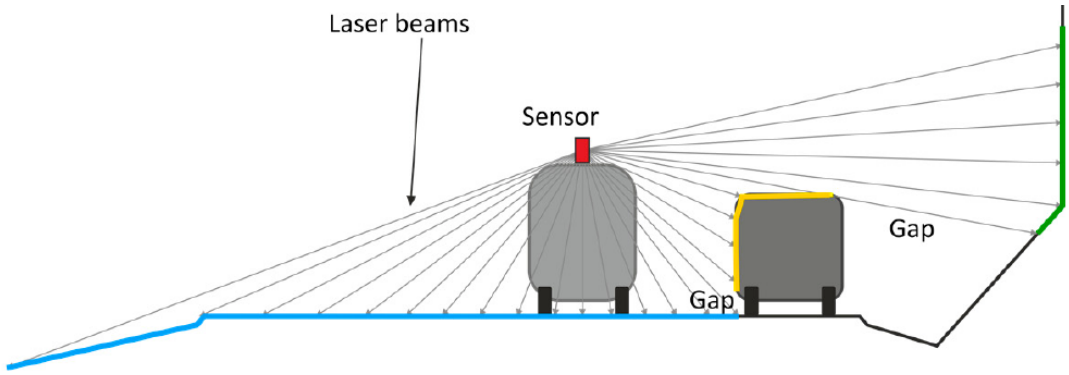
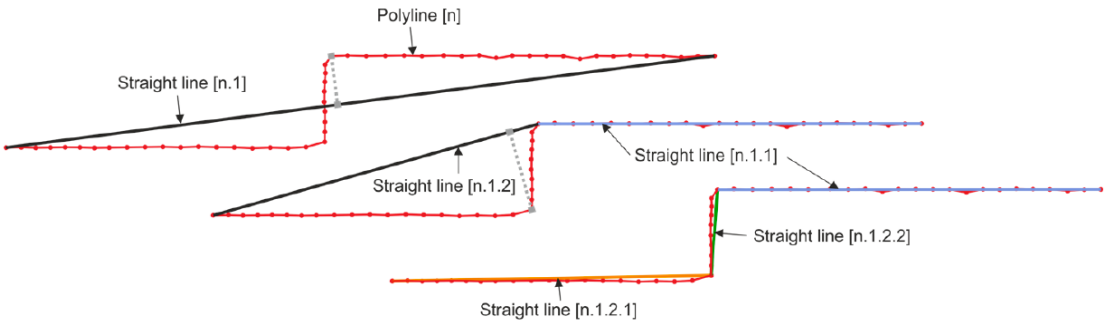


Figure 1.4: Generic approach for road lane detection [6].

objects in terms of the elevation difference (figure 1.5) resulting from the projections of the 3D data gathered by a top mounted high density LIDAR into a 2D plane.



(a) Perspective view of surfaces gathered from the laser scan.



(b) Method for tracing the curb contour.

Figure 1.5: Line cloud approach to detect road curbs for MLS applications

A similar method was explored in [9], and this framework employed in the "Boss" vehicle for the DARPA Urban Challenge to show its robustness and efficiency. This

approach uses a point array gathered by a two-dimensional LIDAR, mounted on top of the vehicle with a downwards pointing angle. This data is then processed according to the cascade method in figure 1.6 to identify road and road-edge detection.

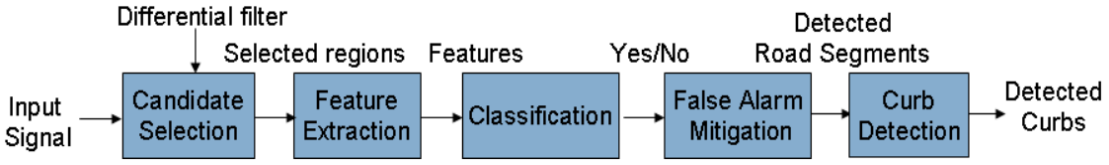


Figure 1.6: Flowchart of road/road-edge detection algorithm employed in [9].

The road and road-edge points are first detected in elevation data, and then validated on the ground plane. An elevation-based road signal is extracted from the LIDAR range data (figure 1.7 top left). The signal is processed by a filter (figure 1.7 top right) to select candidate regions for road-segment classification (figure 1.7 bottom). In each candidate region, features are extracted and classified as road-segment or not. The positive results are passed through the false alarm mitigation module using a rule based scheme (e.g., minimal road width requirement) to further reduce false alarms. The detected road edges are the left-most and right-most boundaries of the detected road-segments. The LIDAR range data is also projected onto the ground plane to identify curb lines using a road curb detection module. The line representation of the projected points is identified and compared to a simple road model the top-down view to determine whether the candidate region is a road segment with its road-edges.

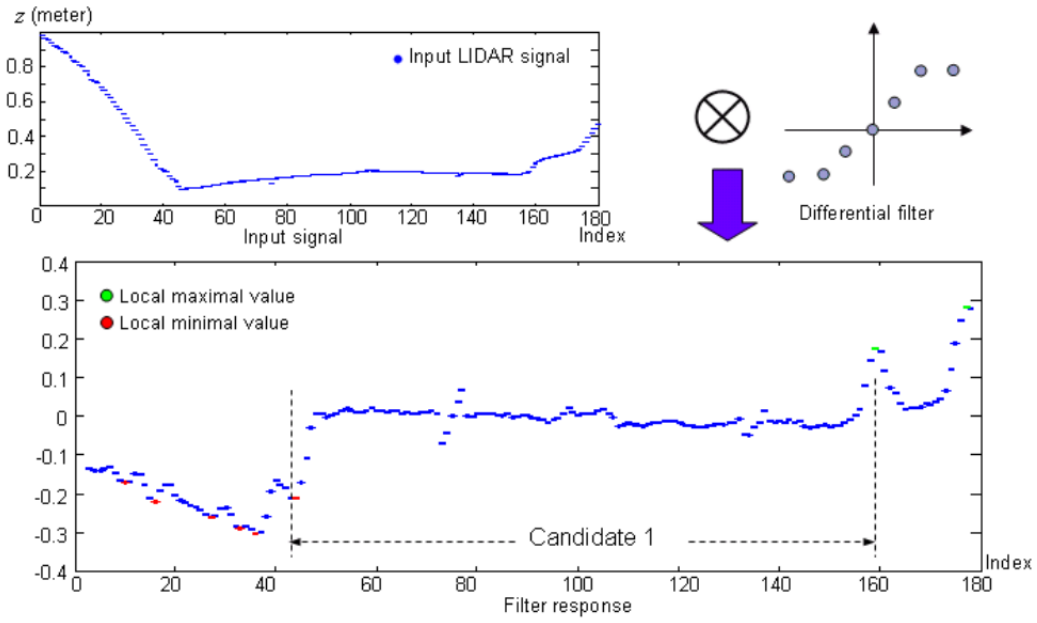


Figure 1.7: Candidate road-segment region selection

These algorithms then gather the points belonging to the curb and accumulate them according to the movement of the vehicle, figure 1.8, and in the case of [8] filter them to eliminate noise as demonstrated in figure 1.9.

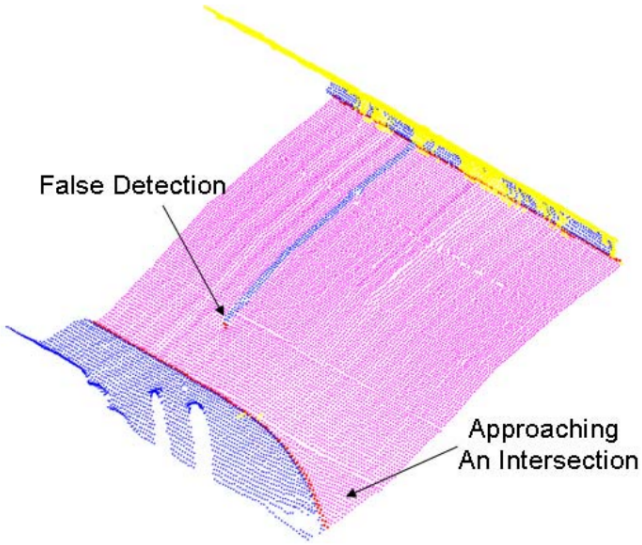


Figure 1.8: Sample detection results in multiple scans. The detected road points, road-edge points and curb lines are in magenta, red and yellow, respectively [9].

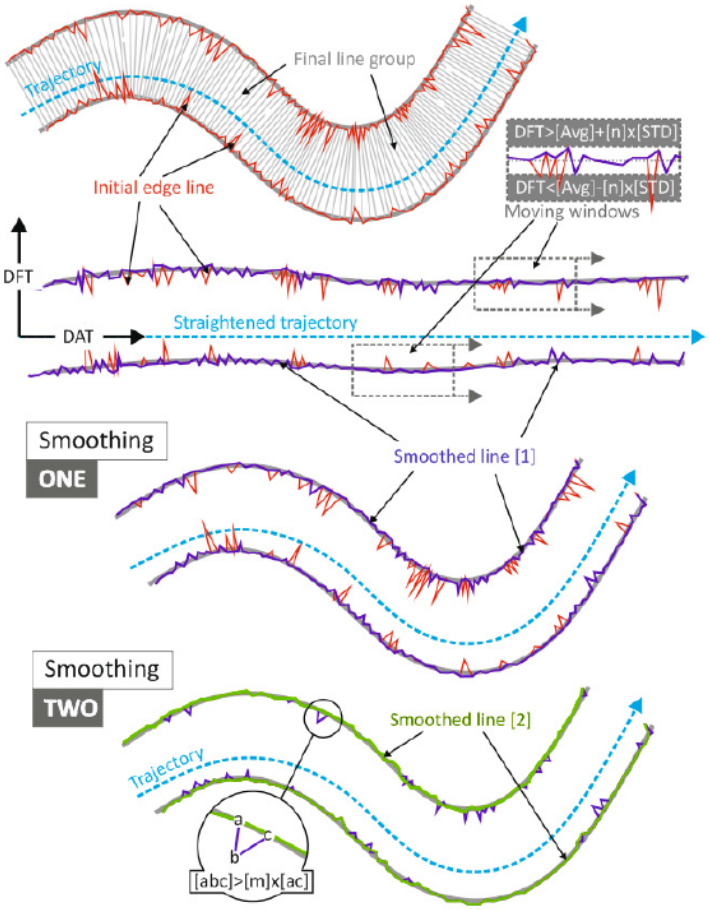
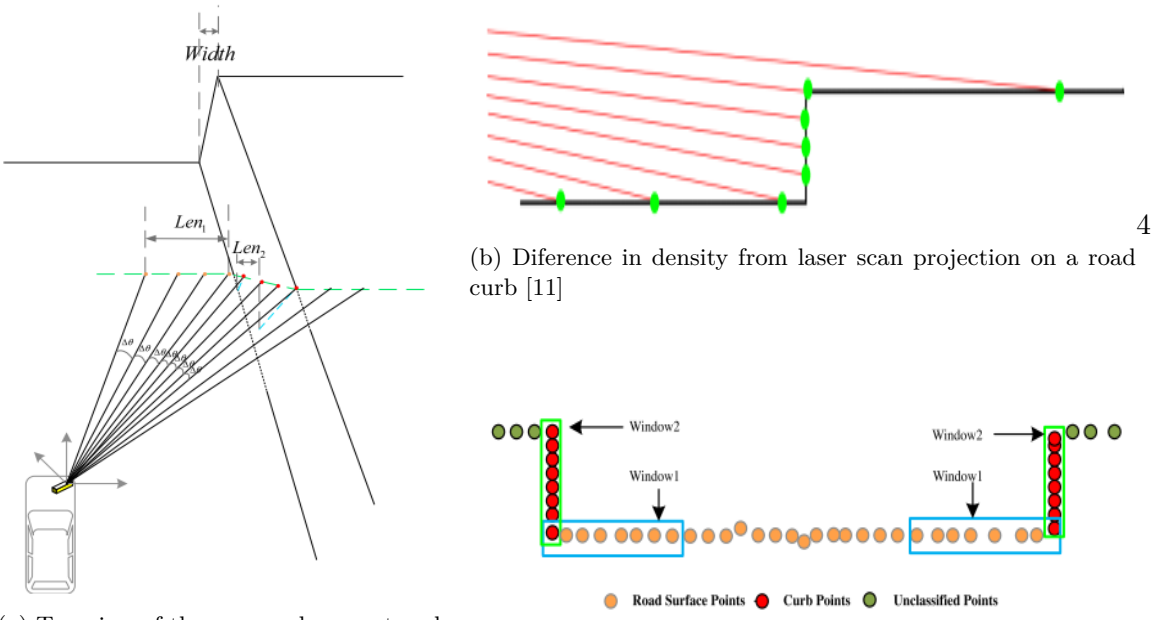


Figure 1.9: Example of line road edge polyline smoothing on a hypothetical road section [8].

These methods have fast processing speeds [9], and produce attractive results in the straight roads with the same elevation, but fall short in occluded, sunken or uphill road areas.

Other researchers [10] following a similar method of analysing the laser scan cross section, experimented with other sensor placements taking advantage of the geometry produced by the laser projection (figure 1.10). This feature when coupled with the elevation jump and the slope change characteristic of the road curbs produced very good results in correctly identifying the road limits.



(a) Top view of the sensor placement and scan projection on a road curb [10]

(b) Diference in density from laser scan projection on a road curb [11]

(c) Curb cross section showing the surface classifiers [10]

Figure 1.10: Point density element of the method used by [10].

All these methods are based in data gathered by a 2D LIDAR or use 3D LIDAR data but project it into a 2D plane, and from that extract the road curbs. However, the projection causes loss of 3D information, which degrades the performance of the detection, making them unsuitable to use on data sets with different geometrical features, such as large slopes or uneven road surface [7].

Recently, in [7] a method is proposed that utilizes accurate, and efficient methods to extract curbs from 3D mobile LIDAR point clouds.

A gradient concept to 3-D point clouds is employed in this method, through considering the points density in a local area. The method first, generates voxels for the point cloud. Then, the intensity of each voxel is defined by the points density, i.e., the number of points inside the voxel. Finally, the 3-D sampling density gradient is calculated by the difference of the intensity between adjacent voxels. The intensity is approximated by the number of points in a local area. This gradient proves to better represent the real situation, relative to the other approaches in which the gradient definition is based on elevation difference between adjacent points. In figure 1.11(a), representing the approach used in previous works, the elevation along z-axis is increasing evenly, so the gradient is a constant along z-axis and zero along y-axis. However, the gradient along x-axis varies

due to different elevations. This is not desirable, because the gradients along the normal direction of the facade (i.e., along x-axis) are different, as shown in figure 1.11(a). The proposed definition, presented in this paper, of 3D sampling density gradient is based on the intensity difference between the adjacent points, and gradient is zero along either y-axis or z-axis and a constant along x-axis, as shown in figure 1.11(b).

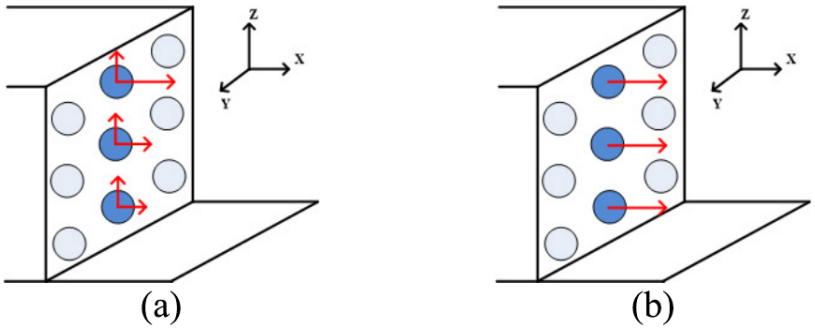


Figure 1.11: Different definitions of the 3-D gradient. (a) Traditional methods using elevation gradients. (b) Proposed method intensity gradients. [7]

From the fact that the curb is parallel to the y-axis and yoz plane (vertical plane), and roadway and side-walk are parallel to the xoy plane (horizontal plane), three sampling density gradients where defined, G_x, G_y, and G_z, and based on this concept three primary situations where conceived:

- The voxel lays within the surface: there is only one large gradient, such as the large G_x in the curb areas and the large G_z in the roadway or side-walk areas.
- The voxel lays in the intersection of two surfaces: there are more than one large gradient, such as the large G_x and G_z along the curb edges.
- The voxel lays in the intersection of three mutually nonparallel surfaces: all gradients are large, such as the curb corners.

Following this, a energy function was developed to be applied to each voxel in order to determine, based on the gradient direction, to each of the three situations above the voxel belongs to. This results in a list of candidate curb points that needs to be refined by a method that seeks to eliminate the non curb voxels from the candidate list by finding the optimal path that constitutes the curb formed from the candidate voxels. The result is a 2D grid formed by the voxels attributed to each of the three situations (figure 1.12).

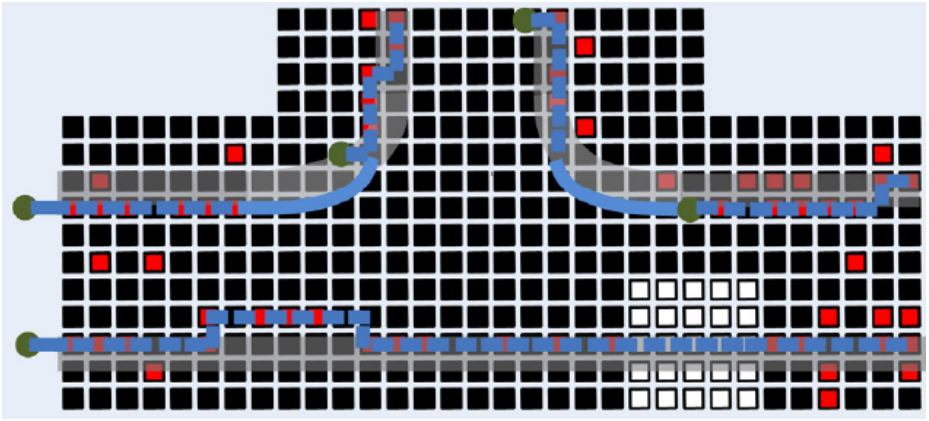


Figure 1.12: Result of the refinement of the candidate points. [7]

Figure 1.13 represents the identified curbs in red during an intersection, overlaid in the dataset point cloud.

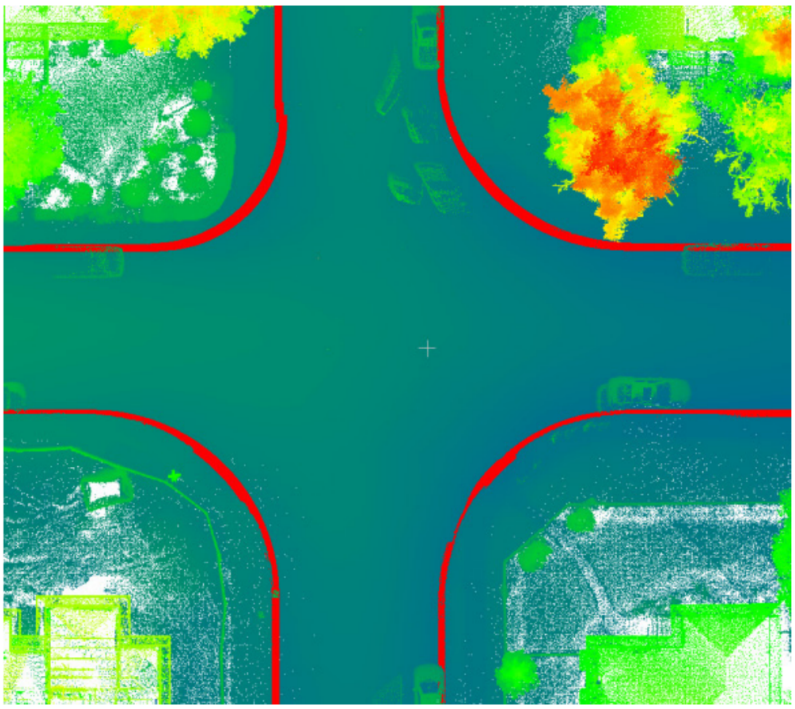


Figure 1.13: Results of curb detection on an entire crossroad. [7]

This algorithm proves to work effectively for large-scale mobile LiDAR point clouds. Different quantitative evaluations, indicate that this method is more accurate than the existing algorithms [7]. However, this methodology requires very large and dense point clouds in order to provide sufficient data to evaluate the gradients. For reference this was built for the use in a vehicle equipped with a *Riegl VMX-450* system, which has two *Riegl VQ-450* 360° LIDAR sensors, capable of providing 1.1million measurments/sec of laser data [12].

Different approaches even use convolutional neural networks to extract the road features from the provided point cloud, with reasonably good results [13], although these are relatively slow and not yet reliable enough to be applied in AD applications.

1.5.2 Local map reconstruction by *P. Salvado*

Within LAR, several solutions have been presented along the years relating to the reconstruction of the environment around the car. From these, the most relevant and closest to the work proposed in this dissertation is the one developed by *P. Salvado* as part of the *AtlasCar 1* project [14]. His approach for reconstructing the road ahead of the car involved the use of a *Hokuyo* 2D LIDAR sensor located on top of the vehicle at an angle in order for the laser beam to hit the ground plane at some distance in front of the vehicle, these measurements where then accumulated forming a 3D map of the perceived road.

In order to accurately accumulate the point cloud, the system needed to know the pose of the vehicle so that it could apply the correct transformation between the sensor frame and the world frame. Obtaining the instant pose of the vehicle was achieved through vehicle odometry, based on the *Ackerman* kinematic model (figure 1.14). The data to calculate the vehicle's odometry was gathered by an encoder placed on the rear axis left wheel to calculate the car's speed, and a potentiometer in the steering wheel column to measure the orientation of the wheels.

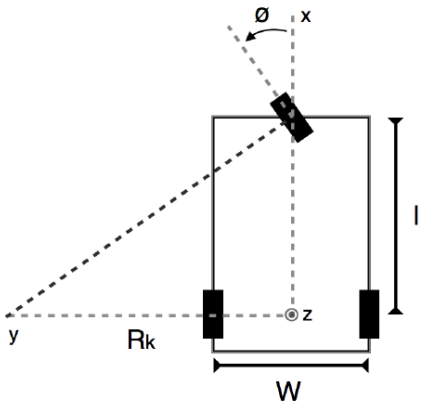
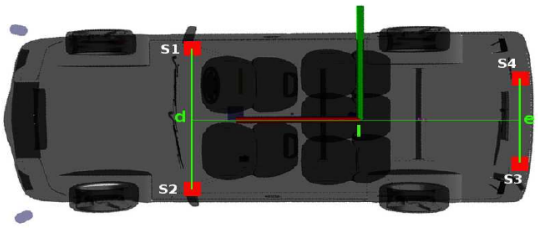


Figure 1.14: Simplified *Ackerman* bicycle model.

Additionally because of the method and hardware used, this system was very vulnerable to changes in the orientation of the car, in cases of acceleration, deceleration or sharp turns. So a system to correct the laser measurements was also created, using four *Sharp GP2D12* analogue distance measuring sensors to calculate the angle of the car relative to the road. Figure 1.15 illustrate the placement of the sensors and the method to calculated the angles.



(a) *Sharp GP2D12* sensor placement

$$roll = \arctan\left(\frac{\frac{s1+s4}{2} - \frac{s2+s3}{2}}{\frac{d+e}{2}}\right)$$

$$pitch = \arctan\left(\frac{\frac{s3+s4}{2} - \frac{s1+s2}{2}}{l}\right)$$

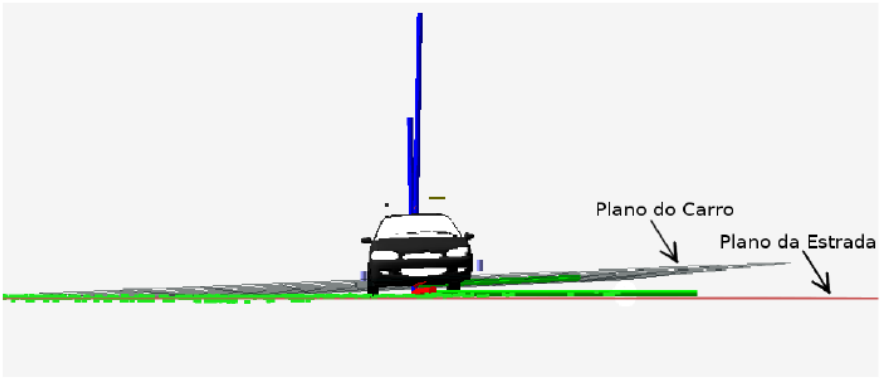
$$z = \frac{s1+s2+s3+s4}{4}$$

(b) Equations for pitch and roll calcculation.

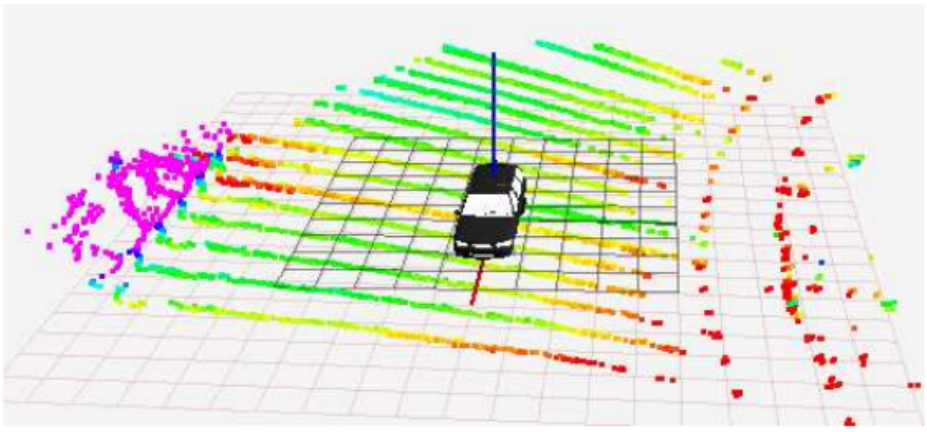
Figure 1.15: Method for calculating vehicle orientation relative to the road [14]

Given the nature of the optoelectronic *Sharp GP2D12* sensors, this measuring system was prone to a lot of noise which required a filtering of the data gathered by the sensors before it could be used to calculate the vehicle's roll and pitch, adding an additional layer of processing and thus slowing the measuring rate.

The result of combining the orientation module and the 2D laser accumulation can be observed in figure 1.16.



(a) Frontal view of the recontruction with the orientation system in place



(b) Local reconstruction of the road on turn.

Figure 1.16: Local road reconstruction performed through the method developed in [14] using a downwards pointing 2D LIDAR sensor.

By observing these results, we can see that the generated point cloud has a considerable gap between measurements, which is not ideal when trying to extract features from that point cloud; furthermore, the method for vehicle localization still has some limitations mainly due to the error accumulation when integrating the car's velocity, and the vehicle orientation measuring system should be more robust, as pointed out by *P. Salvado* in his conclusions [14].

1.5.3 Inclination module by *Armindo Silva*

In 2017, as a way of improving the system proposed by *P. Salvado*, *Armindo Silva* [15] developed a more robust method to measure the vehicle's orientation in its pitch and roll components.

The system uses four SENSICK DT20 Hi optoelectronic sensors placed in the four bottom corners of the car, to measure the distances to the ground plane.

These sensors are connected to an *Arduino nano* which serves as the processing unit that receives the distance data from the sensors as the input and converts them into the orientation angles. First, a calibration is needed to offset the placement of the sensors along the Z axis, then, since the processing unit has the distance between the sensors hard coded, the trigonometric relation (equation 1.1) is applied, and its results are the values for roll and pitch which are outputted to the installed LCD screen. Figure 1.17 serves as a graphical representation of the described method.

$$angle = \sin\left(\frac{sum\ of\ sensor\ readings}{distance\ between\ sensors}\right) \tag{1.1}$$

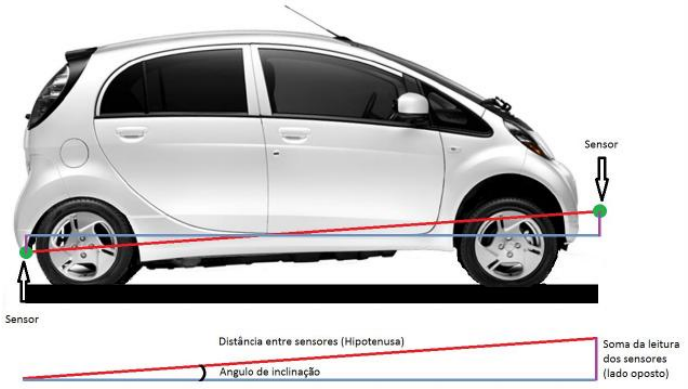


Figure 1.17: Method for determining roll and pitch of the vehicle.

Chapter 2

Experimental Infrastructure

This chapter describes in detail the tools used in this project, both in the form of hardware and software. The hardware presented in Section 2.1 are the sensors that were used to gather various data about the *AtlasCar2* and its environment, and in Section 2.2 the placement of these sensors is shown and detailed. Lastly in Section 2.3 the main software tools that serve as a base for the dissertation are described in detail.

2.1 Hardware

2.1.1 SICK LD-MRS400001

The SICK LD-MRS400001 3D laser scanner (Figure 2.1), is a four beam long range LIDAR scanner capable of measuring distances up to 250m, designed for outdoor use, given its ability to measure distances through glass, fog and dust (multi echo technology) and its IP69K enclosure rating. It has four scanning planes spaced by 0.8° between them and two settings for aperture, 85° if all four are used or 110° if only two planes are used, as shown in the Figure 2.2 diagram. Table 2.1 lists SICK's LD-MRS400001 most relevant features.

Table 2.1: SICK LD-MRS400001 specifications [16]

Field of application	Outdoors
Laser class	Class 1, IEC 60825-1 (2007-3)
Scan planes	4 with a total aperture of 2.4°
Aperture angle	4 scan planes: 85° | 2 scan planes: 110°
Scanning frequency	12.5 Hz | 25 Hz | 50 Hz
Angular resolution	0.125° | 0.25°| 0.5°
Operating range	0.5 m to 250 m
Max. range with 10 % reflectivity	50 m
Systematic error	±300 mm
Statistical error	100 mm
Interface	Ethernet
Weight	1 Kg

Given that the usage scenario for this sensor is to scan the road in front of a moving vehicle, one other important specification is the scanning frequency. This sensor presents three possible configurations for frequency coupled with angular resolution: *a*) 12.5 Hz with 0.125° angular resolution, *b*) 25 Hz with 0.25° angular resolution, *c*) 50 Hz with 0.5° angular resolution.

Figure 2.1: SICK LD-MRS400001 Sensor.

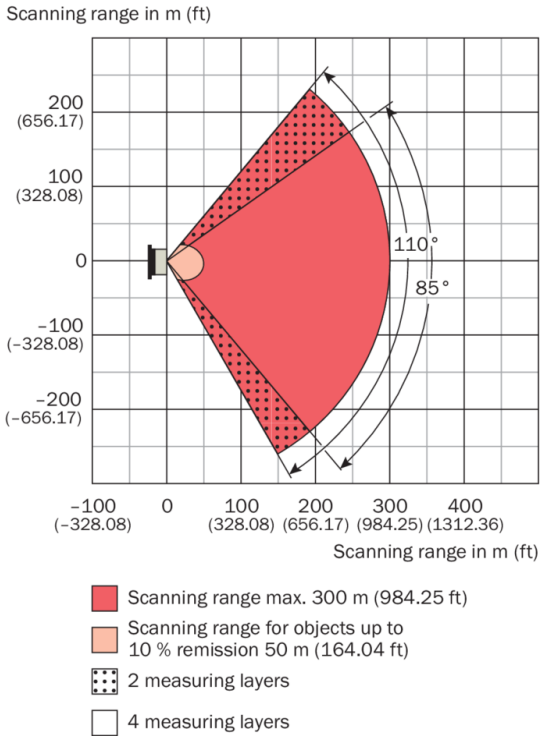


Figure 2.2: SICK LD-MRS400001 operating range.

Due to the need for 3D information, the sensor present in the *AtlasCar2* is configured to use the four scanning planes with 85° aperture, and a scanning frequency of 50 Hz, limiting angular resolution to 0.5°, but in this case having a higher frequency is more valuable. As such, this sensor will be responsible for acquiring the data about the environment in front of the vehicle, which will be used for the road boundary detection.

2.1.2 SICK DT20 Hi

SICK DT20 Hi (Figure 2.3) is a optoelectronic sensor to measure distances between 50mm and 1000mm also ready for outdoor use. It makes use of a red laser beam that allows for precise and reliable measurements, having a ± 1 mm linearity and ≥ 0.25 mm repeatability [17], independent of colour and roughness of the target object. Additionally its very high measuring frequency of up to 400Hz (2.5ms response time) is a great advantage given that the laser is to be mounted on a moving vehicle. The higher frequency will allow to neglect the error in the measurements caused by the difference in the position of the sensor during the emission and remission steps of the measuring process.

Despite the simple working principle this sensor has a lot of extra options that can be changed in the menus and display integrated in the hardware, these options are described in the *Additional function* section of Table 2.2 and make this sensor very configurable and adjustable to our needs.

Figure 2.3: SICK DT20 Hi Sensor.

Table 2.2: SICK DT20 Hi specifications [17].

Measuring range	50 mm to 1000 mm
Repeatability	$\geq$ 0.25 mm
Linearity	$\pm$ 1 mm
Response time	$\geq$ 2.5 ms
Measuring frequency	$\leq$ 400 Hz
Light source	Laser, red
Laser class	2 (IEC 60825-1:2014, EN 60825-1:2014)
Typ. light spot size (distance)	3 mm x 6 mm (300 mm)
Analogue output	1 x 4 mA to 20 mA ($\leq$ 300 Ω)
Enclosure rating	IP65
Indication	LC display, 2 x LED
Weight	135 g
Additional function	Set moving average fast/medium/slow, switching mode: distance to object (DtO), teach-in of switching output, Invertible switching output, teach-in of analogue output, Invertible analogue output, Multifunctional input: laser off external teach-in deactivated, switch-off display, lock user interface

The current setup of these sensors in the *AtlasCar2* incorporates a limit on the measuring range (between 200mm and 700mm) since for the use case it is adequate, additionally they are configured with 2.5ms response time which corresponds with a 400Hz measuring rate [15].

2.1.3 Novatel SPAN-IGM-A1

The Novatel SPAN-IGN-A1 (Figure 2.4(a)) is a device based on Synchronous Position, Attitude and Navigation (SPAN) technology that brings together two different but complementary technologies: Global Navigation Satellite System (GNSS) positioning and inertial navigation.

(a) Novatel SPAN-IGN-A1 Receiver. (b) Novatel GPS-702-GG Antenna.

Figure 2.4: Hardware used for global positioning in *AtlasCar2*.

Despite being able to act as a stand-alone Inertial Measurement Unit (IMU) sensor, its real strength comes when coupled with a Global Positioning System (GPS) antenna such as the GPS-702-GG in Figure 2.4(b), also implemented in the *AtlasCar2*. This system combines the absolute accuracy of GNSS positioning and the stability of IMU gyroscope and accelerometer measurements to provide an excellent 3D navigation solution that is stable and continuously available, even through periods when satellite signals are blocked [18].

The current setup in the *AtlasCar2* includes the use of both of these sensors configured to use GPS signal received through the antenna at 20Hz, the signal then goes to the SPAN receiver to be processed along with IMU data gathered at 200Hz, and the global localization and instant pose results, along with other relevant data, are outputted over RS232 communication. Table 2.3 details more specific information about the Novatel SPAN-IGM-A1.

Table 2.3: Specifications for the Novatel SPAN-IGM-A1 Receiver [18].

SPAN System Performance		IMU Performance	
NovAtel CORRECT™		Gyroscope	
SBAS	60 cm	Input range	±450 deg/sec
DGPS	40 cm	Rate bias stability	6 deg/h
RTK	1 cm + 1 ppm	Angular random walk	0.3 deg/$\sqrt{hr}$
GNSS position	20 Hz	Accelerometer	
IMU measurment	200 Hz	Range	±18 g
INS solution	Up to 200 Hz	Bias stability	0.1 mg
Time Accuracy	20 ns RMS	Velocity random walk	0.029 m/s/$\sqrt{hr}$
Max Velocity	515 m/s		

2.2 Sensor Placement on *AtlasCar2*

After presenting the sensors that are to be used in this project it is important to show their placement in *AtlasCar2*, to serve as a visual reference to the work that follows.

Since the *AtlasCar2* project is still a prototype for a AD applications, its sensors are placed outside the vehicle's bodywork and mounted in supports built by former LAR students, Figure 2.5 shows the sensors and their respective placement, with Figure 2.6 detailing the placement of the four SICK DT20 Hi sensors used in the inclination module.

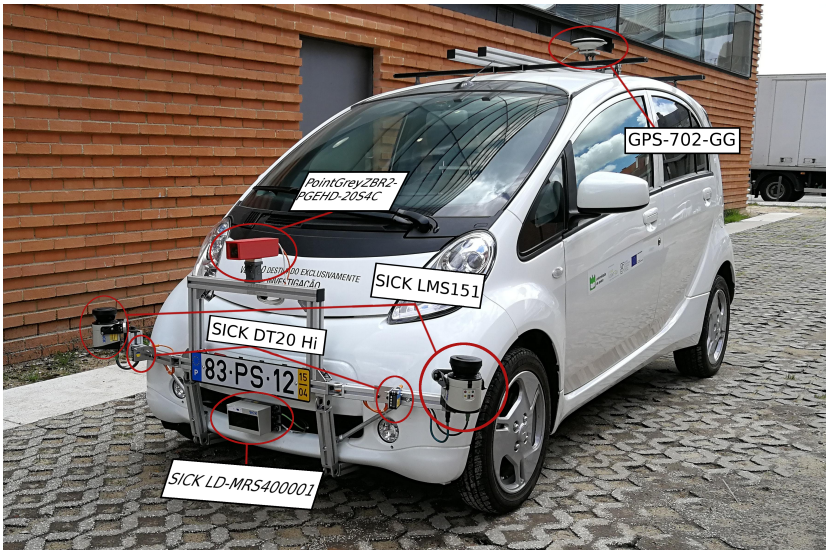


Figure 2.5: Sensor placement *AtlasCar2*.

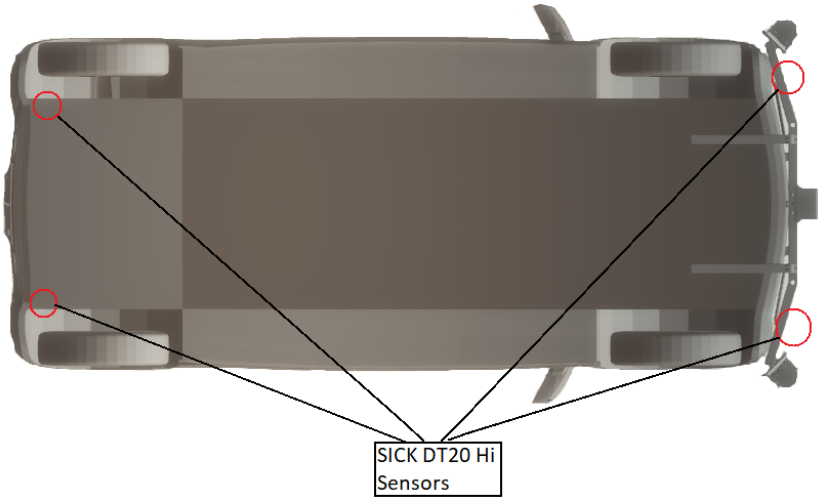


Figure 2.6: Placement of the SICK DT20 Hi sensors.

For the work to be developed in this dissertation the main focus will be the SICK LD-MRS400001 3D laser scanner detailed in 2.1.1, but the placement of the other sensors is also important given that an restructuring of the reference frame tree will be implemented.

2.3 Software

2.3.1 ROS

"The ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms" [19].

ROS was created in 2007 as part of the STanford Artificial Intelligence Robot (STAIR) developed at Stanford University [20]; this library was later refined and generalized, resulting in a general multi-lingual tools-based, framework free and Open Source [21] that as been in development since then.

It is an environment specially designed to help build large scale robotics projects, by focussing on modularity of different systems and having tools to interconnect them. These modules are called *nodes* in ROS nomenclature, and they communicate with each other using predefined data structures called *messages*, published and received through the *topics* that connect the *nodes* of the program. Figure 2.7 represents an example of the architecture for program developed in ROS.

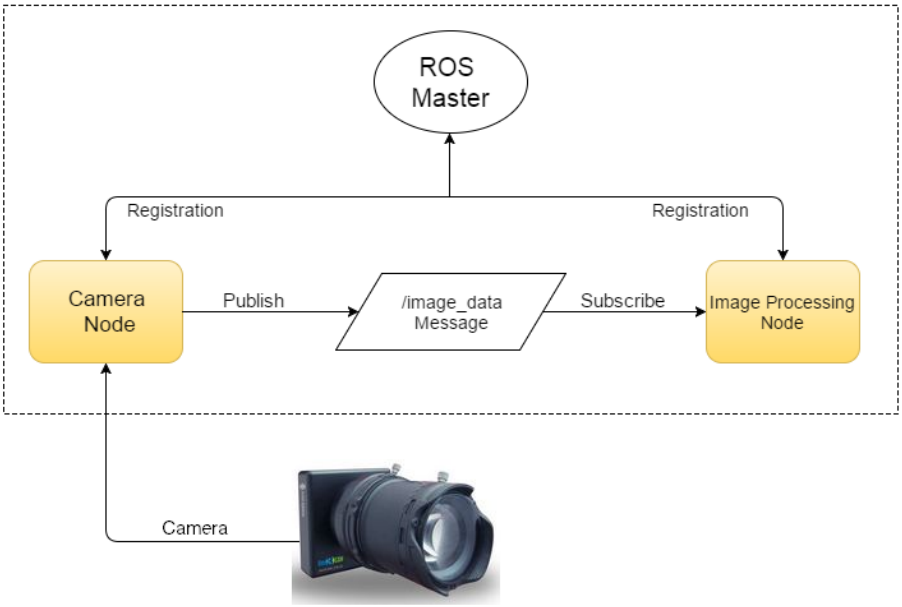


Figure 2.7: Example of a basic ROS framework with two nodes exchanging messages under the supervision of the ROS Master [22].

This great advantage allows for different groups of people to independently work on several different systems and functionalities of the robot, that once completed can be easily made to work together. Additionally because it is so generally used it as a very large selection of available and well documented libraries to interface with hardware used in robotics applications, making easy to incorporate new sensors to a project.

The *AtlasCar2* project is currently developed in using the ROS platform, so in order for the developed solution to be easily integrated in the project, all the code that will be developed in the context of this dissertation will also follow the ROS framework, and make use of its libraries.

2.3.2 PCL

In order for a robot to work in an unknown and ever changing environment it needs to be able to perceive the world around it. This problem brought the rise of range sensor technologies like Microsoft's Kinect, and LIDAR that gives high quality 3D representations of the world in the form of point clouds. As a result mechanisms had to be developed so that we can efficiently handle all that 3D data; for this reason, in 2011, the Point Cloud Library (PCL) was born.

The PCL is a large scale open project [23] for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms including, filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract key-points and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them [24].

PCL is split into a series of smaller code libraries, that can be compiled and used separately. This modularity is important for distributing PCL on platforms with reduced computational or size constraints. Another way to think about PCL is as a graph of code libraries (Figure 2.8) [24].

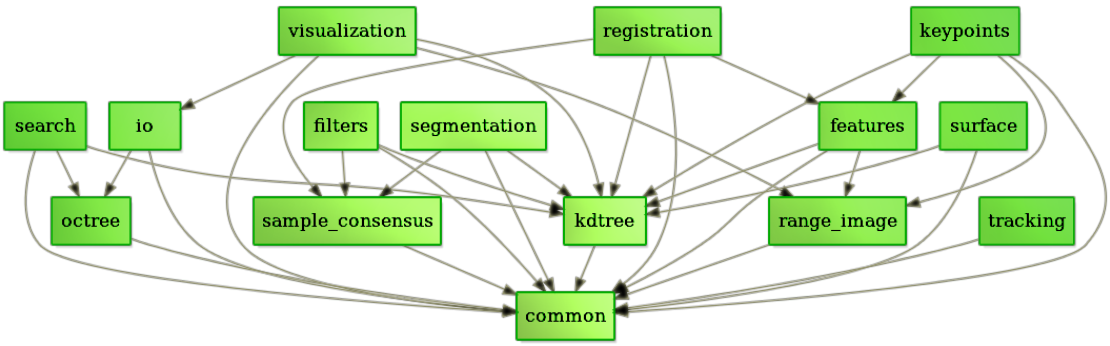


Figure 2.8: Graph of the libraries that constitute the *Common* module [24].

For this dissertation the most relevant modules to be used, are the:

- *Common* module: which serves as the base for all other PCL modules, as it contains data structures, such as PointCloud class, conversion methods, and general functions that are used by most of the other modules.
- *Filters* module: that contains outlier and noise removal mechanisms for 3D point cloud data filtering applications, important tools in the analysis of point cloud data.

2.4 Summary

In this chapter the tools which will serve as the base for this dissertation where presented. The Novatel SPAN will provide the global localization and instant pose of the vehicle, the SICK LD-MRS laser scanner will be responsible to gather 3D data about the environment in front of the vehicle, and that data in the form of a point cloud will

be processed and analysed using tools present in the PCL, with ROS being the base architecture on top of which all these modules will be implemented and interconnected.

Chapter 3

Road Reconstruction Module

In mobile robot applications, as is the case with the *AtlasCar 2* Autonomous Driving (AD) vehicle, it is essential to know the instant position and orientation of the robot (called *pose* in robotics applications), both relative to the ground plane and to the world around it. Having this information is particularly important in applications where the robot is moving and needs to reconstruct the world around it, given that the perceived environment is constantly changing. Correctly defining reference frames is important because the sensors perform their measurements relative to themselves, and to reconstruct the environment this data needs to be represented relative to a fixed world frame, so that during vehicle movement the readings can represent the features of the perceived world. This is achieved by creating reference frames for each sensor and each "joint" of vehicle, and defining the correct transformations between them, therefore between the sensors and the fixed world frame. Due to the movement of the vehicle these transformations have to be constantly updated to keep track of the sensor positions and orientation.

The transformations between the reference frames can be divided into two stages; one local, referring to the position and orientation of the sensors relative to the fixed vehicle frame, and one global, referring to the pose of that fixed vehicle frame relative to the world. Given the early stages of the *AtlasCar 2* project, such a mechanisms for vehicle localization are not yet implemented. As such, in this chapter a solution will be presented that will allow to correctly determine the instant pose of the vehicle, both from a local and global perspective, as well as the methodology used to transform the data gathered by the sensor into a accumulated point cloud representing the world in which the vehicle navigates.

3.1 Vehicle Orientation System

3.1.1 Inclinometry Module

As a vehicle navigates through a road it is bound to accelerate, decelerate or go through some turns; these events, to name a few, cause the car to change its orientation, varying its *pitch* and *roll* components (as they are referred in robotics figure 3.1), relative to the road. This is a problem, because the onboard sensors are rigidly fixed to the bodywork of the vehicle and consequentially will move with it, changing their respective orientations and measuring different features.

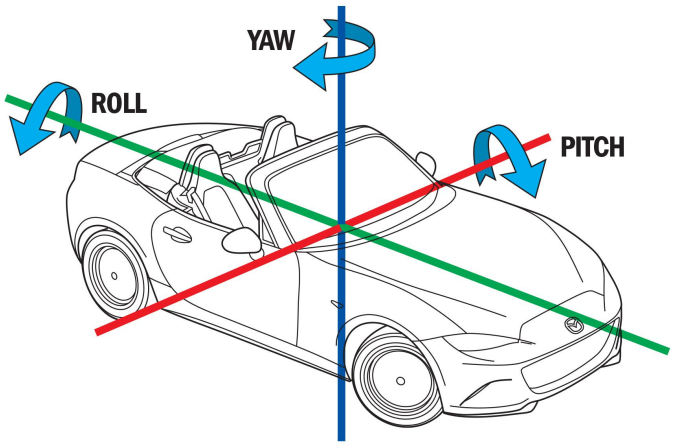
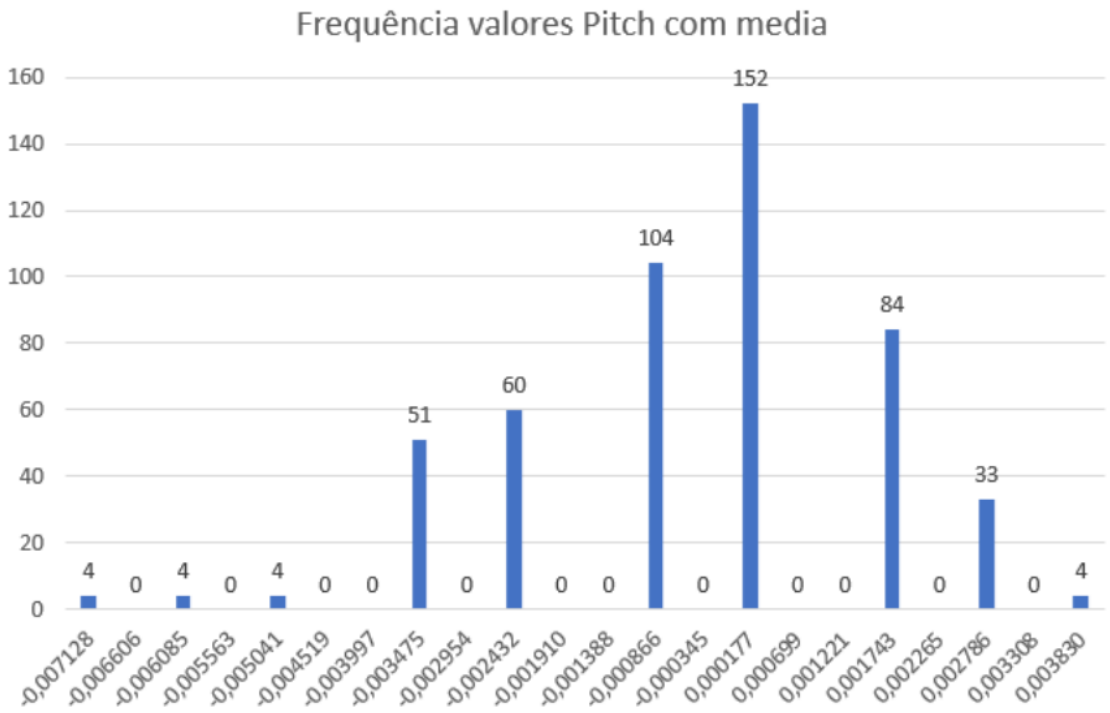


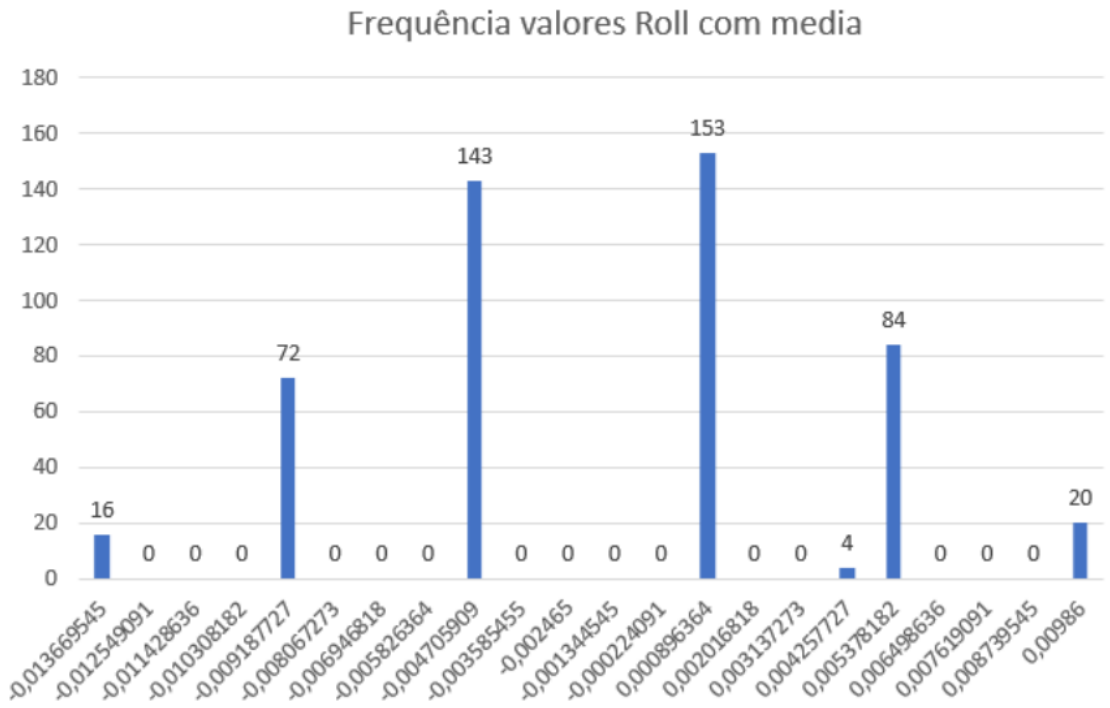
Figure 3.1: *Pitch*, *roll* and *yaw* rotation frames.

To solve this problem the orientation module was developed by *A. Silva* [15] as an improvement on the solution proposed by *P. Salvado* in [14] and *G. Carpinteiro* in [25] to measure the orientation of the vehicle relative to the plane in which the robot navigates. The working principle of this approach is based on having a distance sensor in each of the four corners of the vehicle (as shown in figure 2.5) and directly measuring the distance between the body-work of the car and the ground, then use that data coupled with the information about the fixed distance between each sensor to calculate the *roll* and *pitch* of the vehicle, according to equation 1.1 described in section 1.5.3.

The system currently implemented in *AtlasCar 2* makes use of four high precision SICK DT20 Hi optoelectronic sensors, and a *Arduino Nano* as the processing unit, that receives the data, calculates *roll* and *pitch* and performs a mean filter to reduce the error range. This process makes for a robust solution, as shown by the histogram of the angles, figures 3.2a and 3.2b, calculated from a sample of 500 readings, which show an error range of 0.011 48° for *pitch* and 0.024 65° for *roll* [15].



(a) Histogram of the measured *pitch* values.



(b) Histogram of the measured *roll* values.

Figure 3.2: Range for the calculated angles in a static position [15].

3.1.2 Conversion to ROS node

This module was developed as a stand-alone solution, and the calculated values for *pitch* and *roll* where only displayed in an onboard LCD. So, in order to make use of this information and migrate it to the ROS environment, changes had to be made to the program running in the micro-controller so that it could act as a ROS node.

To accomplish this the *Rosserial_Arduino* package available from the ROS repositories [26], was used. The *Arduino* now needs to be connected to the central processing unit on the *AtlasCar 2* that runs the ROS *master*, and sends the orientation data through *RS232* communication in the form of a *Float32Array* message to the *DadosInclin* topic. This message contains three values, referring to *pitch*, *roll* of the vehicle and the average height measured by the four sensors. The integration with the ROS framework allows the orientation data to be available to other nodes for them to use it in their respective processes.

Additionally a functionality was added for the automatic calibration of the sensors. The node also subscribes to a topic that publishes information about the *mission start* so that when it receives this information the program calibrates the sensors based on hard coded values obtained from a scenario where the vehicle was empty of passengers on a plain surface.

3.2 Vehicle Positioning System

The problem of measuring the instantaneous pose of the robot relative to the the world in which navigates is fundamental in robotics, and as such there are many solutions that allow to determine this status.

In the field of AD applications one of the preferred methods for determining de vehicle position is through the use of GPS, given the scale of the map in which the vehicle can navigate, GPS proves to be a reliable and accurate solution to this problem.

GPS technology is comprised of a constellation of satellites and a remote receiver that uses range measurements to the satellites and triangulation techniques to compute the position of the receiver's antenna [27]. Redundant phase observations from 5 or more satellites provide the information to resolve the ambiguities, thus translating each satellite's estimated phase cycles into precise range measurements. High precision satellite-to-receiver range measurements allow the computation of the baseline (inter-station) vector between the receivers and hence the position of the remote receiver to decimetre or better accuracy [27].

Despite the fact that GPS can provide very accurate measurements in ideal conditions, there are situations that cause the system to receive signal from less than four satellites, caused by entering a tunnel, or certain terrain topologies that obscure the GPS signal, this provokes large errors or even an absence in position measurement.

As a solution to this problem a new system architecture called *Tightly-Coupled Architecture* [28] was developed, which allows to combine GPS and IMU data to obtain very precise position measurements. This solution has since become a part of the industry standard for vehicle localization in AV applications.

A tightly-coupled algorithm uses a centralized Kalman filter that integrates the estimated pseudoranges (ρ_{GPS}) and Doppler shift ($freq_{Doppler,GPSfreq}$) from the GNSS receiver and the information of position, velocity and attitude coming from the mech-

anization equations of the inertial sensors [28]. Figure 3.3 illustrates a simple block diagram of a tightly-coupled architecture.

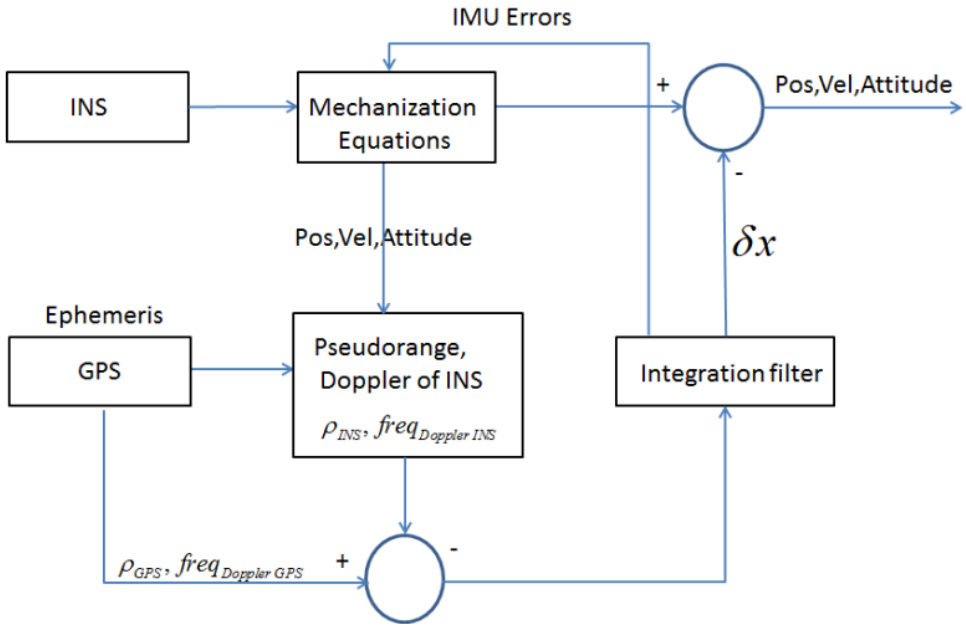


Figure 3.3: Block diagram of a common tightly-coupled architecture [28].

The *AtlasCar 2* currently is equiped with a Novatel GPS-702-GG Antenna, and Novatel SPAN-IGM-A1 Receiver, that natively implements this technology [18], and so it will serve as the hardware to measure the vehicle position and orientation relatively to the world providing continuous six degrees of freedom information about the state of the car.

3.3 ATLASCAR2 Frame Tree Restructuring

In order to make use of the orientation and localization information gathered by the methods detailed in sections 3.1 and 3.2, the robot frames have to be correctly implemented. The inclusion of these new methods required an overhaul of the *AtlasCar 2* frame tree created by *D. Correia* in his work [29].

Firstly the global position data provided by the SPAN is converted from geographic to Cartesian coordinates through the *gps_conv* node provided in the *gps_common* ROS package and that data in then published to the *odom* topic. However this position is relative to the earth frame, so the *frame_tf_broadcaster* ROS node was created to offset the position by the value of the first measurement, effectively making so that the start of the system corresponds to the zero position on this new fixed frame, called *map*. This node also creates a new frame, *ground*, that corresponds to a point on the ground plane directly below the vehicle's GPS antenna, the position of this frame is defined relative to the *map* fixed frame where its position is updated based on the compensated GPS data and its orientation (only the *yaw* component, since the road is locally considered to be flat), is obtained through the IMU measurements. Figure 3.4 depicts the path, blue line, of the moving vehicle on the road obtained with the method described above.

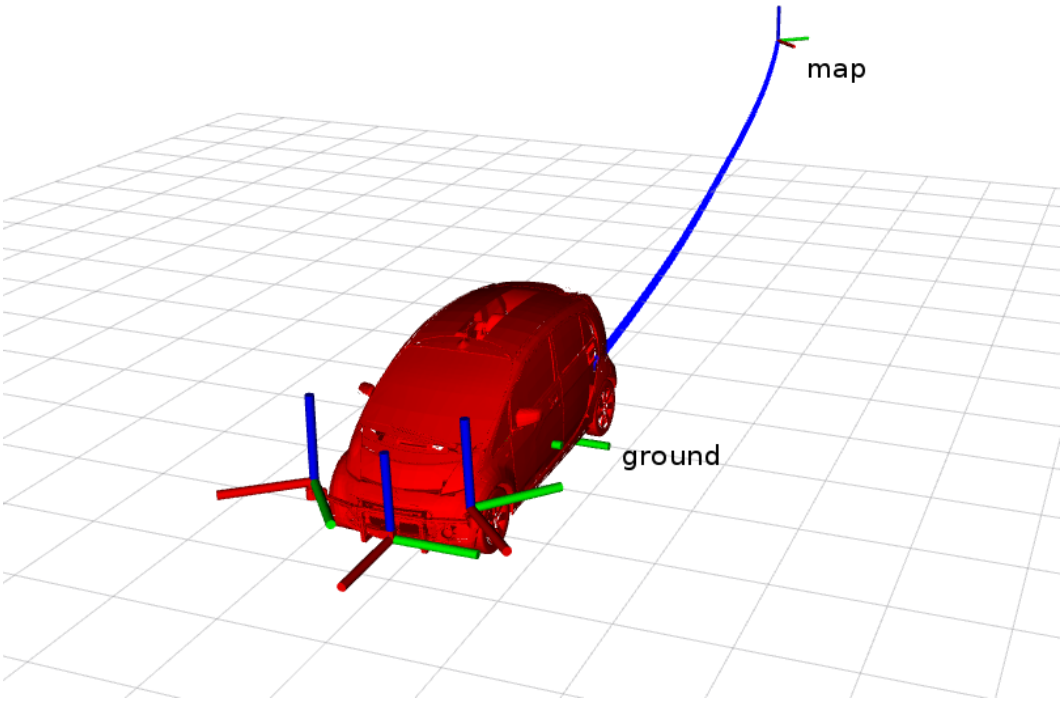


Figure 3.4: Position of the *AtlasCar 2* relative to the *map* frame during motion.

For the representation of the vehicle orientation relative to the road plane an additional frame was created, *car_center*. This frame is displaced by static transformation in x from the *ground* frame, in order to represent the centre of the vehicle, moreover, the height of this frame is defined also relative to the *ground* frame and is determined as the average height measured by the SICK DT20 Hi sensors in the orientation module. Additionally this frame also possesses rotation components relative to the *ground* frame, defined by the *pitch* and *roll* angles also provided by the orientation module.

Lastly the sensor frames need to be defined relative to this new rotating frame. The node that builds the transformation tree was developed so that it could make use of the calibration files generated by the *multisensor calibration* package developed by *D. Silva* in [30], and for this, a new frame was created. This new frame, *moving_axis*, served to bridge the transformation tree referring to the placement and orientation of the sensors provided by the calibration files and the transformation tree referring to the position and orientation of the *AtlasCar2*.

The *moving_axis* frame will allow to maintain the architecture in which the sensor frames are defined using calibration files, relative to one main frame, *lms151_E* in this case, and still reflect the influence of the vehicle rotation during movement. Figure 3.5 includes a representation of these frames in the 3D model of the vehicle.

Figures 3.6 and 3.7 illustrate the interaction between the frames of the *AtlasCar2*, and the resulting projection of the laser scans, resulting from the introduction of pitch and roll rotations, respectively, to the vehicle.

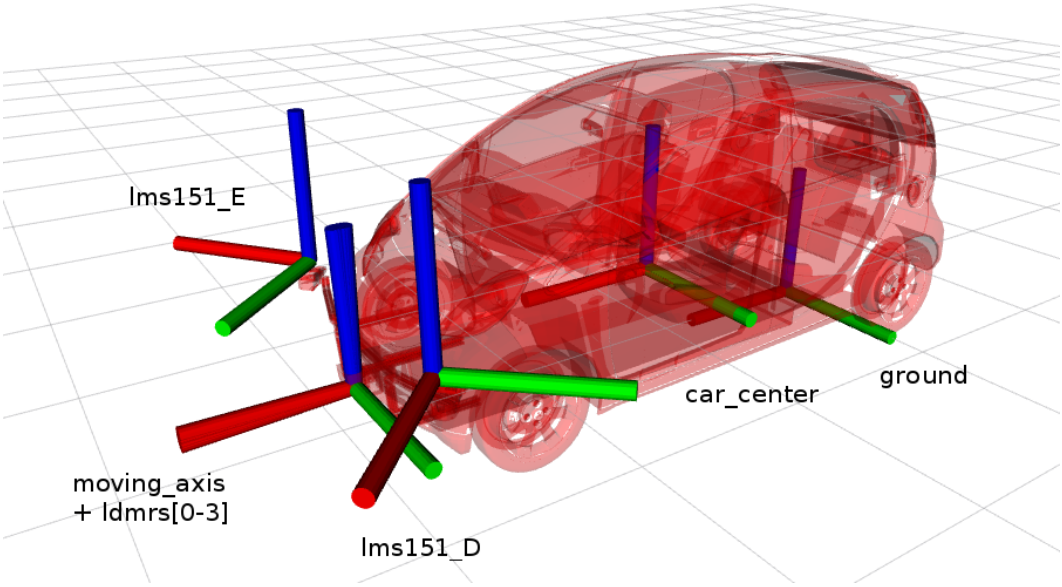


Figure 3.5: *AtlasCar 2* frame placement with STL model.

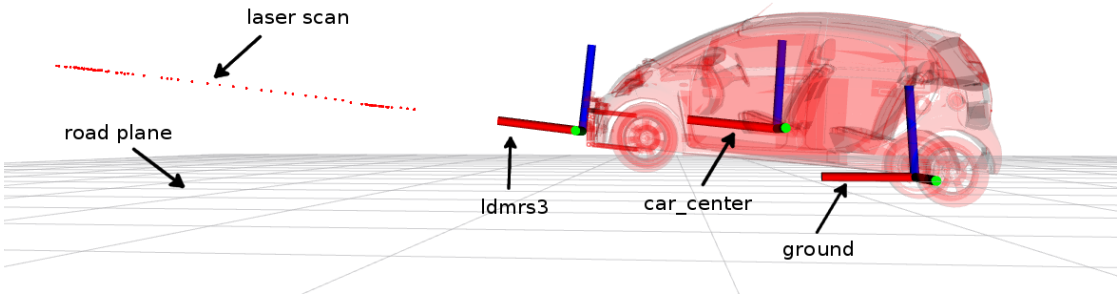


Figure 3.6: Side view of the influence of a simulated vehicle *pitch* in the transformation tree and in the laser scan readings.

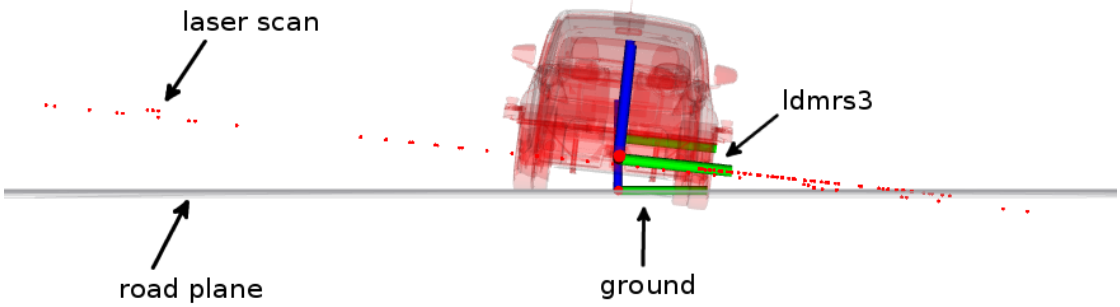


Figure 3.7: Frontal view of the influence of a simulated vehicle *roll* in the transformation tree and in the laser scan readings.

Figure 3.8 illustrates the transformation tree structure for the updated frames for the *AtlasCar 2*.

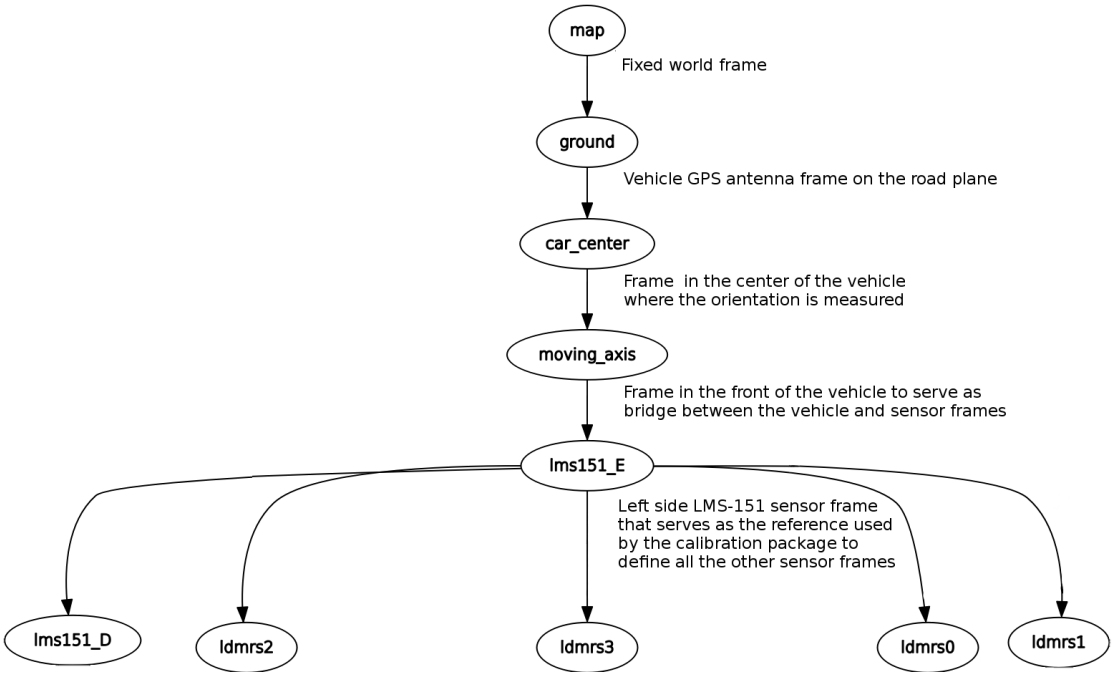


Figure 3.8: Transformation tree for *AtlasCar 2*.

3.4 Road Reconstruction

The next step was to reconstruct the environment surrounding the car as it navigates. For this a node was developed, *road_reconstruction_node* as part of the *road_reconstruction* package, with the objective of gathering the laser scan readings from the SICK LD-MRS sensor, and accumulating them in order to greatly increase the 3D information, forming a point cloud representative of the perceived environment.

Figure 3.9 represents the distances with which the laser scan plane intersect with the ground plane for a given sensor placement.

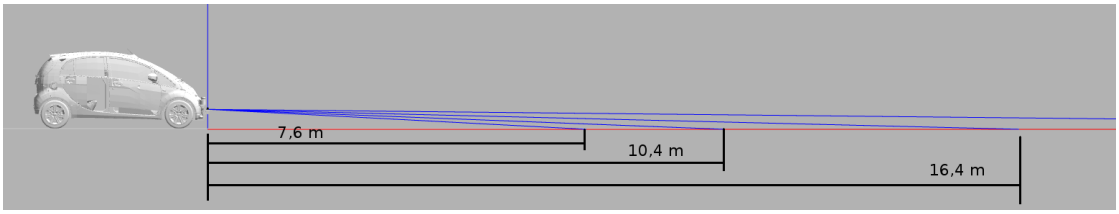


Figure 3.9: Intersection of the scan planes with the ground plane.

As observed in the image above given the limited amount of scan planes (four), the resulting projection is, in ideal conditions, a set of points arranged in four lines at the distances indicated in the image. Figure 3.10 represents this projection on a real section

of road. This information proves to be insufficient to consistently extract features about the road and the environment. To counteract this effect there is a need to increase the number of data points used to represent the road.

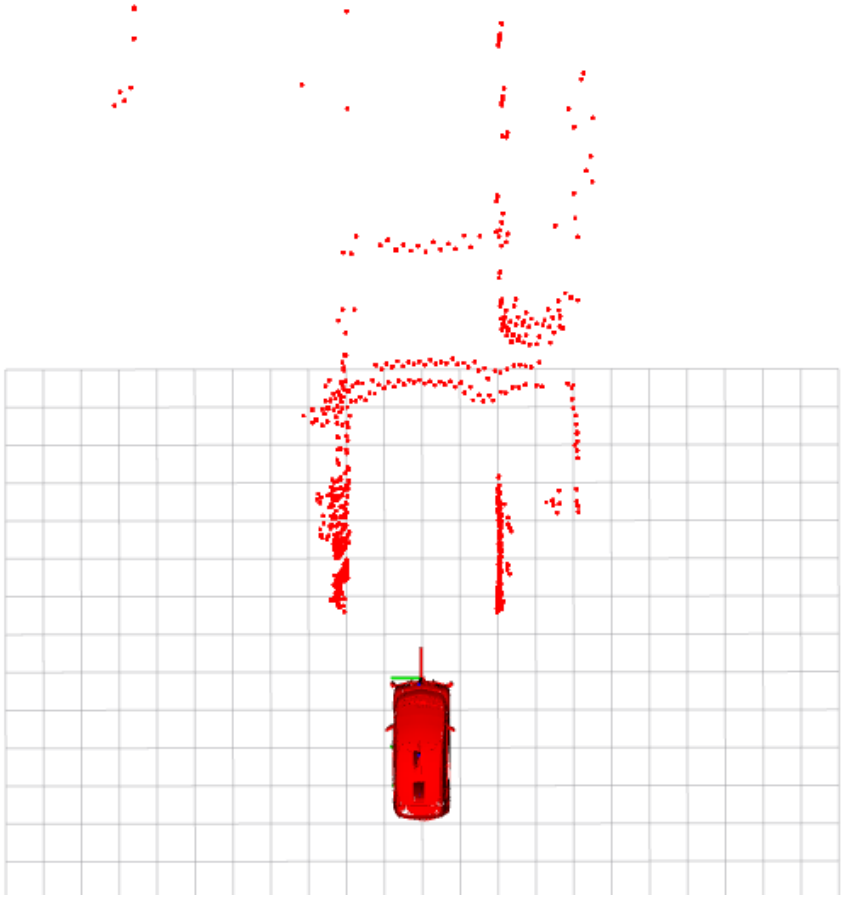


Figure 3.10: Top view of the four laser scan that intersect the road plane in a real world situation.

This is done by taking advantage of the movement of the vehicle. Since the sensor is displaced during the measuring process because of the vehicle movement, it will register different sections of road each time it moves along the vehicle's path. As an example, the sensor has a measuring frequency of 50 Hz (table 2.1), if the car is moving at $50km/h$ the sensor performs a measurement, on all four scanning planes, every $0,29m$. By accumulating these successive measurements it is possible to form a point cloud that represents the road and its features with good accuracy and density to allow to extract the road boundaries.

To accumulate the readings gathered from the LIDAR sensor the *laser_assembler* ROS package was used [31]. This package provides a service which takes successive data from a single laser scan and assembles it, relative to a input frame, in a rolling buffer of a predefined size (figure 3.11), allowing to control the size of the road reconstruction point cloud. When a new cloud is available the node poses a request to the service which in turn returns the point cloud buffer for the respective laser scan. This is done for each of

four laser scans, providing the *laser_assembler* service with the correct frame for each scan (from *ldmrs[0-3]*).

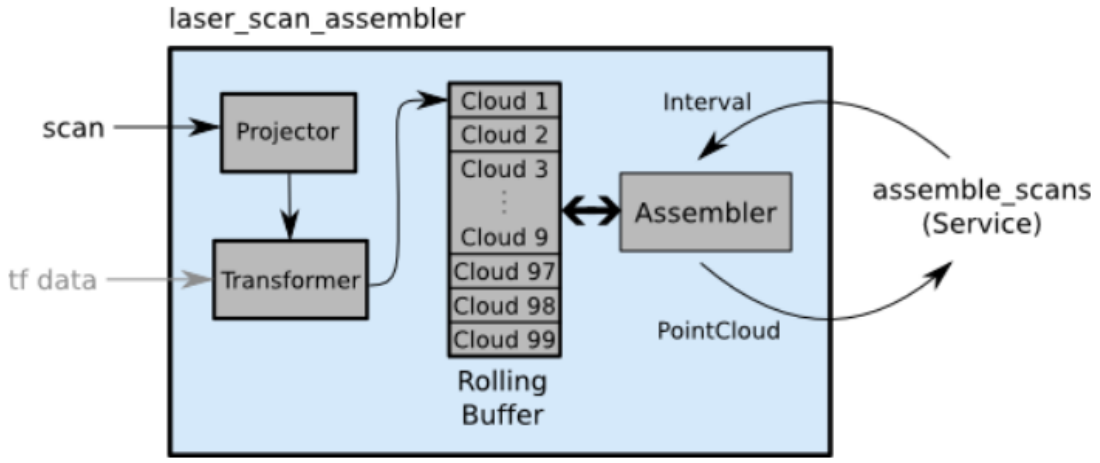


Figure 3.11: Diagram of the working principle for the *laser_assembler* service.

This method is very efficient and there is almost no delay between the acquisition of the data and its assembly to the point cloud, making it a good solution even for a fast moving vehicle, as can be seen in figure 3.12, that represents the laser scans in red and the previous accumulated points in white.

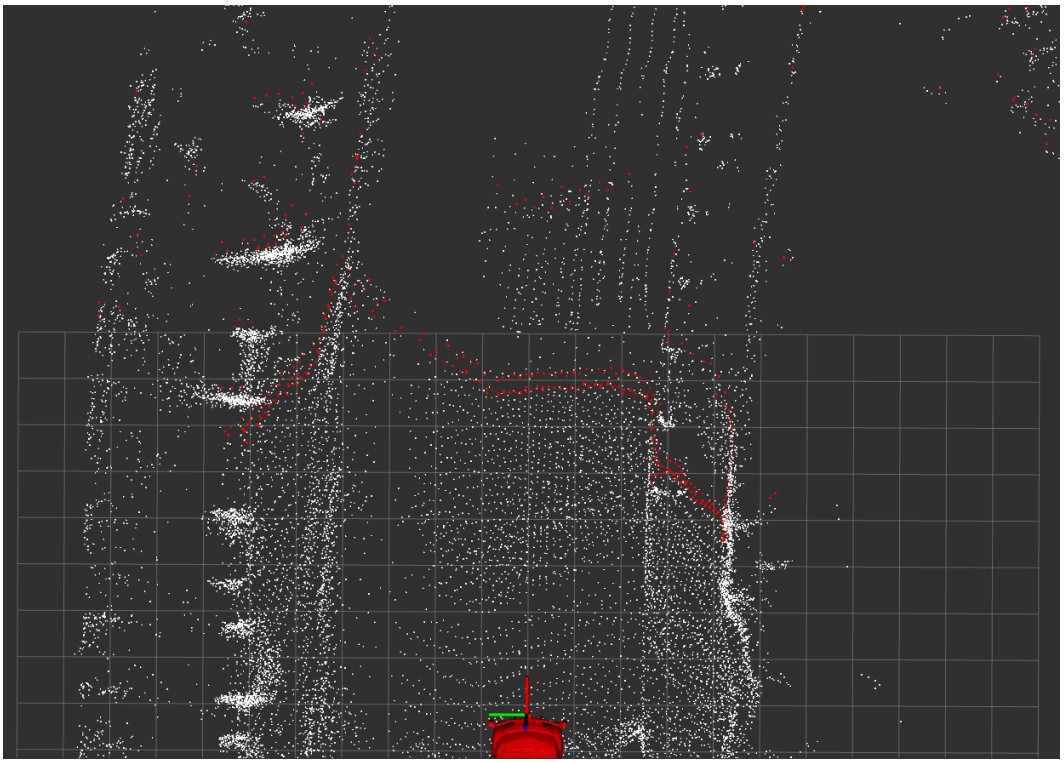


Figure 3.12: Representation of the laser scans (in red) and accumulation point cloud (in white).

This node is also responsible for taking the assembled point clouds corresponding to each of the scan planes and merging them into a single point cloud ready to be processed. Figure 3.13 illustrates a reconstruction of the road accomplished through the described method, where the point cloud is coloured based on height relative to the ground. For the purpose of this application, since the road side-walk is most relevant feature that will need to be identified and to reduce the size of the resulting point cloud, only the two bottom most scan planes of the sensor where used for the reconstruction of the road presented in the figure.

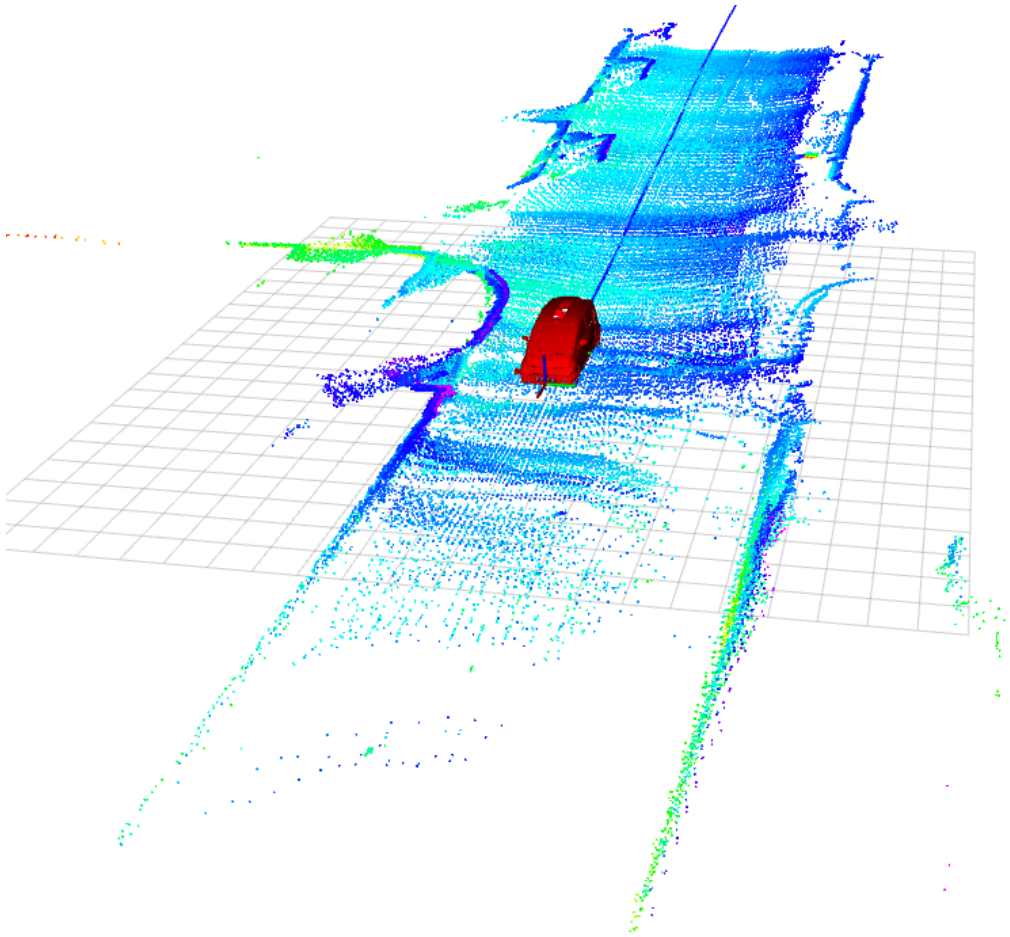


Figure 3.13: Example of the reconstruction of a straight section of road using only the two lowest laser scans of the LIDAR sensor.

The reconstructed section of road presented in figure 3.13 proves to be a good representation of the environment, with rich information about the road and curb profiles. This reconstruction also presents some interesting properties regarding the representation of the road boundaries, which can easily be identified in the image due to the higher concentration of points characteristic of these regions. Additionally, from the this particular image we can observe that despite only using the two lowest laser scans, the reconstruction is performed to a distance of approximately $20m$, in this case, from the front of the vehicle, suggesting that it is viable to perform the road boundary extraction from this point cloud to use in navigation algorithms.

Chapter 4

Methodology for Road Boundaries Extraction

This chapter describes the methodology developed to extract the road limits from the point cloud generated by the *Road Reconstruction* module. This process is explored in depth by performing an analysis of the laser projections of the LIDAR on the road and curb, and by translating these projections through a mathematical model that allows to evaluate the levels of accumulation during the movement of the vehicle. Finally, the filtering process used to obtain the road limits is explained, along with some considerations regarding the performance of the method.

4.1 Curb Extraction Through Accumulated Point Cloud Density

When analysing the point cloud generated by the road reconstruction module, detailed in Section 3.4, and comparing it to the point cloud obtained by the traditionally used top mounted LIDAR sensor [32, 33, 34] (figure 4.1), there is a clear difference regarding the section of the world that is represented in the two data sets. This difference in how the world is perceived by the two approaches, is caused by the combination of the type of LIDAR sensor that was used and its placement in the vehicle.

Traditional applications use a high density (64 beam), rotating LIDAR, placed on the very top of the vehicle, which provides a 360° map of the world and all the objects in it (cars, pedestrians, trees, etc.). This is done so that this cloud can be used for multiple AD applications, like detecting other cars, pedestrians, road limits. In contrast the LIDAR sensor used in this work and its placement near the ground plane does not provide much information about all the objects around the vehicle, but instead, focuses only on the features closer to the road, such as side-walks, barriers or small holes in the ground, that are important to the problem of determining the road limits.

Given the difference in the nature of the point cloud obtained in this work, the main algorithms used to detect the road boundaries that rely on the high density top-down laser scan data could not be directly implemented. Hence, a new method for road boundary detection had to be developed for this specific sensor arrangement and data, which must take advantage of the unique perspective provided by the configuration of the sensor and the accumulation of the scanned data.

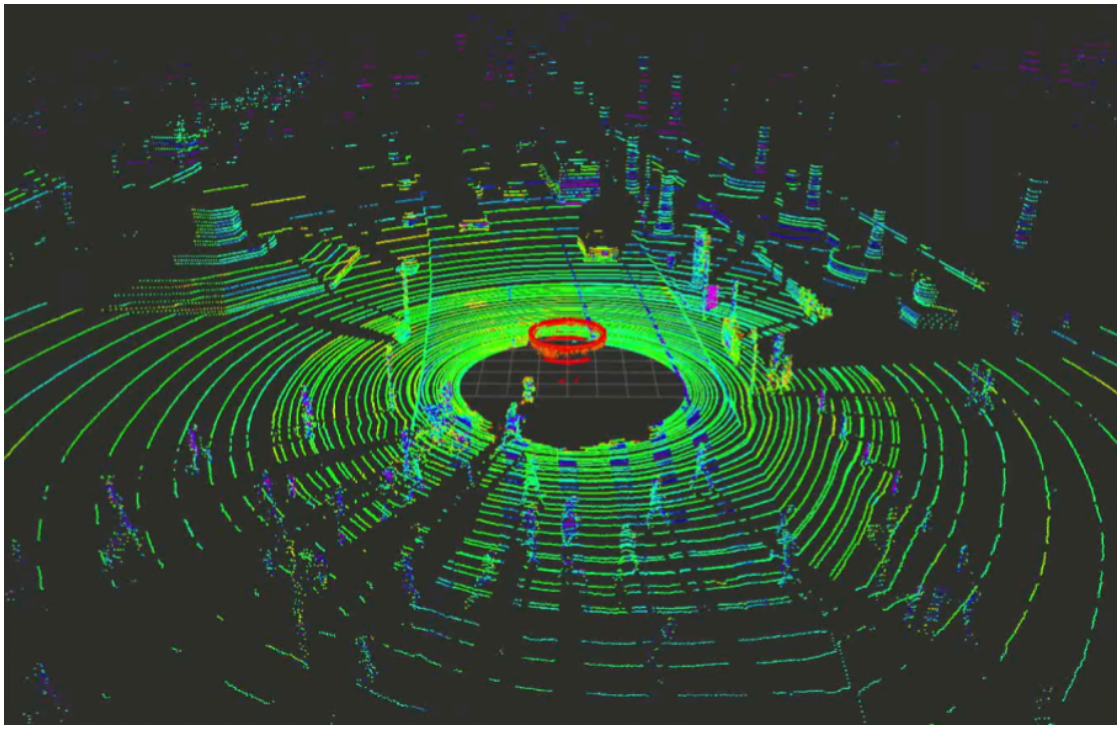


Figure 4.1: Resulting point cloud of traditional rotating LIDAR configurations [35].

Through the analysis of the projection from the laser beams in a side-walk (Figure 4.2), it was observed that there was a big displacement (along the direction of the road) between the points where the scan hits the top of the side-walk (1) and the road plane (3). Additionally, the sections of the laser beam that hit the vertical face of the curb (2), and form the diagonal between the aforementioned points, produce a long line at the edge of the road that also follows the direction of the curb.

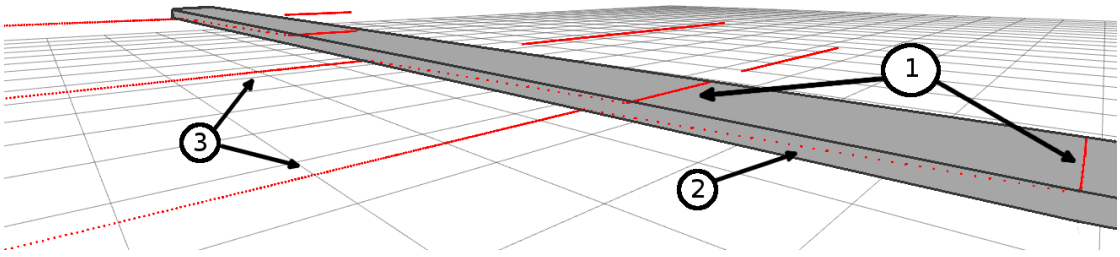
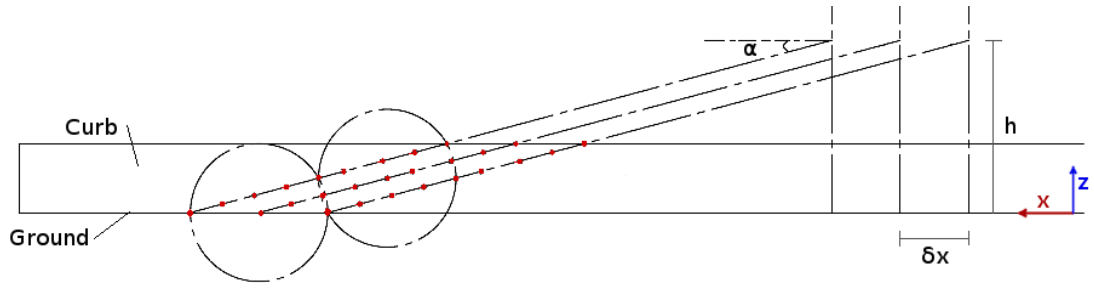


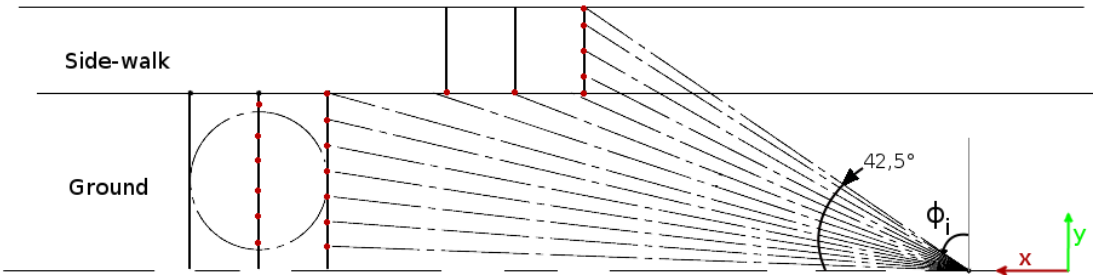
Figure 4.2: Projection of the LIDAR scan planes in a standard side-walk.

When the accumulation of the successive scans is taken into account these diagonal lines created by the vertical projections form a plane of closely adjacent points that corresponds to the curb surface. Figure 4.3 represents a simplification of the laser projection from a side (4.3a) and a top (4.3b) views.

By analysing these projections we notice that, despite the points in the diagonal projection being more spaced out than the ones in the horizontal planes (ground and side-walk), the accumulation produces a high concentration of points in the vertical plane



(a) Simplified diagram of the laser projections of a single scanning plane during accumulation of three sucessive readings - Side view.



(b) Simplified diagram of the laser projections of a single scanning plane during accumulation of three sucessive readings - Top view

Figure 4.3: Difference in the local concentration of points in the vertical and horizontal planes, shows the density to be higher in vertical planes.

as demonstrated by the circles in Figure 4.3. This difference between the concentration of points in the vertical and horizontal planes decreases with α, in which lesser values produce higher concentrations on the vertical plane, while keeping the concentration of the points in the horizontal planes constant. Decreasing the angle would mean that the projections would be too far away from the vehicle, but since the sensor used in *AtlasCar2* is placed very close to the ground this problem is rectified. One other factor that influences the local point density is the velocity of the vehicle, as it influences the interval between successive readings of the sensor (δx), with greater velocities producing higher intervals, and therefore reducing the overall density of the measured section of road.

The behaviour, resulting from the combination of sensor placement, vehicle velocity and laser scan accumulations, provides a way to differentiate between the ground and the curb planes. This is done by analysing the local concentration of points in the road reconstruction point cloud, that we will call Accumulated Point Cloud Density (APCD), in which the lower density regions correspond to horizontal planes and higher density ones to vertical planes. This distinction will serve as the methodology with which the road boundary detection algorithm will be based on.

4.2 Mathematical Model for Laser Scan Projections in 3D Space

In order to apply this methodology to identify the road boundaries, it is required to have a deeper understanding of the system in place, and more specifically its interaction with the different road and curb configurations present in real world situations.

With this goal in mind a mechanism had to be developed in order to replicate the system in place and evaluate, in a controlled environment, the influence of the many variables that can have an effect on the laser projections. This system must have a model that allows to quantify the concentration of points in a desired region to serve as a quantifiable value for the influence of the parameter to be studied. This study will provide some theoretical guide lines to be used in the boundary detection algorithm of the real world system.

4.2.1 Laser Scan Simulator

The solution was to build a simulator with the intent of evaluating the influence of the variables belonging to the two main aspects of this approach.

- Influence in APCD due to the placement of the sensor in *AtlasCar2* and its movement along the world.
- Influence in APCD due to the configuration of the curb relative to *AtlasCar2* and the ground plane.

This interaction between the laser beams and the curb is what provides the APCD values and is what will allow to establish the guide-lines for the real-world case, so it is important for them to be accurately modelled.

For the first part referring to the placement of the sensor, the main variables to be studied are the height and tilt angle relative to the ground plane. For this the tri-dimensional geometric equations 4.1 that translate the laser beam behaviour were deduced. Figure 4.4 illustrates the simulated laser beams produced by these equations.

$$\forall X \in \mathbb{R} : \begin{cases} Y = \dfrac{X - \delta x}{tan(\phi_i)} \\ Z = h - (x - \delta x) \cdot tan(\alpha) \end{cases} \tag{4.1}$$

These equations reflect the path of the laser beams based on the sensor placement defined by the variables indicated in figures 4.3b and 4.3a, where:

- Height (h) - Distance in meters from the center of the sensor to the ground;
- Tilt (α) - Angle between the scanning plane and an horizontal plane parallel to the ground;
- Displacement (δx) - Distance between the sensor and the origin in the direction of movement, increasing this value with time, allows to simulate vehicle movement;

And its specifications:

- Beam angle (ϕ_i) - Angle formed by the combination of the sensor aperture (85° total) and the angular resolution (0.5°) for each beam i;
- Scanning planes - The value of the angle between scanning planes (0.8°) is added to the tilt of the sensor (α) in order to produce the four scanning planes characteristic of this sensor;

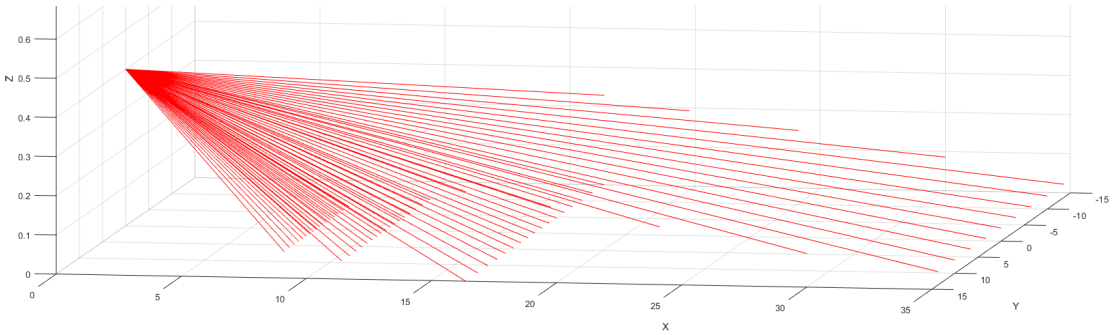


Figure 4.4: Simplified illustration of the laser beam paths in four scanning planes, with $h = 0.4m$, $\alpha = 0.6°$, $\phi = 5°$ and $\delta x = 0m$.

Another important feature that was implemented on the simulator is the capability to simulate vehicle movement. This functionality is important since the APCD methodology heavily relies on the accumulation of successive sensor readings. For this, the information about vehicle velocity (v_{car}), needs to be provided, and this information is then combined with the data acquisition frequency of the LIDAR sensor (f_{car}) (50 Hz), to calculate, according to equation 4.2, the distance between consecutive readings of the LIDAR sensor. In the simulator this is represented by the displacement variable δx. Figure 4.5 represents a side-view of the successive scans according to the calculated displacement.

$$\delta x = v_{car} / f_{sensor} \tag{4.2}$$

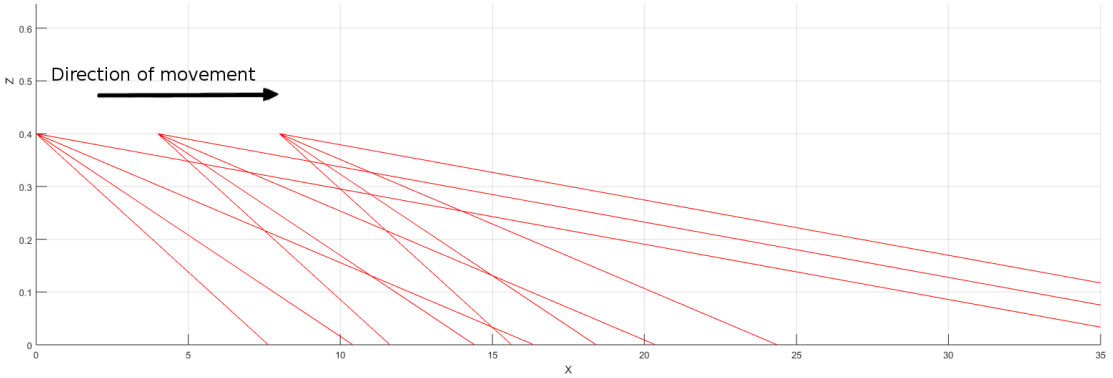


Figure 4.5: Simplified side-view illustration of the laser beams in four scanning planes during movement, with $h = 0.4m$, $\alpha = 0.6°$, $\phi = 5°$ and $\delta x = 4m$.

The second part of the simulator, that refers to the curb configuration, was implemented by creating a plane, defined by a point $p = (x_p, y_p, z_p)$ and a normal $\vec{n} = (a, b, c)$

according to the parametric equation 4.3, these two components are respectively responsible for the position and orientation of the road curb.

$$(x_p - X) \cdot a + (y_p - Y) \cdot b + (z_p - Z) \cdot c = 0 \tag{4.3}$$

4.2.2 Measuring Point Cloud Density

As presented in Section 4.1 the APCD method for determining the road limits is based on the difference in the concentration of laser scans in different regions of the accumulated point cloud. So the approach to measure the APCD in the simulated environment has to allow the user to adjust its parameters according to the placement and size of the region intended to be studied.

The first step was to implement "collision" between the laser beams and the plane that represents the curb geometry. For this, the system of equations 4.4 is solved for each beam (680 across all scanning planes), and the values of (X, Y, Z) corresponding to the intersection are kept. There is also the possibility to only keep the points under a certain height, by applying a $Z \leq h_{curb}$ restriction, to simulate the vertical size of the curb and only accumulate the intersection points belonging to it (Figure 4.6).

$$(X, Y, Z) = \begin{cases} Y = \dfrac{X - \delta x}{tan(\phi)} \\ Z = h - (x - \delta x) \cdot tan(\alpha) \\ (x_p - X) \cdot a + (y_p - Y) \cdot b + (z_p - Z) \cdot c = 0 \end{cases} \tag{4.4}$$

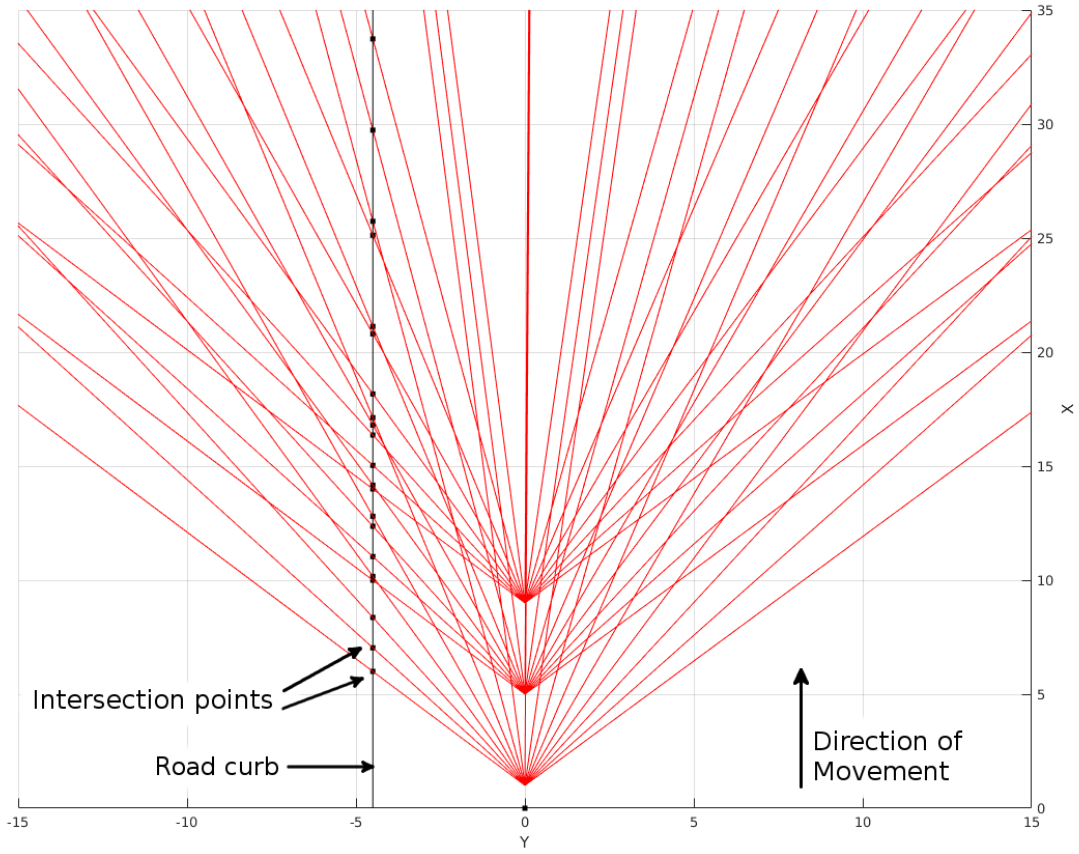
After having the values for the intersection between the laser beams and the plane, the region in which the concentration of points is going to be measured needs to be defined. For this purpose a sphere is created with its centre in the point p and radius r, and the region is formed by circumference resulting from the intersection between the created sphere and the plane.

The approach is then to quantify the points that intersect the plane and are contained within the circumference of the region. For this, the inequality 4.5 is verified using the (X,Y,Z) values from the intersection points gathered above. Naturally, the sphere will intersect with the curb plane resulting in a circle contained within the plane, and so only this circle is considered as the region.

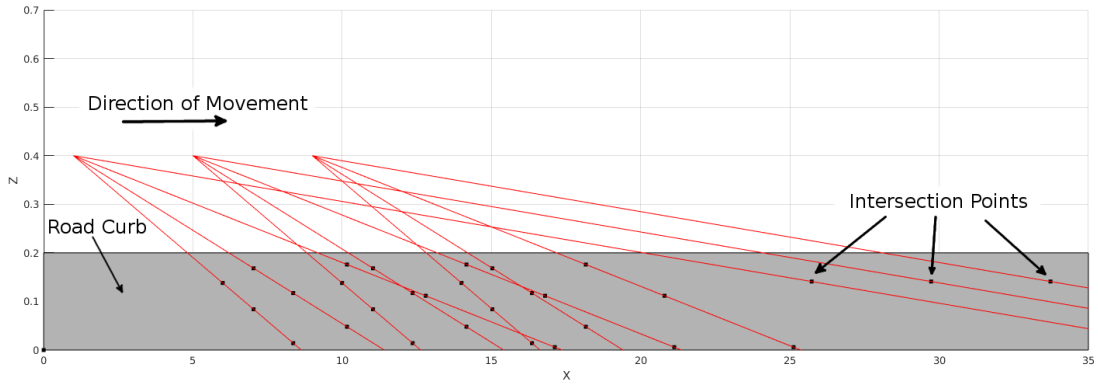
$$(X - x_p)^2 + (Y - y_p)^2 + (Z - z_p)^2 \leq r^2 \tag{4.5}$$

This procedure is repeated always maintaining the same position and radius of the regions for each of the laser beams in all scanning planes, and the values are accumulated for each displacement that represents the movement of the car, Figure 4.7 shows the results for this accumulation using a circle centred in $(14, -4.5, 0.25)$ with radius $0.5m$. When the simulation ends, the car having moved the intended distance, the APCD is calculated following Equation 4.6, where N_{total} refers to the total number of points contained within the region to be examined and A_{circle} refers to its area.

$$APCD = N_{total}/A_{circle} \tag{4.6}$$



(a) Intersection of the laser scans with the road curb during movement - Side view.



(b) Intersection of the laser scans with the road curb during movement - Top view

Figure 4.6: Illustration of the intersection between laser beams and curb plane with representation of the accumulated points (in black), for a vertical curb $h_{curb} = 0.2m$ and $h = 0.4m$, $\alpha = 0.6°$, $\phi = 5°$ and $\delta x = 4m$ sensor configuration.

As a result, we now have a mechanism to measure APCD in controlled conditions and a method to evaluate the influence of the several variables of the system.

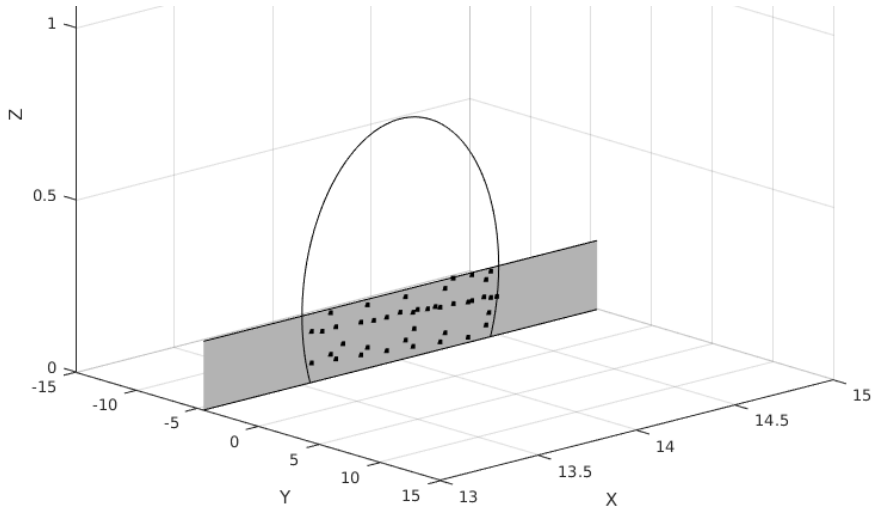


Figure 4.7: Illustration of the intersection points contained within a study region at the end of the vehicle movement. Considering a vertical curb, $h_{curb} = 0.2m$, $h_{sensor} = 0.4m$, $\alpha = 0.6°$, $\phi = 5°$ and $\delta x = 2m$ curb and sensor configuration.

4.2.3 Limitations of the Simulator

Despite having built this simulator with the capability to test many variables of the sensor and the world that influence the density of the point cloud, the real world shows to be a lot more complex and, as a result, a lot harder to mathematically model. Therefore, during the development of this simulator, some measures had to be taken to simplify the problem at hand and focus on the more important aspects of it.

Given the initial stages in the study of this new approach, it was decided as a first step to better study and understand the more basic situations that we can encounter in the real world. As a consequence, for now, the simulator is only capable of replicating the movement of the vehicle in a single direction, with a velocity only along the X axis. Additionally it is always assumed that the vertical face of the curb is perfectly flat (a plane), which could be an over-simplification when studying larger filtering regions.

Concerning the sensor, some simplifications have also been made, namely the errors associated to the measurements have not been modelled, and other features of the world that could interfere with the measurements such as moving vehicles or pedestrians, have also been left out. Additionally, only constant velocity for the vehicle is implemented in the model, meaning that the displacement between scans are the same during each individual simulation.

Even though many simplifications have been made when comparing to real world situations the purpose of this simulator is to provide general guide-lines of the expected APCD values, and it is not intended to provide strict rules for the *AtlasCar2* navigation, so the impact of these simplifications is diminished and the obtained results still provide very useful information about the more specific workings of the APCD methodology.

4.3 Point Cloud Processing and APCD filtering

After having the general guide-lines provided by the studies generated on the simulator about the accumulation behaviour in specific situations, it is important to transport this same methodology to the real world case.

An algorithm had to be developed to measure the APCD from the road reconstruction point cloud, in a similar way as the method used in the simulator, so that the conclusions about data density and sensor placement drawn using the simulator can be applied in this algorithm.

4.3.1 PCL Radius Filtering

The objective for the road boundary detection module is to provide a point cloud with only the data belonging to the limits of the road, to be used in navigation algorithms in the *AtlasCar2* project. The chosen approach to build this point cloud is to start with the road reconstruction point cloud and filter it by removing the irrelevant points and leaving the ones that refer to the identified road limits.

The filtering processes has to be capable of measuring the density in a region of the point cloud and, based on that density, make the decision to discard or keep the points based on criteria obtained from the simulations.

So, the chosen method for filtering the point cloud involves the use of the *Radius Outlier Removal* algorithm from the filters module of the Point Cloud Library (PCL) [36]. The working principle of this filter is very straightforward: the user defines a radius d and a number of neighbours N_{total}, then the algorithm goes through all the points in the cloud one at a time counting the number of neighbour points contained within a sphere with radius d, the point will be considered an outlier if it has too few neighbours based on the k parameter, otherwise it is kept.

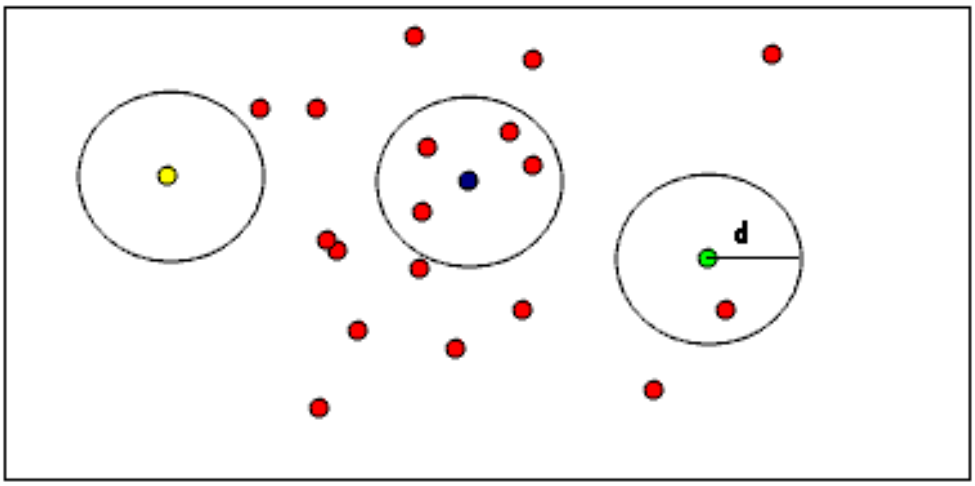


Figure 4.8: Illustration of the *RadiusOutlierRemoval* filter working principle [37].

This algorithm allows to make use of the APCD values obtained from the simulations by translating them into the radius and the number of neighbours parameters of the filter. Additionally, these parameters can be changed inside the code at runtime, allowing

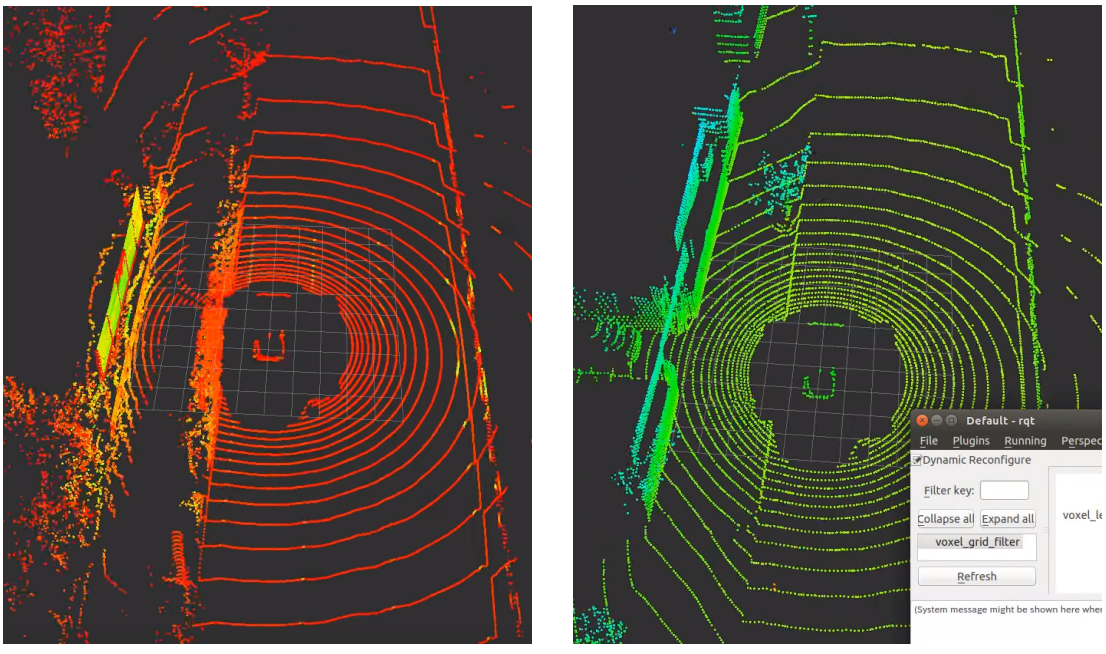
for the APCD values to change according to the situation of the vehicle, such as its orientation and velocity that will influence the APCD requirements, granting the filtering processed the ability to adapt based on other measured data.

4.3.2 Optimization of the Filtering Process

To have useful reliable enough data to use in navigation algorithms, the generated point cloud must have information about the road limits in a reasonable distance in front of the vehicle, so that it can make its decisions before the vehicle reaches the obstacle. Since vehicles are built to move at relatively high speeds, small delays in processing may cause the loss of information about the distance of objects farther away.

Because of this it is important that the processing of the point cloud and identification of the road boundaries be made in the least time possible, so as to diminish the loss of information. The method described above is very fast and simple, but there are still some ways to improve it.

One of these ways is to implement a *Voxel Grid* filter [38] to the point cloud before processing it with the *RadiusOutlierRemoval* filter. The *Voxel Grid* filter is a technique to down-sample a point cloud, reducing the number of points of the dataset. The approach consists of creating a grid of 3D boxes (voxels) covering the whole point cloud, and approximating all the points inside each voxel to its centroid, effectively reducing the number of points while still accurately maintaining the underlying surface. The size of the voxels (leaf size) can be controlled allowing to regulate the intensity of the down-sample, Figure 4.9 shows an example of the *Voxel Grid* filtering process.



(a) Raw point cloud data.

(b) Point could data filtered with a 0.09 leaf size *Voxel Grid*.

Figure 4.9: Example of Voxel filtering in a LIDAR generated high density point cloud.

In this case, a very fine *Voxel Grid* filter was applied, with a leaf size of 0.05 m in all directions, this provides some down-sampling of the cloud while maintain all the relevant information regarding the point concentration.

Another way to decrease the processing time is to evaluate only the density of the relevant points, i.e only the points of the cloud in front of the vehicle. This was achieved by applying a *Conditional Removal* filter [37] to eliminate all the the points outside of a defined bounding box. Figure 4.10 illustrates the bounding box and its dimensions.

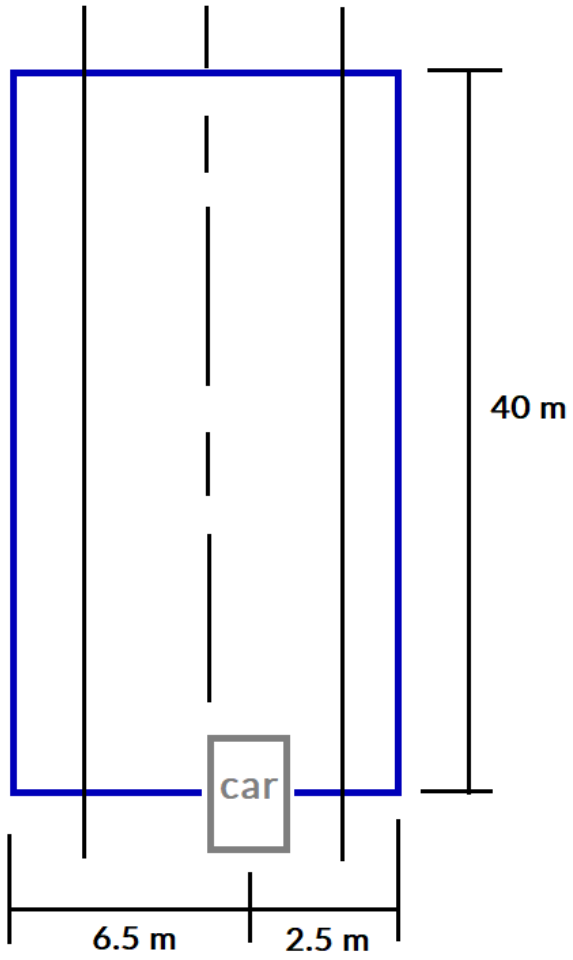


Figure 4.10: Dimensions of the bounding box for the *Conditional Removal* filter.

With these two pre-processing methods the number of points in the cloud has been significantly reduced, which translates to much fewer points that the *Radius Outlier Removal* will have to cycle through, making this a fast method for the road boundary detection. However for the purposes of the work presented in this document, the *Conditional Removal* filter was not employed, because it limits the size of the cloud, eliminating the filtered regions behind the vehicle. Even though these regions are not relevant for navigation purposes (because they represent sections of road already passed by the car), they provide better visual information about the continuity of the method when detecting the road boundaries, which is important for evaluating its performance.

Chapter 5

Experiments and results

The next stage in this project is to test and validate the approaches chosen to solve the vehicle position (Chapter 3) and the road boundary detection (Chapter 4) problems, proposed in this work.

This chapter will begin by detailing the experiments performed for evaluating the accuracy of both the localization and orientation components of the *Road Reconstruction* module. These tests served to verify if any errors occur resulting from the sensor measurements or the transformations between frames in this module.

Next the method for evaluating the road boundary detection algorithm will be presented, along with some tests performed in both controlled and real-world situations. Regarding the study in a controlled environment, a series of experiments for the calculation of APCD ranges were conducted using the simulator described in Section 4.2.1, to evaluate the characteristics of the accumulation method for various sensor and curb configurations. The results obtained from the simulations were then used to tune the *RadiusOutlierRemoval* algorithm used for the real world boundary detection, and after implementing this processing algorithm in the *AtlasCar2*, some tests were conducted to test its accuracy in real world situations.

5.1 Vehicle Positioning Results

5.1.1 Global localization

The developed solution for road boundary detection relies on having a point cloud representative of the road in front of the vehicle. That point cloud is generated by the *Road Reconstruction* module and the accuracy with which it translates the sensor data to a reconstruction of the road is dependant on the information about the instant position of the *AtlasCar2* relative to a fixed world frame.

To evaluate the implemented solution for vehicle positioning a real world test was performed by driving the *AtlasCar2* along a closed circuit and registering the global position of the car along the route. Then the *mapviz* application from the *Southwest Research Institute Robotics* toolbox [39] was used to represent that data in a 2D map.

The position results are presented in Figure 5.1, that shows a satellite view area around University of Aveiro where the circuit was made, in the image the blue line represents the successive position measurements made by the *Novatel SPAN-IGM-A1* which are used for vehicle localization.

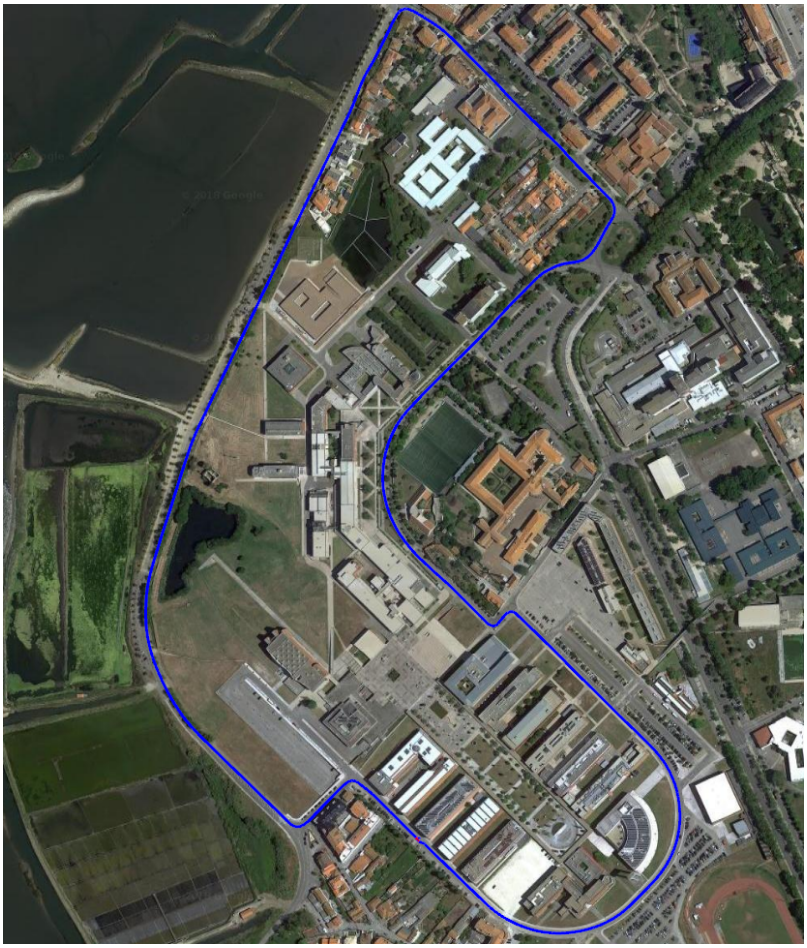


Figure 5.1: Circuit around University of Aveiro.

The circuit travelled by the *AtlasCar2* in this experiment measures a distance of approximately 3km. As observed in figure 5.1, the travelled path (in blue) is very accurately described along its course. Figure 5.2, shows a detail view of the start and end points of the circuit.

Figure 5.2: Start and end points of the circuit (Department of Mechanical Engineering).

However since this information is used to reconstruct the road in front of the car, it is important to determine the accuracy of this method from a more local point of view. Figure 5.3, represents a zoomed in satellite view, acquired using the *mapviz* application, of a section of the road where the path of the *AtlasCar2* is again represented in blue. From this image we can qualitatively evaluate the high accuracy of this system, by observing that the path obtained by the position data is drawn in the correct lane of the road.

Figure 5.3: Top view of the measured global position is a straight stretch of road.

5.1.2 3D Orientation Results

The 3D orientation module is an important aspect of the work developed, since it serves as a base on top of which the other systems are referenced. This module provides information to establish the transformations between the several frames of the *AtlasCar2* model, and so it is important for that information to be correctly interpreted by the system to apply the correct transformations among the several frames, allowing it to

accurately represent the sensor readings based on the information about the vehicle's orientation.

The approach was to test the accuracy with which the information about the vehicle's orientation is used within the transformation tree, and evaluate if the correct transformations are being applied by the system. For this and experiment was developed where the *AtlasCar2* was placed in front of a flat vertical wall, projecting the laser scans on its surface. Next a car jack was placed on the rear of the vehicle in order to lift it and increase its pitch in a controlled environment. Figure 5.4 illustrates the setup of the experiment.

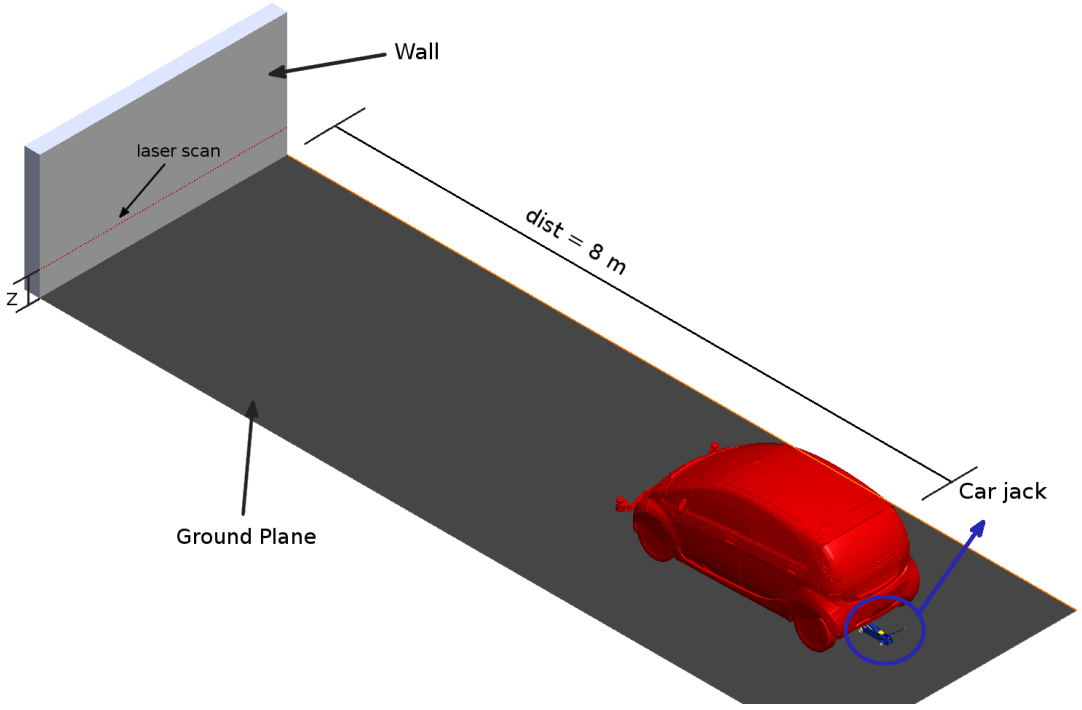


Figure 5.4: Representation of the setup of the experiment.

From this experiment a central point of the laser scan is selected and the goal is to measure its height relative to the ground plane, obtained through the transformation tree of the system, and comparing it to the expected theoretical value, obtained by the geometric model of the transformation tree.

To obtain the height measured by the system, the software was adapted to simultaneously register the *(X,Y,Z)* data relative to the *ground* frame of a chosen point from each of the four laser scan planes, and the *pitch*, *roll* and *Z_mean* (height if the car measure by the distance sensors) components of the vehicle's pose. Next, using a car jack placed on the rear of the vehicle, the *pitch* was slowly increased from 0° to 2° and the height values of the points were registered alongside with the measured values for *pitch* and *Z_mean*.

The theoretical measurements where obtained through a mathematical model of the transformation tree in place on the *AtlasCar2*. Figure 5.5 illustrates the frame tree with the dimensions of the distances between frames.

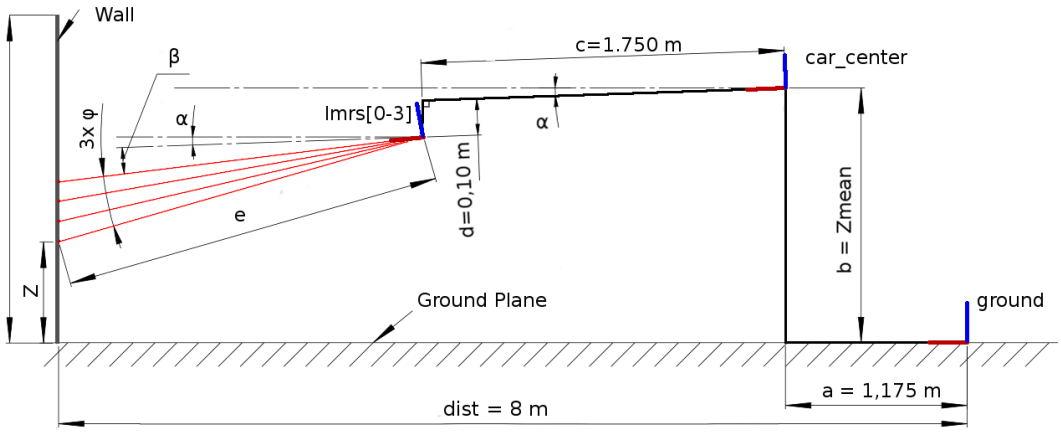


Figure 5.5: Diagram of the transformation tree and its dimensions and variables in the context of the experiment (Not to Scale).

From this scheme the mathematical equations for the model where determined, and are described in equation 5.1, using the variables presented in figure 5.5. Additionally, the experiment was performed using a $\beta = 0$ configuration for the LIDAR sensor, since this is the tilt value of the sensor currently installed in the *AtlasCar2*.

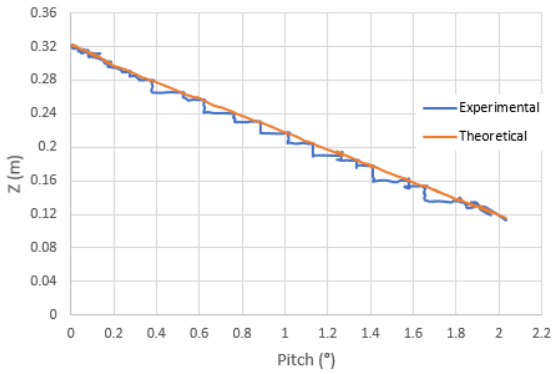
$$\begin{cases} dist = a + \cos(\alpha) \cdot c - \sin(\alpha) \cdot d + \cos(\alpha + \beta + \phi_i) \cdot e_i \\ Z = b - \sin(\alpha) \cdot c - \cos(\alpha) \cdot d - \sin(\alpha + \beta + \phi_i) \cdot e_i \end{cases} \tag{5.1}$$

This system of equations was used to calculate the theoretical height(Z) for each of the scan planes, using the *pitch* and *Z_mean* values gathered from the variable radius method.

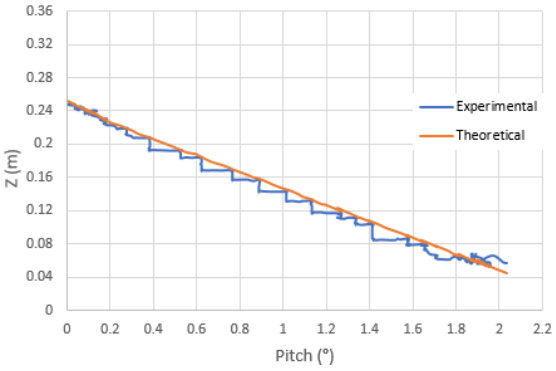
Figure 5.6, shows the results for the height determined by both methods for each of the sensor scan planes.

Using a car jack does not allow to increase pitch in a continuous way due to the nature of this device, this, coupled with the presence of a delay for the changes in pitch to be reflected on the height measurement of the scan plane, cause the "staircase" effect that can be observed in the experimental results. Additionally in figures 5.6b and 5.6d of the lowest two laser scans, from 1.8° and 1.03° respectively, the scan planes leave the vertical face and hit the ground plane instead. In this situation the experimental results prove to be less accurate presenting in some cases deviations of 20 mm from the expected height value ($Z = 0$ mm).

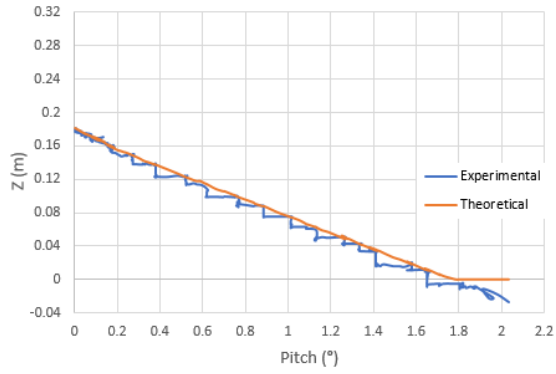
Despite this, the results show that overall the system accurately represents the correct position of the laser scan points, showing the experimental height measurements very close to the theoretical ones, with the small deviations justifiable by the sensor's statistical error (table 2.1). As such we can conclude that the transformation tree is being correctly implemented by the system and producing the expected measurements, allowing the system to accurately compensate for changes to the vehicles orientation.



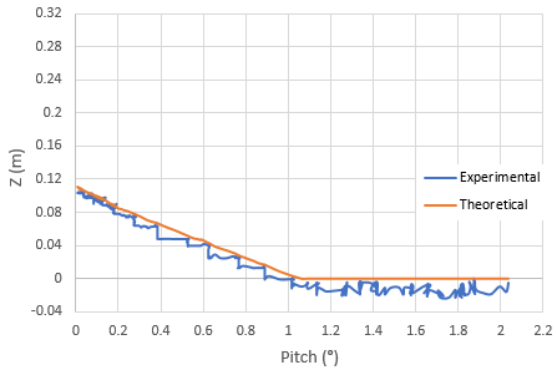
(a) Experimental and theoretical height of the measuring point with pitch, for the top most laser scan plane (*ldmrs3*).



(c) Experimental and theoretical height of the measuring point with pitch, for the second from the top laser scan plane (*ldmrs2*).



(b) Experimental and theoretical height of the measuring point with pitch, for the second from the bottom laser scan plane (*ldmrs1*).



(d) Experimental and theoretical height of the measuring point with pitch, for the bottom most laser scan plane (*ldmrs0*).

Figure 5.6: Experimental and theoretical results for the height values of the four scan planes with vehicle pitch.

5.2 Road Reconstruction Results

The systems to determine the vehicle instantaneous position and orientation validated in Section 5.1 of this chapter, were built with the goal of correctly creating a map of the area in front of the *AtlasCar2* during motion. These systems provide crucial information needed to translate the sensor data into the point cloud that accurately represents the perceived environment. It is important for this created "map" of the road to be precise because it will serve as the foundation to extract the road obstacles and consequentially be a guide for vehicle navigation in autonomous navigation situations.

In this section, several segments of the road obtained by the *Road Reconstruction* module will be presented, allowing to evaluate the performance of the system and the general topology of the resulting point cloud during several situations that the car can encounter during navigation.

5.2.1 Influence of the Orientation Module in Road Reconstruction

It is important to note that, as mentioned in Section 3.4, to simplify the reconstruction problem, the road was considered to be locally flat, meaning that its orientation relative to the world was ignored, and because of this all the images that follow show the road as a flat plane at ground level.

This simplification means that the points of the reconstructed road point cloud, should not be represented below the ground plane ($Z < 0$), and, in situations of turning, breaking, or accelerating, that cause changes to the vehicle's pose, the orientation module, validated in Section 5.1.2, should compensate this and correctly represent the scan point above the ground plane.

To test if this behaviour occurs, the *AtlasCar2* was driven through a flat section of road that is preceded by an intersection, in which the vehicle has to slow down and perform a clock-wise 90° turn, causing an aggressive change to its *pitch* and *roll* components. Figure 5.7 shows a photo of the section of road traversed in this experiment.

Figure 5.7: Photo of the intersection followed by the flat section of road, travelled by the *AtlasCar2* for this experiment.

During navigation, the road reconstruction point cloud was generated for two separate situations. First, without using the orientation module, by fixing the *pitch* and

roll information to zero, and, in the same stretch of road, using the reconstruction module to correctly represent the LIDAR measurements. Figure 5.8 represents a isometric view of the reconstruction point cloud for the case where the orientation module was used, and figures 5.9 and 5.10 represent side and rear views for both of these situations, respectively, at the exact same time and location.

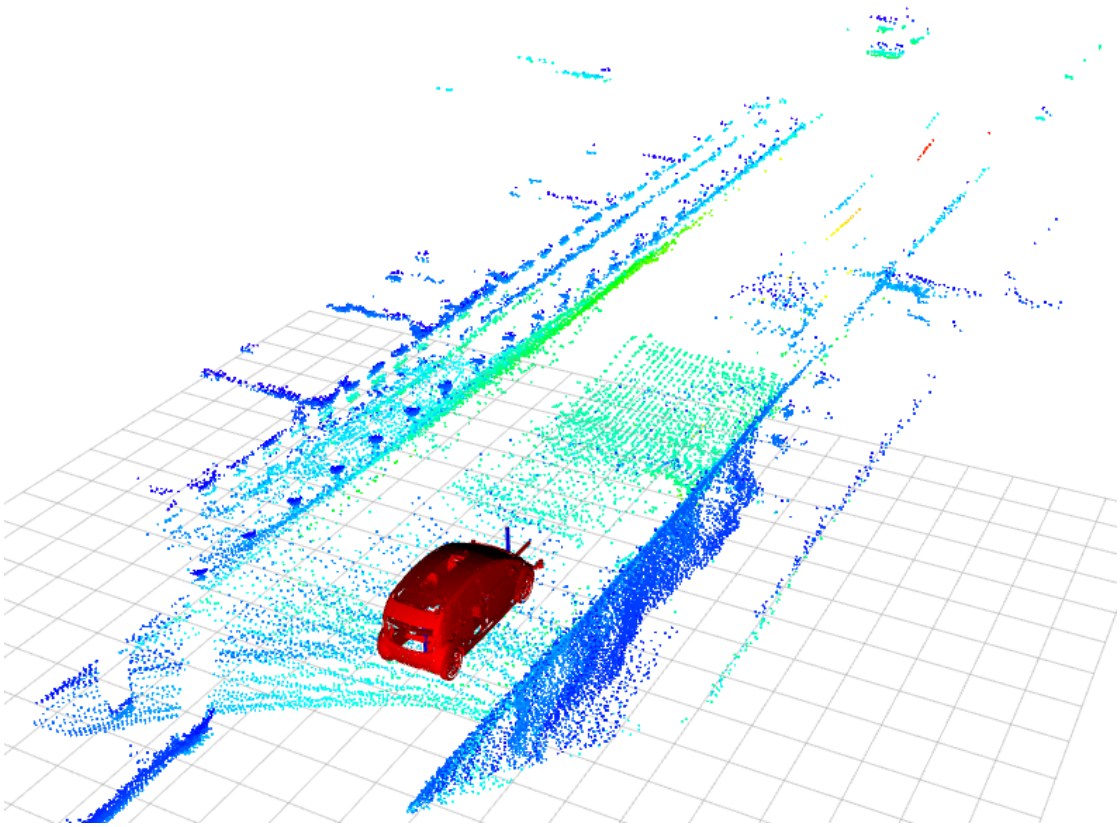
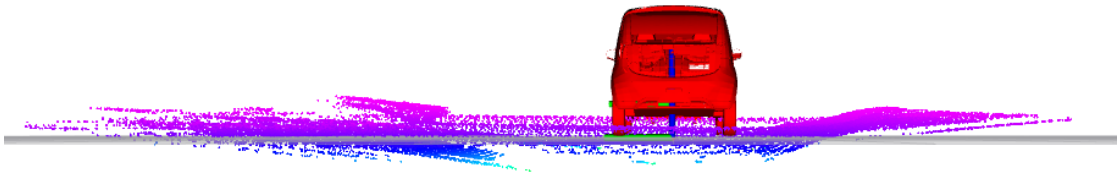


Figure 5.8: Isometric view of the road reconstruction point cloud after the intersection. Point cloud is 16 m wide and 28m long.

During this road section the vehicle's *pitch* changes from about 0.53°, during breaking, to $-0.9°$ during the acceleration after the turn, and the *roll* changes from 0° to 0.9° and back 0° during the turn. The dimensions of the sections of road a presented in figure 5.8 and are the same for both cases, additionally the maximum and minimum values for the points of the cloud where measured and are presented in their respective figures.

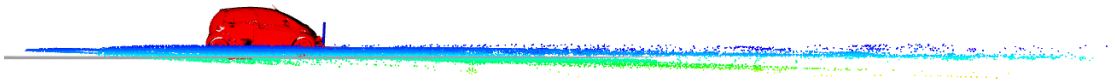
Analysing the figures we can observe the beneficial results of correcting the LIDAR measurements with the orientation data. For instance, comparing the rear images from both cases we see that in figure 5.9b the points on left side of the curb are represented much lower than the points on the right side, this is caused by the *roll* rotation caused by the clock-wise turn, in contrast in figure 5.10b, representing the compensated cloud, the points from both sides of the road have approximately the same height, suggesting that the measurements were correctly compensated, also from these images we see that both failed to completely eliminate the influence of the vehicle orientation but generally

(a) Side view of reconstruction point cloud, gathered without the use of the orientation information.

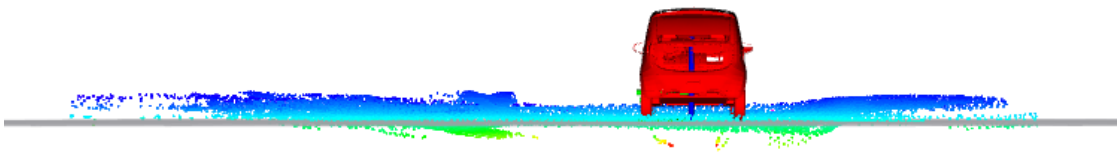


(b) Rear view of reconstruction point cloud, gathered without the use of the orientation information.

Figure 5.9: Section views detailing the reconstruction of the road section relative to the ground plane, without 3D orientation information - $Z_{min} = -1.6$ m and $Z_{max} = 0.3$ m.



(a) Side view of reconstruction point cloud, gathered using the orientation information.



(b) Rear view of reconstruction point cloud, gathered using the orientation information.

Figure 5.10: Section views detailing the reconstruction of the road section relative to the ground plane, using 3D orientation information - $Z_{min} = -0.85$ m and $Z_{max} = 0.64$ m.

the compensated point cloud produces negative points closer to the ground plane, as evidenced by the Z_{min} parameter which is much smaller in this case. Another evidence of this can be observed in the side views of both cases, where the the situation without orientation data, figure 5.9a, presents a tilted reconstruction, caused by the acceleration of the car, were points of the points of the road are represented below the ground plane and the points of the curb are very close to the ground, in the reconstruction using the orientation data, figure 5.10a, this also happens but the tilt effect is less aggressive, with the road points closer to the ground plane, and the curb points still above this plane, as suggested once again by the Z_{min} and Z_{max} from both cases.

Overall the use of the orientation module brings some benefits to the reconstruction, despite not eliminating all the effects of the changes in vehicle's orientations, it allow for a more uniform point cloud that better represents the real world scenario.

5.2.2 Influence of Vehicle Movement on Road Reconstruction

Given the nature of the method used, during the vehicle navigation there are many factors that can influence the resulting point cloud, for example straight routes, turns, acceleration and decelerations, all have different impact in the reconstruction. Next some of these situations will be explored and the reconstruction results will be presented and discussed.

Figure 5.11 shows a top view of the *AtlasCar2* with a section of a reconstructed road point cloud, corresponding to a simple situation, where the car is travelling in a straight road at regular velocity. From this reconstruction we can gather some important information about this system. By observing the length of the point cloud created in front of the vehicle we can distinguish two instances: closer to the vehicle (about 8 m from it) the point cloud is more dense containing information about both the road and the objects and features around it, such as the parked cars to the left of the image and the side-walk to the right; further from the vehicle some information about the road is lost but the points that intersect vertical features are registered, this means that the reconstruction provides information about the section of road approximately 17 m in front of the car. It is also relevant to note that the motion of the vehicle during its navigation causes its front to lift, imposing a pitch of about $-0.3°$ or more depending on the velocity of the car, and helping the sensors measure distances further away than they normally would.

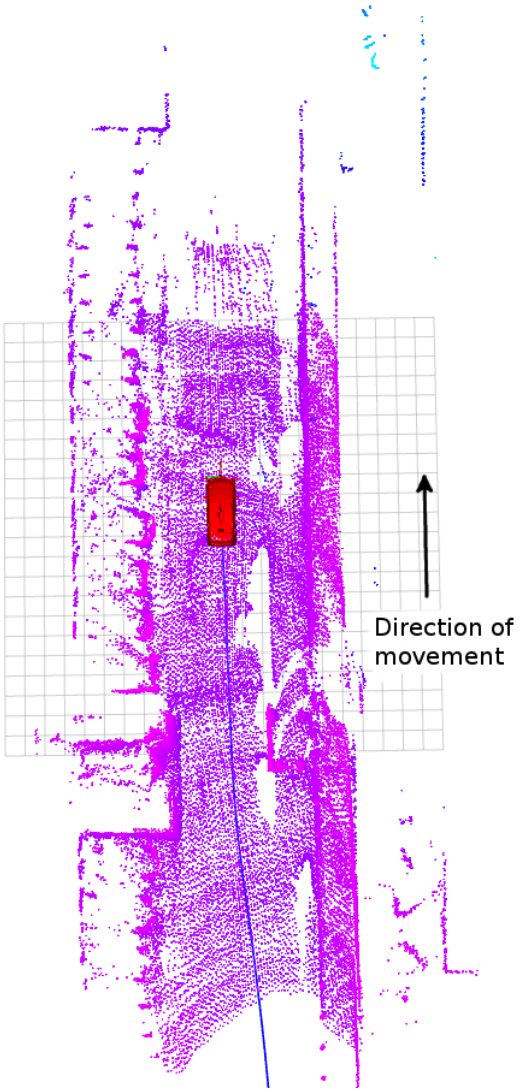
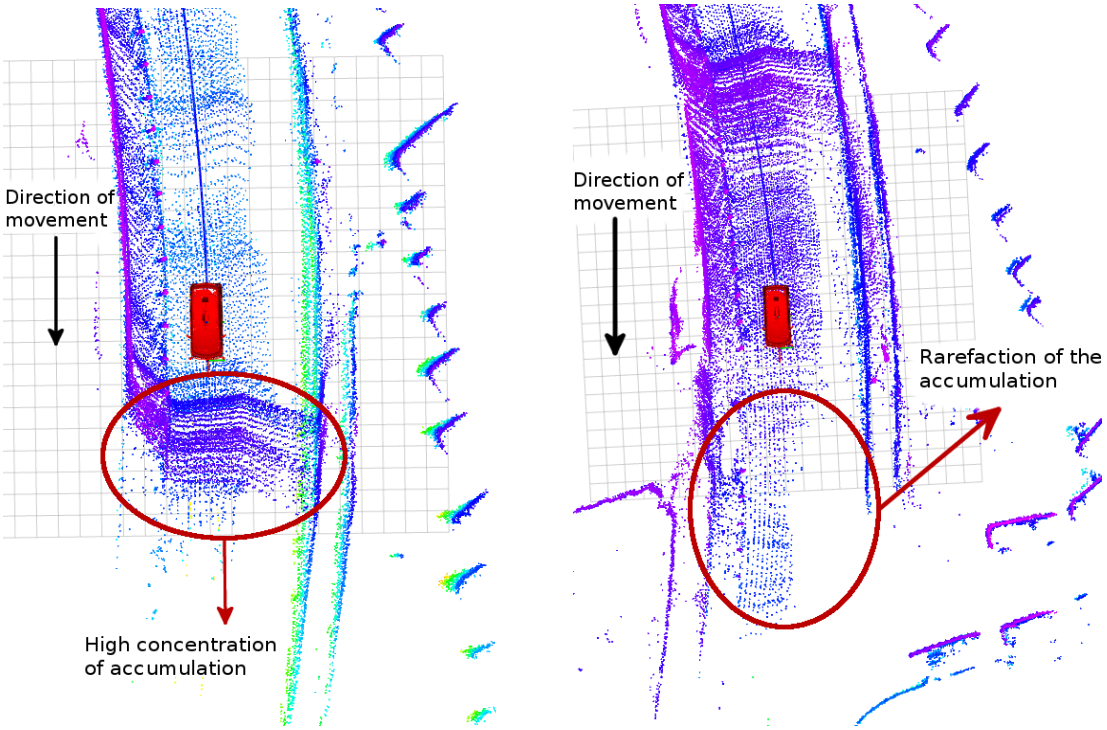


Figure 5.11: Top view of the road reconstruction on straight section of road in regular conditions of approximately: velocity (40km/h) and orientation levels (pitch: $-0.35°$, roll: $0°$)

However during navigation there are situations where the vehicle is rapidly changing its velocity, accelerating and decelerating, this causes great fluctuations to the vehicle's pitch, and consequentially greatly alters the orientation of the sensor and its measurements. To test the influence of these factors experiments where performed in which the driver of the car rapidly changes the vehicle velocity by accelerating and breaking and the resulting point cloud was gathered along with the information about the car's pitch. Figure 5.12 shows the generated point cloud during this experiment, where 5.12a refers to the deceleration and 5.12b to the acceleration of the vehicle. Additionally all the results presented in this section make use of the vehicle orientation data to correct the sensor measurements.

Analysing the first image (5.12a) we can observe the impact of the increase in the pitch of the vehicle, which causes the sensor to point aggressively downwards and as

a consequence, measure distances much closer to the car. This effect induces some undesired behaviours, in that it causes the reconstruction algorithm to temporarily lose information about obstacles further way from the vehicle and creates a zone with much greater density of points than the areas around it. For the situation presented in the second image (5.12b), that refers to the heavy acceleration causing the front of the vehicle to lift and decreasing its pitch, similar but opposite behaviours occur. In this case the orientation of the sensor causes it to register points further from the vehicle but the density of the reconstruction is diminished relative to the surrounding areas.



(a) Accumulation of laser scans during heavy deceleration, causing a change to vehicle pitch from $-0.55°$ to $0.19°$.

(b) Accumulation of laser scans during acceleration, causing a change to vehicle pitch of from $-0.46°$ to $-0.9°$.

Figure 5.12: Influence of pitch in the road reconstruction point cloud.

One other common factor that influences the reconstructed road is when the vehicle performs a turn; this motion produces an increase or decrease, depending on the direction of motion, in the roll component of the vehicle, causing the sensor to rotate along its axis. Figure 5.13 demonstrates a situation where the vehicle is performing a clock-wise turn, which causes a rotation of the sensor. This behaviour results in a diagonal projection of the laser scans on the road plane where in this case, because of the clock-wise turn, the laser scan to the left of the car hit the ground closer to the vehicle than the right side ones. As such the high complexity of the vehicle movement during turns, caused by sensor rotation and the varying vehicle velocity, results in regions with very uneven local densities, making it very difficult to predict the expected density in these situations.

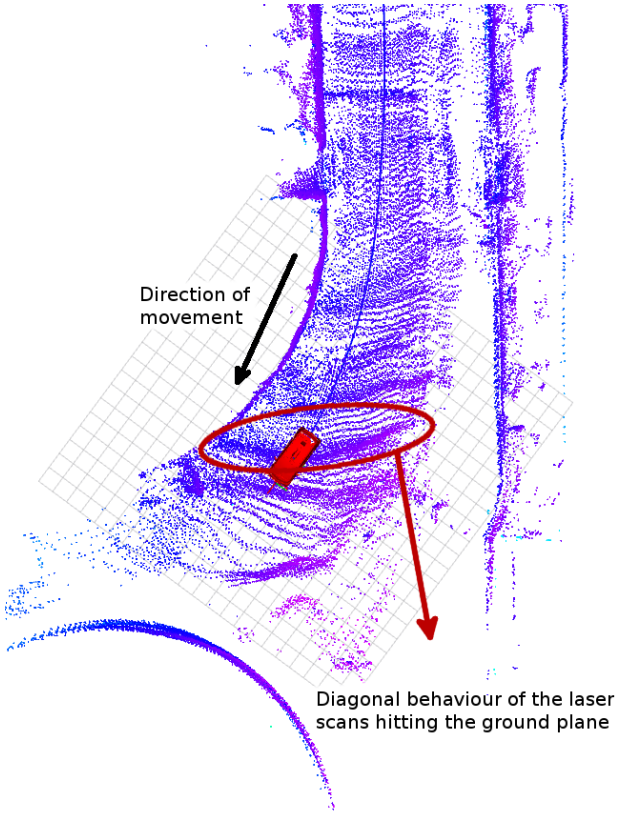


Figure 5.13: Top view of the road reconstruction of the *AtlasCar2* performing a turn, inducing roll levels of about 1.4°.

All these situations, produced by changes to the vehicle's orientation and/or its velocity cause temporary changes to the local point density of the road reconstruction point cloud and will have to be taken into consideration when implementing the filtering algorithm for the road boundary detection.

5.3 Simulation Results

After having the reconstructed road point cloud, the next step is to determine the parameters for the *RadiusOutlierRemoval* so that we can use it to filter the cloud and obtain the limits of the road. However given the complexity and the number of variables that can influence these two parameters, a simulator was built evaluate their influence on APCD, as described in Section 4.2.1.

The two main parameters needed for the *RadiusOutlierRemoval* filtering algorithm are the radius of the filtering sphere and the number of intended laser scan points contained within that sphere. Because this method is dependent on two variables, to test the influence of the various external parameters that can cause changes in the measured APCD, like vehicle velocity, sensor placement and curb geometry, we first have to establish a fixed value for one of the *RadiusOutlierRemoval* parameters to allow to calculate the density and to have consistent results from the experiments that follow. As such, an initial simulation was performed with the intent of determining the ideal

sphere radius parameter for the filter, and use this value in the remaining simulations that evaluate the influence of external parameters in APCD.

For this, a simulation was made where the vehicle is moving at a standard speed of 50 km/h and the road limits are defined by a Type I curb (Figure 5.14), meaning a vertical curb on each side of the vehicle (1.5 m to the right and 4.5 m to the left), the accumulation point is located at 14 m from the initial position of the vehicle, and the simulation ends when the car moves 12 m.

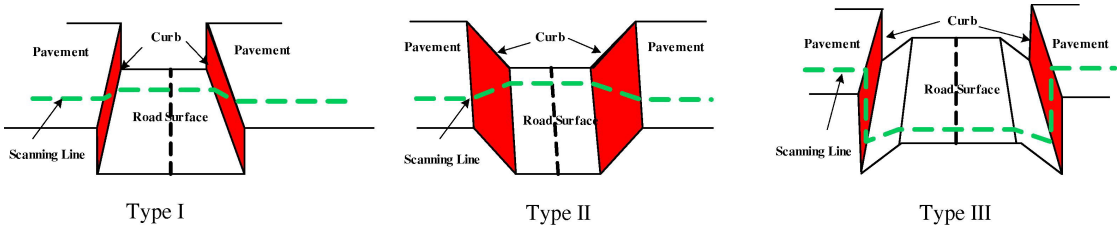


Figure 5.14: Types of road profiles [10].

Figure 5.15 shows a representation of the simulation scenario used in this experiment and all the ones that follow, representing the constant variables common to these simulations.

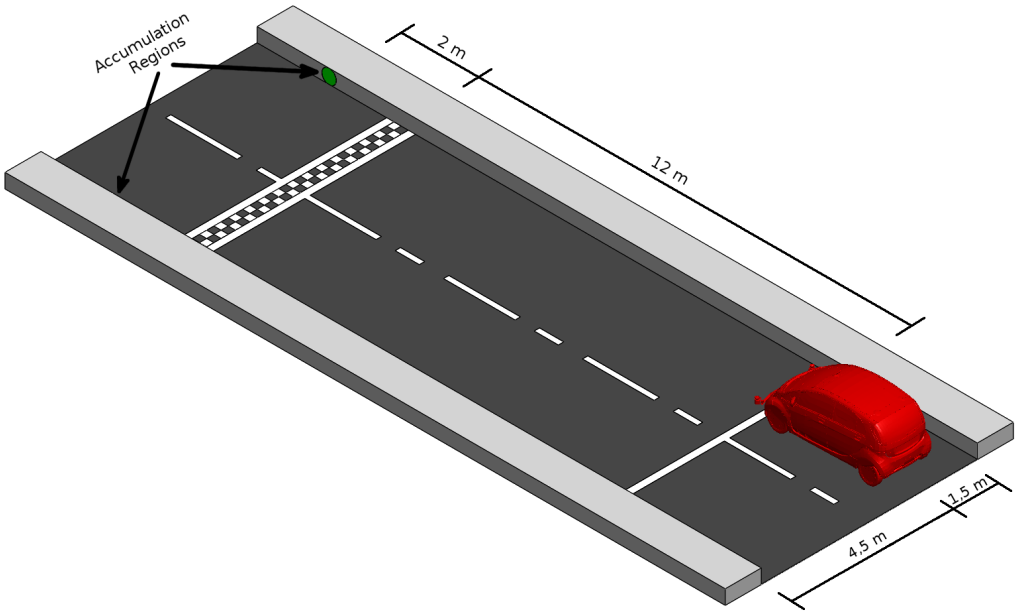


Figure 5.15: Setup of the simulation scenario.

Using this configuration the APCD was measured according to Equation 4.6 for multiple radii of the filtering circle. Figure 5.16 shows the result of the simulation, where the *Density* refers to the APCD in $points/m^2$, and X refers to the distance travelled by the vehicle in meters.

From the graph we can observe that the 0.2 m radius is the one that produces a higher APCD followed by the 0.1 m radius but this shows a more inconsistent behaviour

having some intervals where no points where contained in the circle due to it being to small.

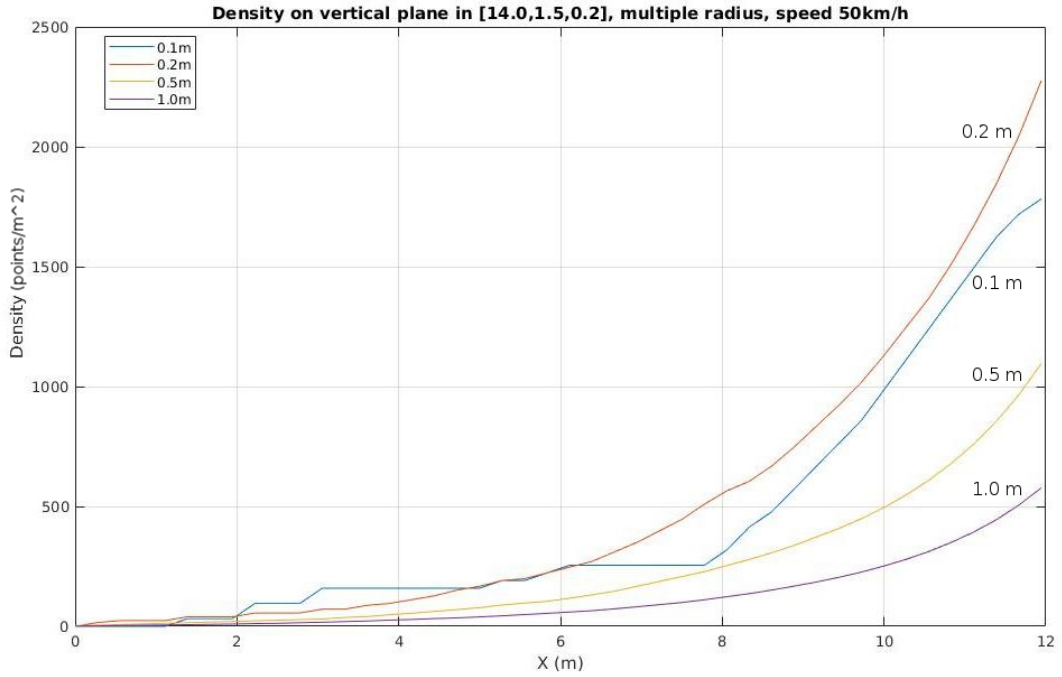


Figure 5.16: Influence in APCD of the filtering circle radius, for accumulation in a vertical plane.

Another important aspect of this methodology for road boundary detection is the difference between the APCD measured on the curb and on the road plane. So, it is important to maximize that difference since doing so will facilitate separating the two planes. For this reason another simulation was performed, in the same conditions as the previous, to evaluate the influence of the filtering radius in APCD but this time with the measuring region on the road plane. Figure 5.17 shows the results of the simulation. From this graph we can observe that generally the APCD values are much lower (about 25 times less) than the ones registered in the vertical curb from the previous experiment. Even so the 0.2m radius shows to be the one that results in lower APCD values, making it the preferable option.

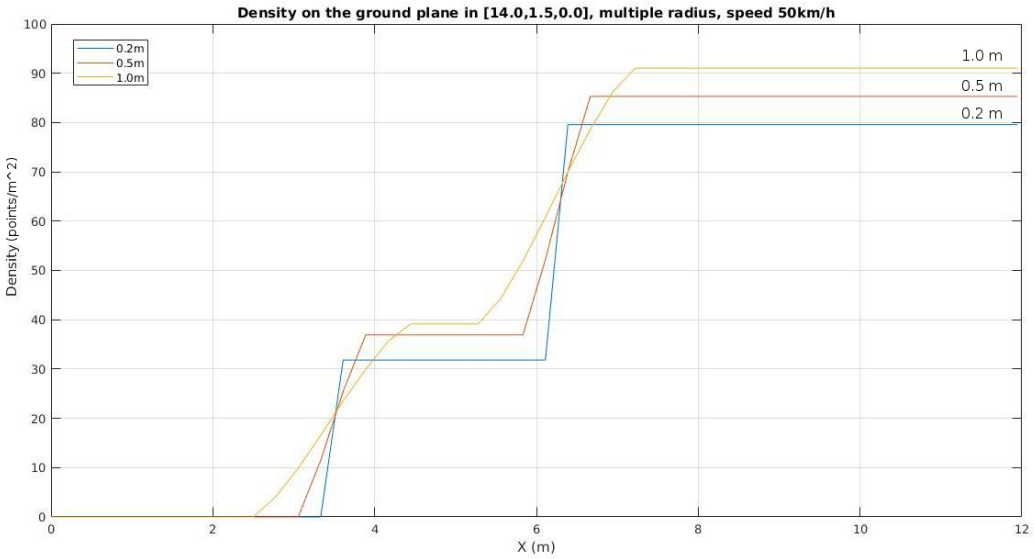


Figure 5.17: APCD values on the road plane measured using different radii for the filtering circle.

Given the results gathered from the these two experiments, the 0.2m radius proved to be the ideal size of the filtering sphere in both situations resulting in greater densities on the curb plane and lower ones in the ground plane, meaning that it is the one that best differentiates between the two surfaces. As such all the following simulations will be evaluated with a filtering circle with a radius of 0.2 m.

5.3.1 Influence of Vehicle Velocity in Point Cloud Density

During the navigation it is expected for the vehicle to frequently change its velocity, and since the rate of data acquisition from the LIDAR sensor is always the same, these fluctuations in speed will cause the interval between successive readings of the sensor to also change. This results in different density regions with different vehicle speeds.

To test how the velocity of the car influences the values of APCD a simulation was preformed, to calculate APCD on a Type I curb using a filtering circle with 0.2 m radius, during three different velocity scenarios, evaluating in each, the rate of the accumulation which measures the APCD relative to the time that takes the vehicle to travel the simulation distance (Figure 5.18 and the accumulation with proximity to the measuring point (Figure 5.19).

The results of the experiments agree with the expected behaviour, where the lower velocity generates much higher levels of APCD, but at a slower rate it takes more time to reach the same density levels. This behaviour suggests that the filtering algorithm needs to have a method for adjusting the APCD threshold based on the velocity of the vehicle and the proximity to the accumulation point, and that this relations should follow the results obtained from the graph in Figure 5.19, where the density grows exponentially with the proximity to the point.

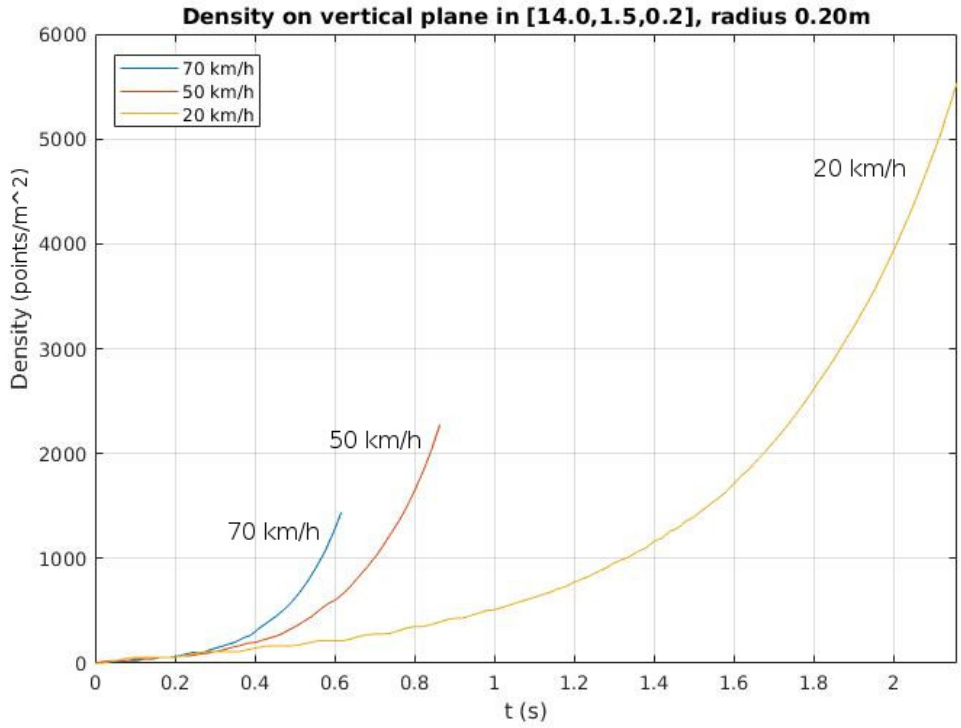


Figure 5.18: Influence of vehicle velocity in APCD with vehicle travel time, in a Type I road profile with filtering circle of radius 0.20 m

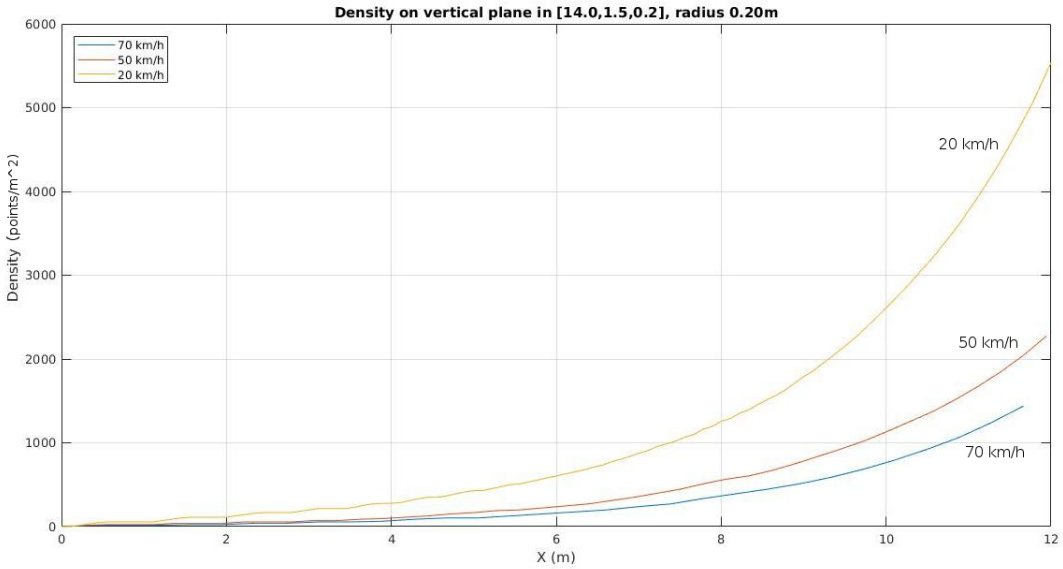


Figure 5.19: Influence of vehicle velocity in APCD with distance travelled, in a Type I road profile with filtering circle of radius 0.20 m

5.3.2 Influence of Sensor Placement in Point Cloud Density

Another parameter that influences the point accumulation is the placement of the LIDAR sensor on the *AtlasCar2*. As shown in Figure 2.5 the sensor is placed below the license plate of the vehicle at about 0.35 m from the ground and with an incidence angle of about 0.6° (angle between the topmost laser scan plane and a plane parallel to the ground). However the placement of this sensor is crucial since it will influence the projection of the laser beams in the various road features and the range of the measurements, so its important to find the ideal conditions for its placement.

To test this, several simulations where performed once again using a Type I road profile with a filtering circle of 0.2m and a velocity of 50 Km/h. First the influence of the distance from the sensor to the ground was tested, by performing five experiments where the sensor was placed between an hight range of 0.2 to 0.6 m at 0.1m intervals. The results of this simulation are presented in the graph of Figure 5.20.

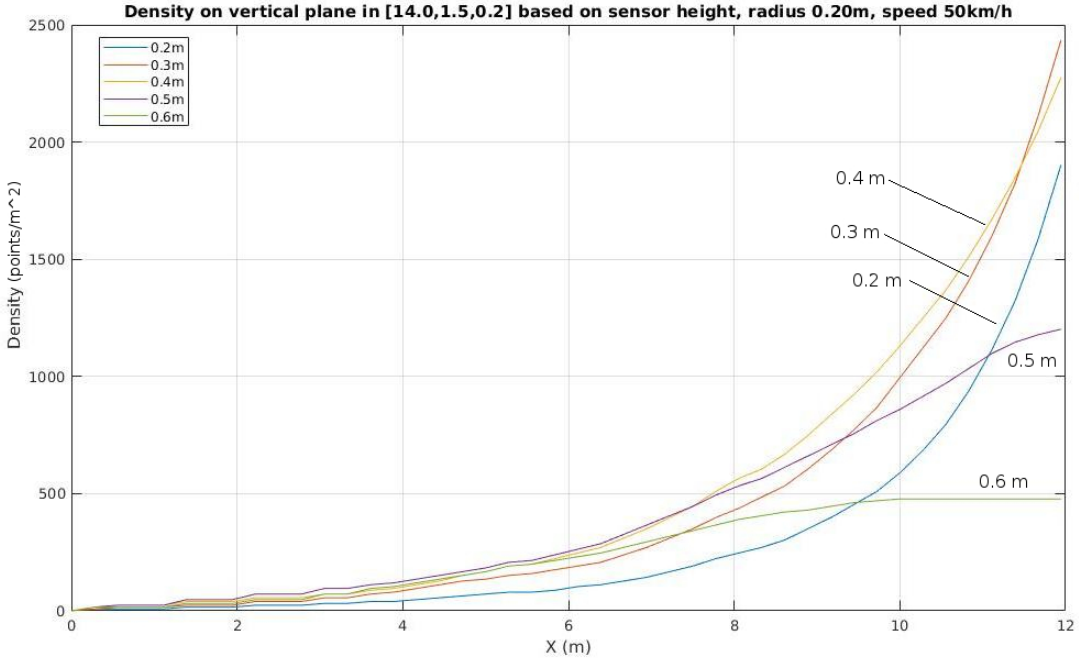


Figure 5.20: Influence of sensor height in the APCD, in a vertical plane with filtering circle of radius 0.20m

The results show that sensor heights between 0.3-0.4m produce the highest APCD levels of about $2250 points/m^2$ and $2500 points/m^2$ respectively, with the 0.5m and 0.6m heights producing significantly worse results of about $1200 points/m^2$ ans $500 points/m^2$ each. As such the height of the LIDAR sensor in the current configuration of the *AtlasCar2* proved to be ideal, since it produces the higher APCD.

Next the influence of the sensor incidence angle was tested, under the same conditions and using the sensor at the ideal 0.4 m height determined by the last experiment. For this test 5 simulations where performed varying the incidence angle from 0° to 3.6° at 0.6° intervals. Figure 5.21 shows a graph with the obtained results.

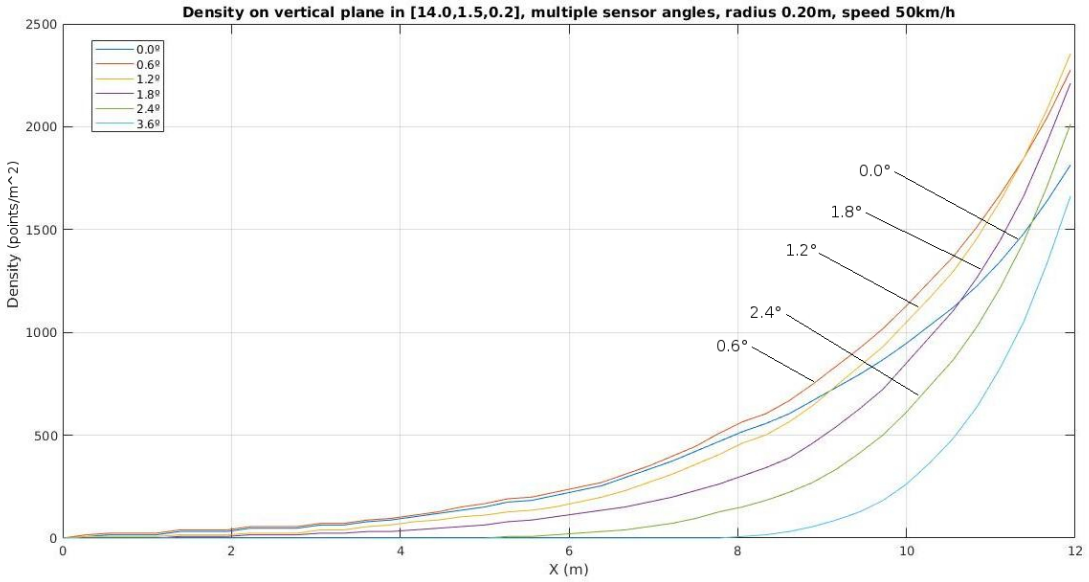


Figure 5.21: Influence of sensor angle, in a vertical plane with filtering circle of radius 0.20m

Table 5.1: Density values for several combinations of sensor positioning.

Height \ Angle	0°	0.6°	1.2°	1.8
0.3 m	533.17	437.68	270.56	111.41
0.4 m	517.25	565.00	461.55	302.39
0.5 m	421.76	533.17	565.00	461.55
0.6 m	278.52	389.93	485.42	525.21

The results show that sensor incidence angles between 0° and 1.2° produce the higher densities and show very similar results for about the first 9 m of the accumulation, with the difference between the angle 0° and the other two only becoming apparent when the vehicle get closer to the accumulation point. Overall the angles between 0° and 0.6° seem to be the ideal solution since they combine high APCD values without losing depth information (distance to obstacles) caused by high incidence angles.

Additional experiments where performed to evaluate more combinations of these two parameters and measure the resulting APCD, since other arrangements of the sensors outside the described ones could produce higher density values. The results of these experiments are presented in table 5.1 and represent the APCD values at a vehicle travel distance of 8 m, for several combinations of sensor height and incidence. Confirming the results obtained above, this table shows that angles between 0.6° and 1.2° produce the higher densities for radii of 0.4 and 0.5 m respectively, and the 0° incidence angle for heights between 0.3 and 0.4 m also produces high results close to these maximum values.

5.3.3 Influence of Curb Topology in Point Cloud Density

In real world situation, particularly in more rural environments boundaries of the road may not be perfectly vertical as was assumed in all previous experiments. So it is important to characterize the behaviour of the laser scan projections and the resulting accumulation in other road boundary types. For this, several simulations were performed using a Type II (Figure 5.14) road profile with different angles of the curb (in red) relative to the road surface (white), where 0° represents no curb (coincident with the road plane) and 90° represents a vertical curb, measuring APCD for each case. Once again, the simulation was performed considering the vehicle moving at 50 km/h, a curb with a vertical hight of 0.15 m, and 0.2 m for radius of the filtering circle. Additionally the APCD values where measured for two zones, one on each side of the curb. Figure 5.22 refers to the curb on the right side, 1.5 m from the vehicle, and Figure 5.23 refers to the one on the curb on the left side, 4.5 m from the vehicle.

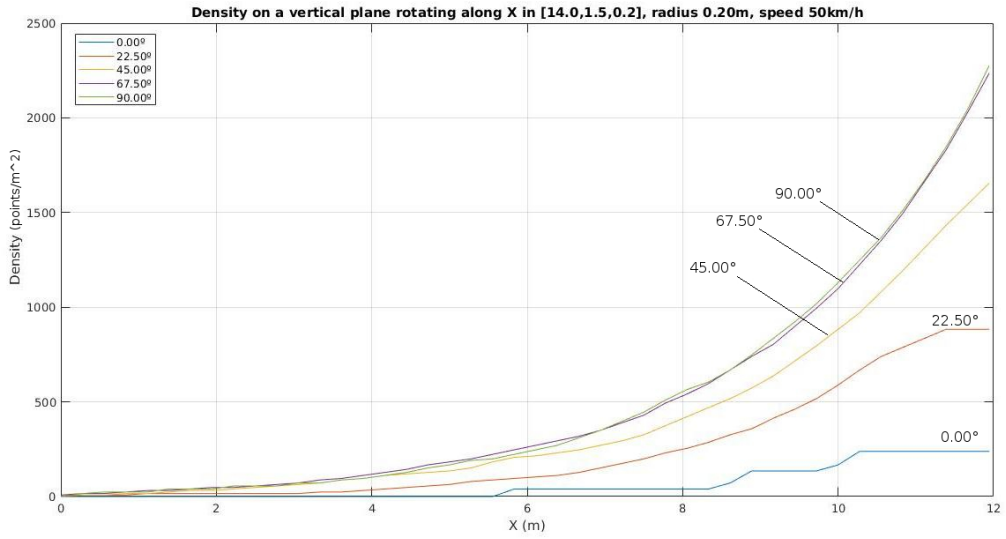


Figure 5.22: Influence of the angle of the right curb surface in a Type II road profile in APCD, measured with a filtering circle of radius 0.2 m.

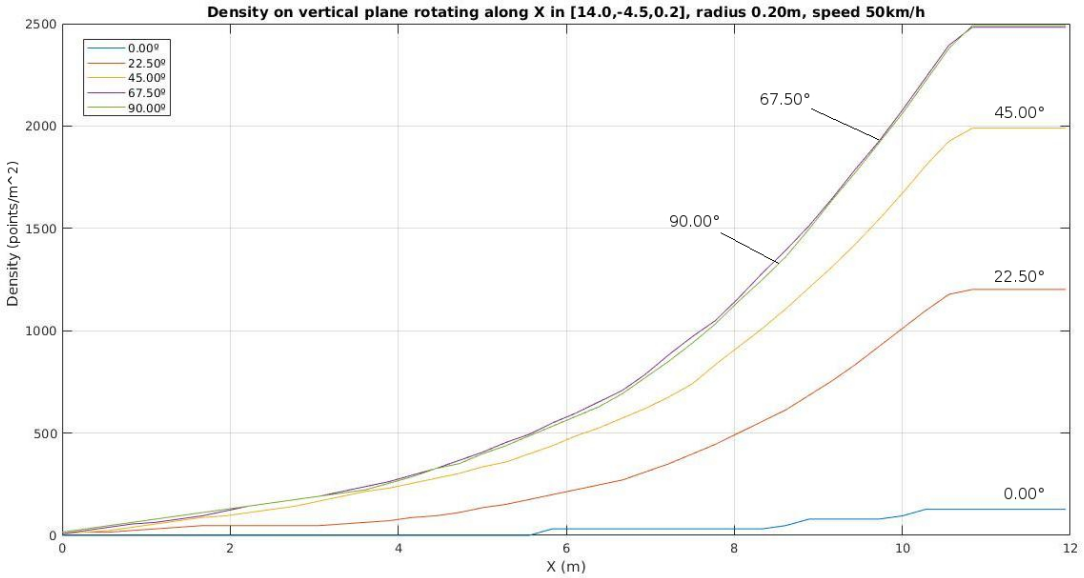


Figure 5.23: Influence of the angle of the left curb surface in a Type II road profile in APCD, measured with a filtering circle of radius 0.2 m.

By analysing the two graphs we observe that they present similar behaviours relative to the nature of the curb and the final APCD values, where in both situations higher curb angles, produce greater APCD values, angle from 67.5° to 90° presenting almost identical results, while lower angles result in very low density values. However the accumulation rate is much greater in the left curb, with almost doubling the APCD values for all angles relative to the accumulation on the right side; though, when the vehicle is closer to the accumulation point, the laser scans no longer hit the curb, capping the accumulation.

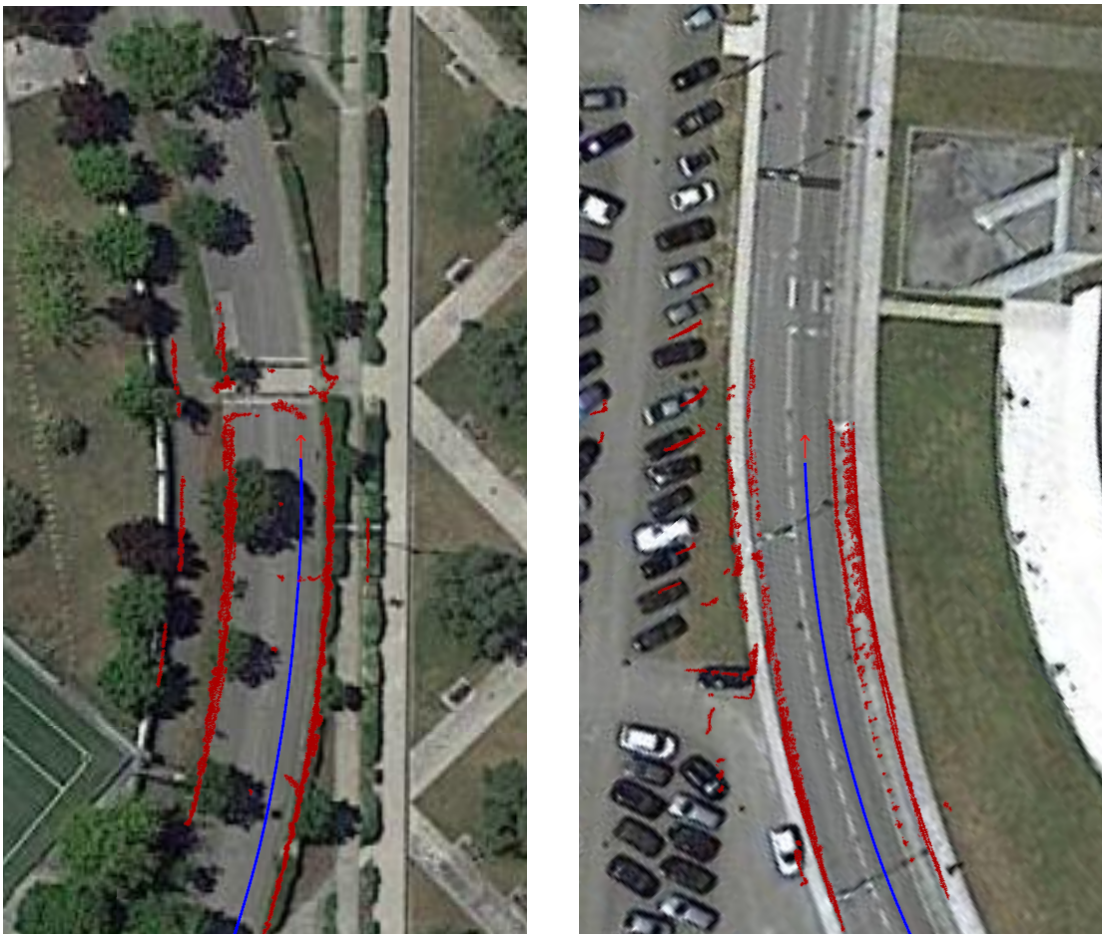
These results demonstrate the need to adapt the algorithm to the geometry of the curb, since low angle curbs could prove to be a challenge to this method because they produce low density values that can be overlooked as road plane by the algorithm.

5.4 Results in Road Edge Detection

The next step was to gather the knowledge from the previous experiments, implement them in the parameter for the APCD filter and test the algorithm in a real-world scenario. The road reconstruction module starts detecting features on the road at about 14-15 m ahead of the vehicle, as can be seen in Figure 5.11, but for vehicle navigation it is important to have filtered information about the road boundaries with enough advance to have time to make the navigation decisions. Considering this factor, it was decided to start the filtering on about 10m front the car, this means about 4-5m of accumulation before filtering and which translates to a APCD of about $180 points/m^2$ on the right curb and $300 points/m^2$ on the left curb according to the data from the graphs in Figures 5.22 and 5.23, considering a Type I curb (perpendicular to the road). Averaging these two values gives an APCD of $240 points/m^2$ that if using the ideal filtering radius of 0.2m translates into 30 neighbouring points as the parameter for the *RadiusOutlierRemoval*

filter. But this value was theoretically determined considering the use of the four sensor scan planes, however, it was experimentally verified that the top most scan plane does not contribute beneficially to the road reconstruction since its laser beams hit too far from the vehicle and the smallest changes to the vehicle's orientation causes loss of measurements, only adding noise to the point cloud, and for this reason it was removed from the reconstruction. This, coupled with the fact that in real world situations, there are factors that induce noise in the measurements, either due to environmental conditions or even as a property of the measuring hardware, which cause loss of data. For this reason the implemented value for the minimum number of neighbouring points was 15 neighbours, to compensate for the noise in the sensor readings.

The following figures present a satellite view of the area surrounding the vehicle, overlapped with the results of the road boundary detection algorithm represented by the point cloud in red, during various situations encountered during navigation, such as turns (Figure 5.24), straight lines (Figure 5.25), a roundabout (Figure 5.26) and an area where the curb is obstructed by parked cars (Figure 5.27).



(a) Road boundary point cloud while performing a anticlockwise turn and approaching a small ramp.

(b) Road boundary point cloud while performing a clockwise turn.

Figure 5.24: Road boundary detection during turns.

(a) Road boundary point cloud during a straight road, after going through a small ramp.

(b) Road boundary point cloud during a straight road with many intersections.

Figure 5.25: Road boundary detection during straight sections of road.

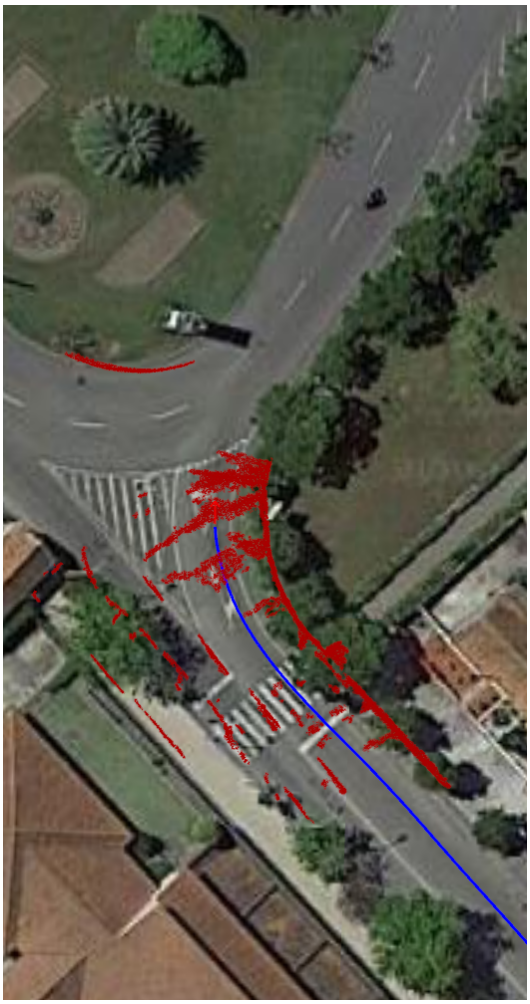
Figure 5.26: Road boundary point cloud during with the vehicle contouring a section of a roundabout.

Figure 5.27: Road boundary point cloud in a streach of road where the rigth side curb is obstruced by parked cars.

5.5 Limitations of the Method

All the results presented in Section 5.4 were obtained using a static algorithm, meaning the filtering parameters do not change with the vehicle's motion parameters, like speed and orientation, and as such it can not adapt to the rapid changes in density caused by these different situations. Evidences of this can be seen in the images from the previous section (5.4), where sometimes random clutters of points appear in the filtered cloud as a result of these situations. However, during motion, because the changes are instantaneous and temporary, these clusters are generally small and a method could be developed to eliminate them; the problem arises when the vehicle slows down during a significant amount of time, for example when reaching an intersection (Figure 5.28): this causes the local density of points to greatly increase, as evidenced in 5.19, deceiving the algorithm. This effect is more pronounced the greater the velocity difference between the vehicle and the one implemented in the algorithm, and the extreme case

is when the vehicle completely stops, this causes heavy concentration of points on the road plane (Figure 5.28a), followed by a halt in road boundary detection (Figure 5.28b) since with no movement there is no accumulation and, therefore, no information about the reconstructed road.



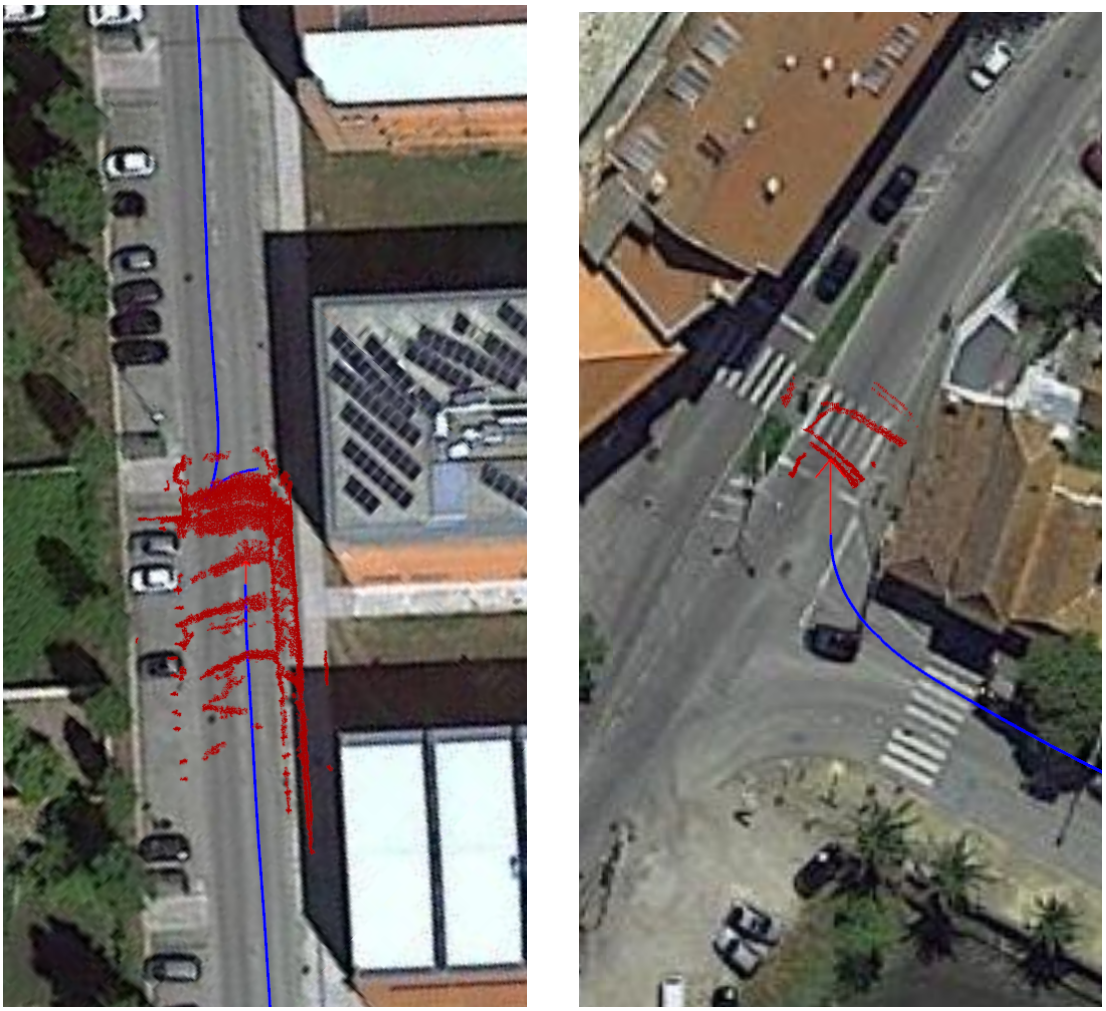
(a) Road boundary point cloud during heavy slow down of the vehicle approaching a roundabout.

(b) Road boundary point cloud during heavy slow down of the vehicle approaching an intersection.

Figure 5.28: Road boundary detection when the vehicle is approaching and intersection.

One other issue with using static parameter or the filtering algorithm is caused by the difference in APCD in both sides of the road. Because the left and right curbs are at a different distance from the sensor, the APCD values in these curb planes will also be significantly different, as evidenced by the results of the simulation presented in figures 5.23 and 5.22. These results show that for the same accumulation distance the density values measured in the left side of the road are almost double the ones measured on the right side. In practical terms this means that the boundary detection on right side will happen closer to the car than the one on the left side, since it need to accumulate during a greater distance to reach the same density levels, causing the misalignment of

the curb detection seen in almost all the figures from Section 5.4.



(a) Road boundary point cloud during heavy slowing.

(b) Road boundary point cloud after the vehicle has been stoppped for short period of time.

Figure 5.29: Road boundary detection during heavy slowing and stopping of the vehicle.

Other external factors can also influence the characteristics of the point cloud such as other vehicle closely in front of the car, that may produce vertical zones of great density in front of the car akin to side walk curbs, causing false positive detection by the algorithm.

Lastly, by observing all the images from this and the previous sections, we see that the detections and representation of the road boundary point cloud occurs closer to the vehicle than expected, suggesting that other optimizations to the method coupled with the ones mentioned in Section 4.3.2 have to be put in place to increase the rate of detection.

5.6 Dynamic Parameters for the Filtering Method

As a way to improve the road detection algorithm and eliminate some of this issues caused by the use of static parameters in the point cloud filter, a methodology was developed to incorporate information about the vehicle velocity as a influencing factor in the parameters of the *RadiusOutlierRemoval* filter (radius of the filter sphere, R, and number of neighbour points, N_{total}).

As such, two different approaches where taken, in order to evaluate the best method; One where the radius value was fixed to its ideal value of 0.2 m (according to the performed theoretical simulations in Section 5.4) and the number of neighbour points was determined through the theoretical behaviour of the influence of vehicle velocity in APCD. The other approach consisted in using the knowledge about the relationship between the velocity and the distance between successive accumulations, to dynamically determine the ideal radius, and then through experimentation find the best correlation between the radius and the number of neighbour points.

For the first method, a simulation was performed using the same setup as the ones described in Section 5.3 (figure 5.15), where the APCD values for a point on the left curb where determined based on vehicle velocity for a accumulation distance of 4 m. Figure 5.30 represents the obtained simulation results for the relationship between APCD and vehicle velocity.

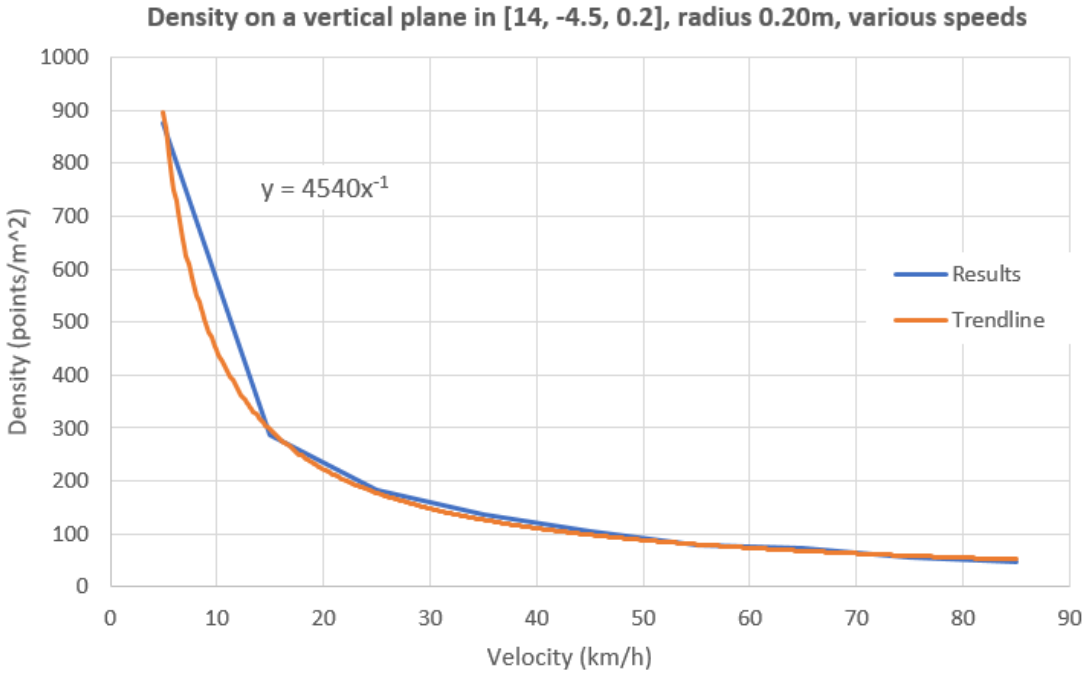


Figure 5.30: Influence of vehicle velocity in APCD, for an accumulation distance of 4 m using a filtering circle radius of 0.2 m.

This simulation allows to obtain the relationship between APCD and velocity ($v_{car}[m/s]$) which is defined by the equation: $APCD = 4540/v_{car}$, obtained through the graph power trend line. Once again this relationship represents a theoretical prediction and isn't af-

fected by any noise nor loss of data, so to use this in the real world filter a factor of 1/3 was applied to the formula to compensate for these factors. Thus, the implemented relation used in this method is $APCD = 1513/v_{car}$. In order to apply it to the number of neighbours has to be deduced from this relationship. Using the APCD formula described in 4.6, and noting that in this method the radius is assumed to be fixed ($R = 0.2$ m) we can easily obtain the number of neighbours (equation 5.2).

$$N_{total} = 1513/v_{car} \cdot \pi * 0.2^2 \tag{5.2}$$

However, due to the exponential behaviour of this function some limits where put into effect to prevent the filter from removing all points during heavy breaking. As such a maximum for $N_{total} = 42$ was established for low speeds and a minimum of $N_{total} = 6$ for high speeds.

The second method used in the dynamic filtering, was based on the relation between vehicle velocity and the distance between successive laser scans. This relationship is given by equation 4.2 with v_{car} in m/s. Since the frequency of the sensor is static and equal to 50 Hz, equation simply becomes: $\delta x = v_{car}/50$. The reason for determining this displacement is that it provides information about the distance between the laser scans on the ground plane, and by applying a filter with a radius, $R \leq \delta x$, means that, on the ground plane, only one scan at a time is being "caught" inside the filter, this way, maximizing the difference between the densities on the ground and curb planes. After defining the radius the next step was to determine the relationship between it and the number of neighbours,N_{total}, that produces the best results. This was done experimentally, iterating over several functions until a solution was reached that was able to correctly extract road boundaries from the point cloud. The final relationships between the three parameters are described in equation 5.3, noting that the radius function was slightly modified to produce better results.

$$\begin{aligned} R &= v_{car}/50 + v_{car}/350 \\ N_{total} &= \lfloor 87.5 * R \rfloor \end{aligned} \tag{5.3}$$

Similarly to the first method a cap was put in place to prevent the filter from removing all points when the vehicle is travelling very low speeds, this way the method has a minimum cap at $R = 0.09$ and $N_{total} = 6$.

Figures 5.31 through 5.34 represent comparisons between the filtered cloud obtained by each of the two methods, in multiple situations, like unobstructed straight segments of road 5.31 and heavy breaking before an intersection 5.34.

Additionally, figures 5.32, 5.33 and 5.34, where obtained in the same instant as the figures 5.25b, 5.28a and 5.28b respectively, so that they can be compared with filtered cloud from the static road boundary detection method.

(a) Fixed radius method - $R = 0.20$ m; $N_{total} = 23$ points.

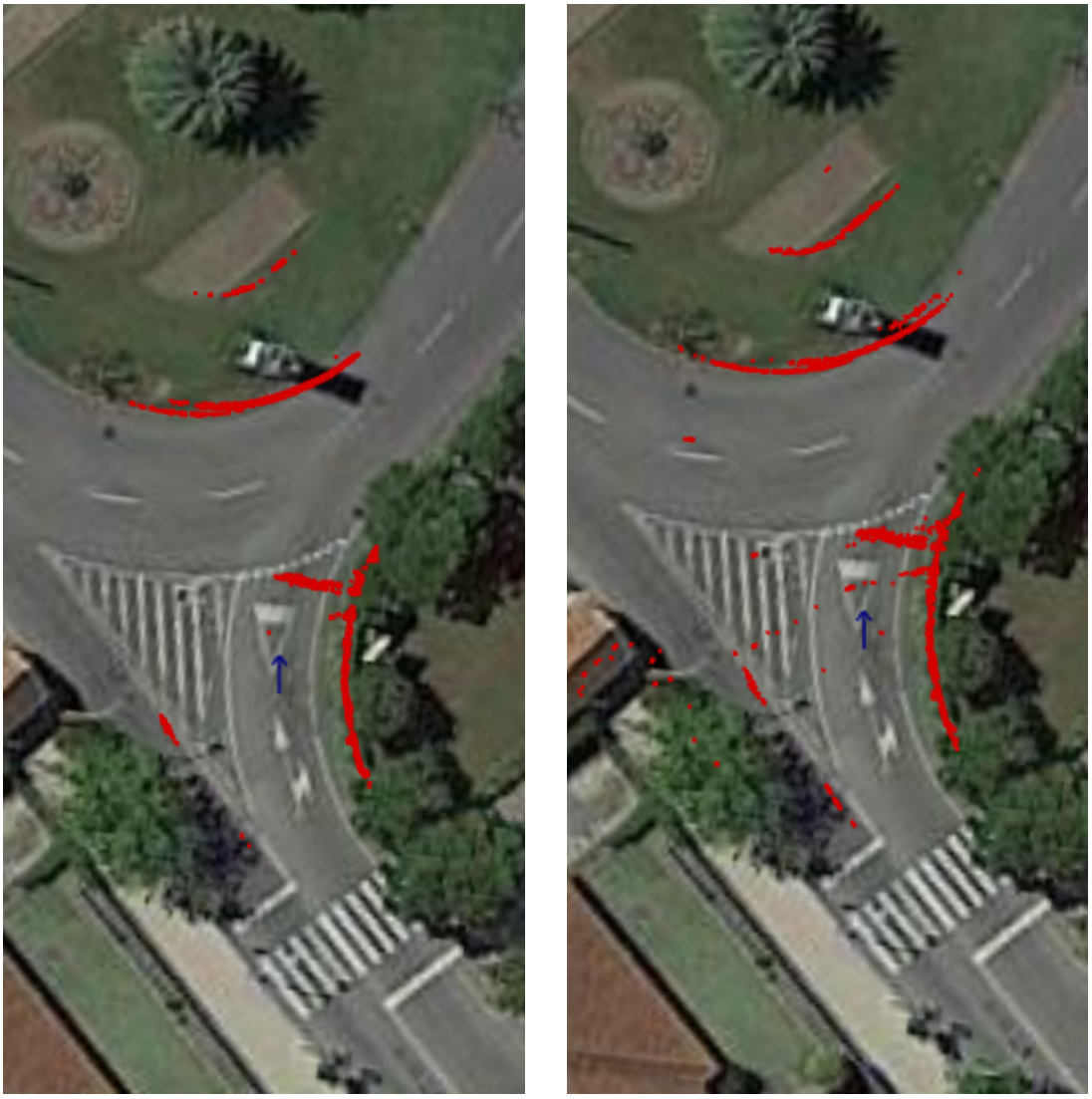
(b) Variable Radius method - $R = 0.17$ m; $N_{total} = 14$ points.

Figure 5.31: Comparison between the two dynamic filtering methods, on a straight section of road, measured in the same instance in time - $v_{car} = 27.3km/h$.

(a) Fixed radius method - $R = 0.20$ m; $N_{total} = 28$ points.

(b) Variable Radius method - $R = 0.145$ m; $N_{total} = 12$ points.

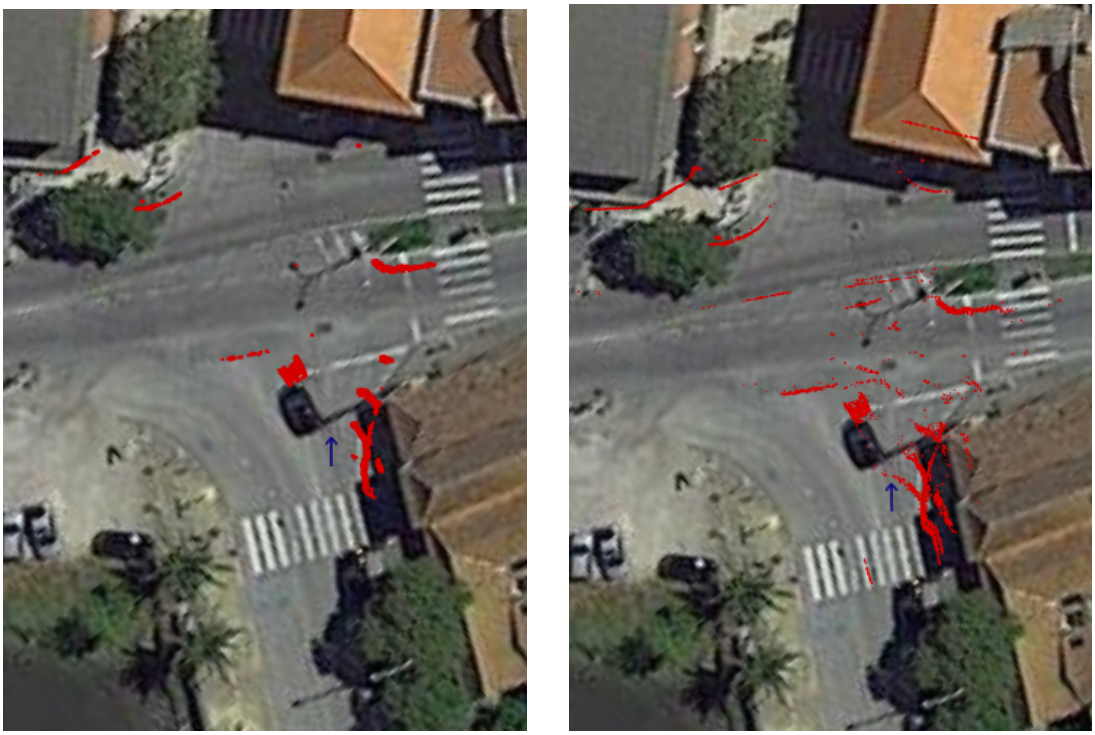
Figure 5.32: Comparison between the two dynamic filtering methods, on a straight section of road with many intersections, measured in the same instance in time - $v_{car} = 23.0 km/h$.



(a) Fixed radius method - $R = 0.20$ m; $N_{total} = 39$ points.

(b) Variable Radius method - $R = 0.105$ m; $N_{total} = 8$ points.

Figure 5.33: Comparison between the two dynamic filtering methods, during slowdown when reaching a roundabout, measured in the same instance in time - $v_{car} = 16.5km/h$.



(a) Fixed radius method - $R = 0.20$ m; $N_{total} = 42$ points.

(b) Variable Radius method - $R = 0.09$ m; $N_{total} = 6$ points.

Figure 5.34: Comparison between the two dynamic filtering methods, during heavy slowdown reaching an intersection, measured in the same instance in time - $v_{car} = 7.7km/h$.

By analysing the results from these images we see that both of the methods presented for dynamic filtering produce very good results through the various speeds along the travelled path, and for the most part, correctly identify the road curbs with little noise in the filtered point cloud.

When comparing these results to the ones obtained by the static algorithm, there is a significant improvement in the detection at low speeds, with the methods being able to identify the curbs despite the excessive accumulation. Moreover, the dynamic methods also perform well in situations of higher speeds matching the accuracy in curb detection of the static method.

Comparing the performance between the methods, despite their different nature both perform similarly at regular and low speeds, with the fixed radius method generally producing less noise in the filtered cloud, but the variable radius method shows to be able to better identify the curb surfaces further form the vehicle and is better at identifying smaller obstacles such as road delineators.

Chapter 6

Conclusions and future work

6.1 Conclusions

The problem for road boundary detection is crucial for autonomous driving applications, given the need to identify the road area for the navigation systems to base their decisions upon. As such, along the years this problem as been addressed by many researchers leading to the development of multiple approaches through multiple sensing techniques. The work detailed in this document presents a new methodology for road boundary detection to be used in a moving vehicle equipped with a four scan plane LIDAR sensor positioned close to the road plane.

To be able to extract information about the road limits, a system for reconstructing the perceived section of the road ahead of the vehicle had to be developed. For this it was important to correctly locate the vehicle, both in terms of its position relative to the world and its orientation relative to the ground at every instant. Thus, a global localization module was constructed based on the *Novatel SPAN-IGM-A1* present in the *AtlasCar2* and an orientation module based on the system created by *P. Salvado* [14] was also implemented to allow to correctly define the sensor in all 6 degrees of freedom, and this way accurately reference the readings relative to a fixed *world* frame. To incorporate these two modules on the vehicle, the reference frame structure (transformation tree) of the *AtlasCar2* had to be rebuilt so that the information provided by these modules could be correctly used to build the transformations between each frame.

Having the car and its sensors correctly defined allowed to perform a reconstruction of the section of road in front of the vehicle, through successive accumulation of the laser scans. The resulting point cloud proved to be very reliable in representing the road and its surroundings, correctly recreating it even in adverse situations caused by changes in the orientation of the car. Additionally, due to the unique placement of this type of sensor, the generated reconstruction showed some interesting characteristics regarding the density of its points.

Studying the projections resulting from the intersection of the laser planes with the road and the curb allowed to develop a method to filter the reconstruction point cloud and extract the road features corresponding to the curb and other vertical obstacles. Given the great number of variables that affect the laser projections, a simulator was built to evaluate the effect of these variables in *Accumulated Point Cloud Density* (APCD). This simulator provided accurate numerical results that were used to determine the ideal position and orientation of the sensor and the parameters for the filtering algorithm

based on the results of the expected values for APCD in various curb configurations and topologies.

After implementing the, parameters for the filtering algorithm (radius and number of neighbour points) determined by the results obtained from the simulations, real-world tests where performed to evaluate the method. The results show that the method performs well, in most cases successfully identifying the road limits in each side of the car, even in situations where the curb and side-walk are obstructed by other parked vehicles. However in some situations the filtered point cloud still presents some noise, caused by fluctuations in the orientation of the car that cause slight changes in local density. These affect the algorithm causing some false-positives to be detected. Additionally these results reveal that the algorithm is still highly dependent on the vehicle velocity, since at low speeds the road limits are not detected due to the high point density resulting from the accumulation. To solve this, two methods where developed to allow the parameters of the filter to dynamically change with vehicle velocity. Both of the methods performed well, solving the main problem of boundary detection during excessive accumulation, with each of them having slight advantages that can further be explored.

Overall, despite some limitations, the road boundary detection goal has been successfully achieved, taking the *AtlasCar2* project one step further in the great challenge that is autonomous driving.

6.2 Future Work

The new method for road boundary detection created in this work still leaves much room for exploration. Some future works may include refining the dynamic behaviour of the algorithm to adapt the filtering parameters to the different ranges of *Accumulated Point Cloud Density* (APCD) caused by changes in vehicle velocity and orientation and accommodate other variables such as detecting the side of the curb to relative to the vehicle to be filtered, and this way increasing the robustness of the method. Other lesser problems inherent to the method should also be tackled, like for example situations where the reconstruction detects a moving vehicle across the path of the car, but due to the accumulation buffer, the obstacle (represented in the filtered cloud) remains for some time even after the other vehicle has passed.

Also, improvements to the algorithm should be done to increase the processing speed of the filtering method, and this way decrease the time between the acquisition of data by the LIDAR and the road boundary extraction. Some of these improvements can include changes to the *Radius Outlier Removal* filter to better suit the needs of this project, by filtering only the newly acquired data instead of the whole reconstructing cloud each time. It is also important to develop a mechanism to quantitatively evaluate the performance of the road extraction algorithm. This may involve building a *ground truth* of the road features in a predefined area so this data can be compared to the one produced by the method and this way quantify its accuracy and precision other than only visually.

Moreover there are still many situations that the vehicle can encounter on the road that where not explored in this work, such as negative curbs present in Type III road configurations (Figure 5.14), and the detection of other obstacles in the road plane like small holes or ramps.

References

[1] James Anderson, Nidhi Kalra, Karlyn Stanley, Paul Sorensen, Constantine Samaras, and Oluwatobi Oluwatola. *Autonomous Vehicle Technology: A Guide for Policymakers*. 2016.

[2] LARlabs. *ATLAS Project*. URL: `http://atlas.web.ua.pt` (visited on 05/02/2018).

[3] Rui Filipe Cabral de Azevedo. "Sensor Fusion of LASER and Vision in Active Pedestrian Detection". Master's Thesis. Universidade de Aveiro, 2014.

[4] Ricardo Morais. "Parametrização de Algoritmos para Deteção de Estrada a Bordo do ATLASCAR". Master's Thesis. Universidade de Aveiro, 2014.

[5] Sérgio António Matos Pinho. "Caixa Automática e Manobras Especiais no ATLASCAR". Master's Thesis. Universidade de Aveiro, 2014.

[6] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. "Recent progress in road and lane detection: A survey". In: *Machine Vision and Applications* 25.3 (2014), pp. 727–745.

[7] Sheng Xu, Ruisheng Wang, and Han Zheng. "Road Curb Extraction from Mobile LiDAR Point Clouds Sheng Xu, Ruisheng Wang, Han Zheng 1". In: 55.2 (2017), pp. 1–28.

[8] Carlos Cabo, Antero Kukko, Silverio García-Cortés, Harri Kaartinen, Juha Hyyppä, and Celestino Ordoñez. "An algorithm for automatic road asphalt edge delineation from mobile laser scanner data using the line clouds concept". In: *Remote Sensing* 8.9 (2016).

[9] Wende Zhang. "LIDAR-based road and road-edge detection". In: *IEEE Intelligent Vehicles Symposium, Proceedings* (2010), pp. 845–848.

[10] Bisheng Yang, Lina Fang, and Jonathan Li. "Semi-automated extraction and delineation of 3D roads of street scene from mobile laser scanning point clouds". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 79 (2013), pp. 80–93.

[11] Matti Lehtomäki, Anttoni Jaakkola, Juha Hyyppä, Antero Kukko, and Harri Kaartinen. "Detection of vertical pole-like objects in a road environment using vehicle-based laser scanning data". In: *Remote Sensing* 2.3 (2010), pp. 641–664.

[12] *Riegel VMX-450*. Compact Mobile Laser Scanning System. Riegel. Mar. 2015.

[13] Yang Gao, Ruofei Zhong, Tao Tang, Liuzhao Wang, and Xianlin Liu. "Automatic extractions of pavement markings on the streets from point cloud data of mobile LiDAR". In: *Measurement Science and Technology* 28.8 (2017), p. 085203.

[14] Pedro Miguel Godinho Salvado. "Reconstrução Dinâmica de Mapa Local para o AtlasCar". Master's Thesis. Universidade de Aveiro, 2012, p. 64.

[15] Armindo Silva. *Inclinómetro planar de precisão para o Atlascar-2*. Tech. rep. Universidade de Aveiro, 2017, p. 21.

[16] *LD-MRS400001S01 | LD-MRS*. 1052960. Detection and Ranging Solutions. SICK. Feb. 2018.

[17] *DT20-P244B | DT20 Hi*. 1040406. Displacement Measurement Sensors. SICK. May 2018.

[18] *SPAN® SPAN-IGM-A1*. SPAN MEMS Technology Integrated with Novatel's powerful OEM615 Reciever. Novatel. May 2016.

[19] ROS. *About ROS*. URL: `http://www.ros.org/about-ros/` (visited on 05/14/2018).

[20] Morgan Quigley, Eric Berger, and Andrew Y Ng. "STAIR : Hardware and Software Architecture". In: *AAAI 2007 Robotics Workshop, Vancouver, BC* (2007), pp. 31–37.

[21] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Mg. "ROS: an open-source Robot Operating System". In: *Icra* 3 (2009), p. 5. arXiv: `1106.4561`.

[22] Dilip Kumar J. *Using see3cam_10cug_c with ROS*. URL: `https://www.e-consystems.com/Articles/Camera/see3cam-10cug-c-with-ros-robot-operating-system.asp` (visited on 05/07/2018).

[23] Radu Bogdan Rusu and Steve Cousins. "3D is here: Point Cloud Library (PCL)". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011.

[24] PCL. *About*. URL: `http://pointclouds.org/about/#open` (visited on 05/14/2018).

[25] Gonçalo Alberto Ramos Carpinteiro. "Medição de Orientações Relativas de um Veículo em Movimento". Master's Thesis. Universidade de Aveiro, 2014.

[26] Michael Ferguson and Adam Stambler. *rosserial_arduino*. URL: `http://wiki.ros.org/rosserial_arduino` (visited on 05/15/2018).

[27] M. M. R. Mostafa, J. Hutton, B. Reid, and R. Hill. "GPS / IMU products ? the Applanix approach". In: *Photogrammetric Week 01* (2001), pp. 63–83.

[28] Gianluca Falco, Marco Pini, and Gianluca Marucco. "Loose and tight GNSS/INS integrations: Comparison of performance assessed in real Urban scenarios". In: *Sensors (Switzerland)* 17.2 (2017).

[29] José Correia. "Unidade de Perceção Visuale de profundidade para o ATLASCAR2". Master's Thesis. Universidade de Aveiro, 2017, pp. 1–98.

[30] David Tiago Vieira da Silva. "Multisensor Calibration and Data Fusion Using LIDAR and Vision". Master's Thesis. Universidade de Aveiro, 2016, p. 107.

[31] Vijay Pradeep. *laser_assembler*. URL: `http://wiki.ros.org/laser_assembler` (visited on 05/16/2018).

[32] Gaurav Pandey, James R. McBride, and Ryan M. Eustice. "Ford Campus vision and lidar data set". In: *International Journal of Robotics Research* 30.13 (2011), pp. 1543–1552.

[33] J Levinson, J Askeland, J Becker ... (IV), 2011 IEEE, and undefined 2011. "Towards fully autonomous driving: Systems and algorithms". In: *Ieeexplore.Ieee.Org* Iv (2011).

[34] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the KITTI vision benchmark suite". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2012), pp. 3354–3361. arXiv: 1612.07695.

[35] BerkleyDeepDrive. *3D Object Detection based on Lidar and Camera Fusion*. URL: https://deepdrive.berkeley.edu/project/3d-object-detection-based-lidar-and-camera-fusion (visited on 05/20/2018).

[36] Radu Bogdan Rusu. *pcl::RadiusOutlierRemoval< PointT > Class Template Reference*. URL: http://docs.pointclouds.org/1.8.1/classpcl_1_1_radius_outlier_removal.html#details (visited on 05/23/2018).

[37] PCL. *Removing outliers using a Conditional or RadiusOutlier removal*. URL: http://pointclouds.org/documentation/tutorials/remove_outliers.php (visited on 05/23/2018).

[38] PCL. *Downsampling a PointCloud using a VoxelGrid filter*. URL: http://pointclouds.org/documentation/tutorials/voxel_grid.php (visited on 05/24/2018).

[39] Marc Alban. *mapviz*. URL: http://wiki.ros.org/mapviz (visited on 05/30/2018).

Appendix A

Instructions to install and run the package

A.1 Masters-Thesis_ATLASCAR2

ROS packages to establish the position and orientation of the AtlasCar2 based on inclinometer and GPS data, perform the reconstruction of the road perceived by the SICK LD-MRS sensor and from that extract the road boundaries.

A.1.1 Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

A.1.2 Prerequisites and installing

This repository contains only the code developed in the context of the thesis. In order to run the code some additional packages need to be installed.

Install the prerequisites for these additional packages:

```
%Flycap - camera drivers
sudo apt-get install libraw1394-11 libgtkmm-2.4-1v5 libglademm-2.4-1
v5 libgtkglextmm-x11-1.2-dev libgtkglextmm-x11-1.2 libusb-1.0-0

%GPS Libraries
sudo apt-get install ros-kinetic-gps-common

%Arduino Libraries
sudo apt-get install ros-kinetic-rosserial-python

%Additional Libraries
sudo apt-get install libopenni2-dev
sudo apt-get install libgeos++-dev
```

After this, download the repository containing the code.

```
cd $ros_workspace/
git clone https://github.com/TiagoSMarques/Masters-Thesis_ATLASCAR2.
git
mv -v Masters-Thesis_ATLASCAR2/* src/ && rmdir Masters-
Thesis_ATLASCAR2
```

The dependencies for the code need to be compiled first, so remove the developed packages (free_space_detection,orientation_module and road_reconstruction) from the src folder, and compile the code:

```
mv -t . src/free_space_detection src/orientation_module src/
road_reconstruction
catkin_make
```

Since the compilation order is not defined by the user, the compilations could eventually fail because some packages may depend on others that have not been compiled yet. If this occurs move the package in which the error occurred out of the src/ folder and recompile with the same command. In the same way for "permission denied" errors in the configuration files run the following code, replacing the name with the name of the file in question:

```
cd directory_of_the_file/
chmod +x name.cfg
```

Once the compilation has finished re-add the moved packages to the src/ folder and recompile.

A.1.3 Setup of the hardware and launching the nodes

In the *AtlasCar2*, connect the Arduino and GPS (2 ports) USB's, and the Ethernet port (with its IP already configured), and launch the nodes for the drivers of the lasers:

```
roslaunch free_space_detection drivers.launch
```

Then launch the node to run the inclinometer module, replacing "port_name" with the device name refering to the Arduino USB:

```
rosrun rosserial_python serial_node.py /dev/port_name
```

Finally launch the road reconstruction system, which launches all the nodes needed, and presents the point cloud data in Rviz for visualization:

```
roslaunch road_reconstruction road_rec.launch
```

Additionally you may need to manually calibrate the inclinometer module, for this, after running the previous command go to the white pvc box containing the Arduino and press the button located on the inside of this box.

A.1.4 Additional notes

To view the representation of the filtered cloud in a satellite view, refer to the mapviz package:

- SWRI Mapviz package - `https://github.com/swri-robotics/mapviz`

A.1.5 Built For the ROS environment

- ROS The Robot Operating System -`http://www.ros.org/about-ros`

Author: Tiago Simões Marques - `https://github.com/TiagoSMarques`