Universidade de Aveiro Departamento de Engenharia Mecânica
2020

Rui Pedro Leite Carvalho Costa

Deteção e seguimento de alvos múltiplos usando LIDAR e obstáculos de velocidade para definição em tempo real das zonas de colisão

Multi target tracking and detection using LIDAR and Velocity Obstacles for real time definition of collision zones

Universidade de Aveiro Departamento de Engenharia Mecânica
2020

Rui Pedro Leite Carvalho Costa

Deteção e seguimento de alvos múltiplos usando LIDAR e obstáculos de velocidade para definição em tempo real das zonas de colisão

Multi target tracking and detection using LIDAR and Velocity Obstacles for real time definition of collision zones

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 Jorge Manuel Soares de Almeida, Professor Auxiliar Convidado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Este trabalho foi desenvolvido com recursos do IEETA - Instituto de Engenharia Eletrónica e Informática de Aveiro, no âmbito do Projeto UIDB/00127/2020, e usou fundos comparticipados pelo DEM/TEMA - Centro de Tecnologia Mecânica e Automação no contexto do Projeto UIDB/00481/2020, ambos financiados pela FCT - Fundação para a Ciência e Tecnologia.

o júri / the jury

presidente / president

Prof. Doutor João Alexandre Dias de Oliveira
Professor Auxiliar da Universidade de Aveiro

Prof. Doutor Hugo Filipe Costelha de Castro
Professor Adjunto do *Politécnico de Leiria, Escola Superior de Tecnologia e Gestão*

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

agradecimentos / acknowledgements

Primeiramente, gostaria de agradecer ao Prof. Doutor Vítor Santos por todo o seu auxílio e acompanhamento durante este período. Pela troca de ideias e pensamento "fora da caixa" que permitiu ir sempre mais além do problema em questão. Acima de tudo agradeço pela paixão contagiante pela robótica que transmitiu.
Ao Prof. Doutor Jorge Almeida pelo auxílio e feedback prestado.
À minha familia, pais, irmão e avó por todo o amor, carinho e motivação durante esta caminhada.
Aos meus amigos, por todos os bons momentos vividos. Ao pessoal do LAR, Diogo e Rúben, pela ajuda constante mesmo estando separados.
A ti, Sofia, por me ajudares a mostrar que consigo e sobretudo pela paciência durante este período.

keywords LIDAR, Obstacle detection, Collision detection, Velocity Obstacles, ROS framework

abstract The implementation of autonomous vehicles on public roads faces many challenges, being the most important ensuring the safety for everyone within that environment. In order for these unmanned machines to be applied in our every-day lives, they first need to be able to correctly identify all possible threats, to then act accordingly. As part of the Atlas project, this dissertation main goal is to determine these threats based on the velocity and dimensions of the detected obstacles. The detection is made using two 2D laser sensors SICK LMS151. These sensors, strategically placed in the front bumper of the ATLASCAR2, allow to perceive its surrounding in an almost 360° (except in the back near the trunk). A detection and tracking algorithm, previously developed at Universidade de Aveiro, was used and refurbished to determine the obstacles' velocity used to verify if the ATLASCAR2 is in a collision path.

Initially, the detection and tracking algorithm was tested using a static LIDAR to retrieved data, resulting in good performances. However the deployment in dynamic environments led to a study of the influence of the ATLASCAR2 ego motion, specifically while it is turning, on the perceived obstacle velocity. The LIDAR's moving coordinate frame imposes an apparent velocity on the obstacles making the tracking algorithm's data not as reliable. From this study resulted also a ROS application which allows to visualize the short term path of the ATLASCAR2.

palavras-chave

LIDAR, Deteção de obstáculos, Deteção de colisões, Obstáculos de Velocidade, Estrutura ROS

resumo

A implementação de veículos autónomos nas estradas enfrenta inúmeros desafios, sendo a segurança de todos os intervenientes o mais importante. Com o intuito de introduzir estas máquinas autónomas no nosso quotidiano, estas devem primeiramente ser capazes de detetar corretamente todas as possíveis ameaças para depois agir de acordo. No âmbito do projeto Atlas, esta dissertação tem como objetivo determinar estas ameaças baseando-se na velocidade dos obstáculos detetados. Esta deteção é realizada através de dois sensores laser 2D SICK LMS151. Estes sensores, posicionados estrategicamente na parte da frente do parachoques do ATLASCAR2, permite uma perceção de aproximadamente 360° em volta do carro (com a exceção da parte traseira do veículo). Um algoritmo de deteção e seguimento, já desenvolvido na Universidade de Aveiro, foi usado e adaptado para determinar a velocidade de obstáculos e para verificar se o ATLASCAR2 se encontrava num percurso que poderia resultar numa colisão.
Inicialmente, o algoritmo de deteção e seguimento foi testado usando o sensor LIDAR estático para retirar dados, resultando numa boa performance. No entanto, a utilização do mesmo em ambientes dinâmicos deu origem a um estudo para avaliar a influência do movimento próprio do ATLASCAR2, com foco na situação em que se encontra a realizar uma curva, na velocidade de um obstáculo detetado. O referencial em movimento do LIDAR impõe uma velocidade aparente nos obstáculos, fazendo com que os dados recolhidos pelo algoritmo não sejam de confiança. Deste estudo resultou uma aplicação ROS, esta permite a vizualisação do movimento (durante um curto espaço de tempo) do ATLASCAR2.

Contents

List of Tables

Intentionally blank page.

List of Figures

Intentionally blank page.

Acronyms

ADAS Advanced Driver-Assistance System.

CC Colision Cone.

ETC Expected Time of Colision.

GNN Global Nearest Neighbour.

GPS Global Position System.

IMU Inertial Measuring Units.

LAR Robotics and Automation Laboratory.

MTT Multi Target Tracking.

ROS Robotic Operating System.

TOF Time of Flight.

UPS Uninterruptible Power Supply.

VO Velocity Obstacles.

Chapter 1

Introduction

One of the present society's goal is to introduce machines into human being's environment to facilitate daily tasks. With that in mind the automotive sector has perhaps the hardest challenge. This industry's ultimate goal is the implementation of fully autonomous vehicles capable of providing their users a safe and comfortable ride, reducing the driver's intervention or even removing the driver. Applications of autonomous vehicles are endless and will certainly make a huge impact in the world, possibly making it safer, cheaper and more efficient. However, with the chaotic and sometimes deadly environment present in today's roads and highways, this technology is still a long way from reaching the highest level of autonomy, defined by SAE International [1].

First of all, an autonomous vehicle must have the ability to perceive its surroundings and identify every element in its neighbourhood. This is a fundamental step and possibly the hardest one, due to the shear amount of raw data gathered and the potential hazard to everyone in the road in case of a miss identification. This raw data collection can be done in several ways, being cameras and LIDAR sensors the most commonly used nowadays. For the purpose of this dissertation, 2D LIDAR sensors will be used for data collection. The raw data must then be filtered for noise reduction and converted into road objects, *i.e.* targets. These targets can be other road vehicles, road signs, pedestrians, among many others.

Once all the targets are identified, the decision making process begins. This second step is responsible to interpret the targets data and identify possible risks for the car to then define the best action to take, in order to avoid or minimize the mentioned risks. The work developed throughout this dissertation is focused in this second step, and studies the possible collision zones between the ATLASCAR2 and its surrounding targets using velocity data. This collision zone data can then be used by a decision algorithm to determine the best collision-free course.

The third and last step of an autonomous vehicle is the ability to autonomously maneuver the vehicle to a safe course.

1.1 ATLAS Project

The study developed in this dissertation is inserted in the ATLAS Project. First started in 2003 in the Robotics and Automation Laboratory (LAR) at the Department of Mechanical Engineering of the University of Aveiro, the ATLAS Project main objective is

to research sensory and active systems to employ in autonomous vehicles. The first hardware developed in this project was the Atlas robot, which competed in the Portuguese national robotics competition.

After many awards and with the knowledge acquired throughout the years, the team decided to take a step further and started working on a full scale prototype car, the ATLASCAR1. The Ford Escort Station Wagon from 1998 was equipped with state of the art sensory equipment to perceive its surroundings to then act accordingly.

A few years later, and after all the research done with the ATLASCAR1, it was time to upgrade the vehicle to a more modern and suitable one. So the ATLASCAR1 was replaced with a Mitsubishi i-MiEV electric car, the ATLASCAR2. This vehicle is equipped with multiple LIDAR sensors, among others, needed for the development of this work.

1.2 Problem Description

With the increasing implementation of autonomous vehicles on public roads, motion planning techniques are of the most importance. These techniques allow autonomous vehicles to determine the best action to take, ensuring the safety of the vehicle itself and its surroundings.

Studies on motion planning are numerous, and may focus on different planning approaches, however, the main goal is common: obtain a collision-free path between two points in space. The environment where the study is applied will hugely influence the approach, and can vary between static and dynamic. A good example of a dynamic environment can be a street or highway with multiple moving vehicles, *i.e.* obstacles. The ATLASCAR2's, as any other car, "workplace" fits in this example, as so the work developed in this dissertation.

Motion planning techniques in dynamic environments is one of the most researched topic in the field of mobile robotics, mainly due to its complexity and applicability, and many approaches were formulated [2]. The study developed in this dissertation focuses on the velocity approach and its main goal is to determine whether there will be a collision or not, and if so where, using the velocity information of the ATLASCAR2 and its colliding obstacle.

Although the work developed in this dissertation does not determine the best path to avoid collision, it is the foundation to determine a collision-free path.

1.3 Objectives

The main goal of this dissertation is to determine whether or not the ATLASCAR2 is in collision course with its surrounding obstacles. In order to achieve this, data will first be gathered by the car's two LIDAR sensors, located near the front headlights, and be put through a target tracking ROS application developed at LAR [3] which reports back the obstacles position and velocity, among other attributes. With the obstacles' and the car's information, a ROS package will be developed which informs the user and subscribing nodes if the ATLASCAR2 is in a collision course. If a collision is imminent, some of its details are also to be calculated, such as the distance to the colliding object and the predicted time to collision.

Since the target tracking ROS application was developed a few years back, a migration to the new version of the ROS environment is needed before it can be applied in this work.

Still with the main goal in mind, a visualisation tool is to be developed to display the short term path of the ATLASCAR2 and graphically better identify the possible collision location. The mentioned tool must rely on another work developed at the university [4], using the car's current velocity and steering wheel angle.

In short, the main objectives of this dissertation are:

- Migrate the required ROS applications developed in the past to the current version of the ROS environment
- Determine the collision zones between the ATLASCAR2 and its surrounding obstacles
- Develop a visual tool that displays the ATLASCAR2 short term path

1.4 Document Structure

This document is divided into seven chapters. In the current chapter 1 a brief explanation of the goals of this dissertation is presented, as well as a presentation of the project where it is applied. Chapter 2 provides an overview of the methods developed and applied by different authors. In chapter 3 is described the hardware currently available on the ATLASCAR2 and the previously developed software used throughout this dissertation. Chapter 4 presents the method used to determine if the ATLASCAR2 is in a collision course. In chapter 5 a study is presented on the influence of the ATLASCAR2 ego motion in the detected obstacles velocity; A presentation of external target and car properties which influence the collision definition is also given. Chapter 6 presents the results obtained when applying the developed architecture. Finally, chapter 7 concludes this work and presents future possible improvements.

Intentionally blank page.

Chapter 2

State of the Art

2.1 Target Detection and Tracking on ATLASCAR2

As mentioned before, the work developed in this dissertation relies on data from a target detection and tracking algorithm. Being one of the most researched topics of the robotics research community, several algorithms exist to detect and track obstacles in a dynamic environment. The two main approaches use cameras and computer vision or LIDAR sensors. Both approaches have their pros and cons and ideally merging the two would result in a more robust tracking.

This work uses data provided by the Multi Target Tracking (MTT) algorithm [3]. The robustness of this algorithm proved itself worthy not only for this dissertation but also for many other works developed at LAR, as described next.

In [5] both approaches were merged to detect pedestrians in exterior dynamic environments (Figure 2.1). By using the fast detection MTT algorithm, the author was able to reduce the detection time compared to only using a computer vision detection algorithm. In this work, the MTT was used to restrict the area where the computer vision algorithm is applied.

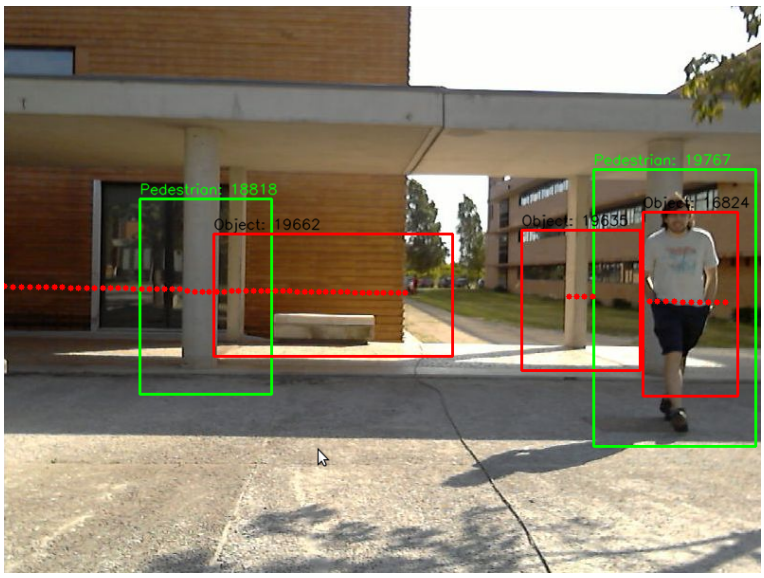


Figure 2.1: Sensor Fusion LIDAR-Camera with LIDAR detected objects in red dots [5].

In [6] the author used LIDAR sensor data as well as a camera to detect, track and label targets on a road in a semi-automatic fashion. A computer vision algorithm was first used to detect targets based on their appearance and then the MTT algorithm was implemented to track the targets. The core of this work is in the computer vision algorithm to label the targets correctly, however, due to the much faster tracking ability of the MTT algorithm, it allowed the author to track the targets' position after the slower labelling process was finished.

2.2 Velocity Obstacles

In the matter of collision detection techniques, a considerable amount of research was made throughout the past decades. Inserted in the grand scheme of motion planning for robots, the collision assessment is essential to determine a collision area to avoid. In this dissertation, the mentioned area is determined by studying the velocity of the surrounding obstacles and this approach was first mentioned by Fiorini et al. [7]. In this work, the Velocity Obstacles (VO) concept was presented. The author studies a collision in a simplistic scenario between a circular obstacle moving at a constant speed in a known linear trajectory, and a circular object avoiding the mentioned collision. By studying the relative velocity vector of the avoiding object, relative to the obstacle, if this vector lies within the *"Relative Collision Cone"*, CC(A,B) in Figure 2.2, a collision between the two will occur. The main goal of this work was to, after computing all the collision-free velocity vectors, select the optimal one. Later on, the author used the same collision detection method (VO), but improving the collision-free velocity vector selection, according to robot motion restrictions [8, 9].

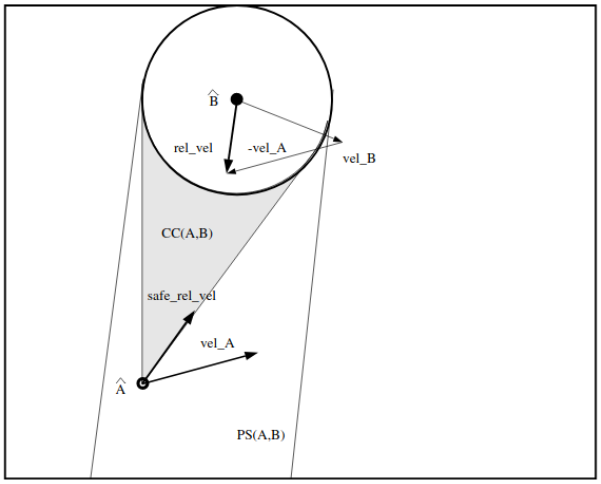
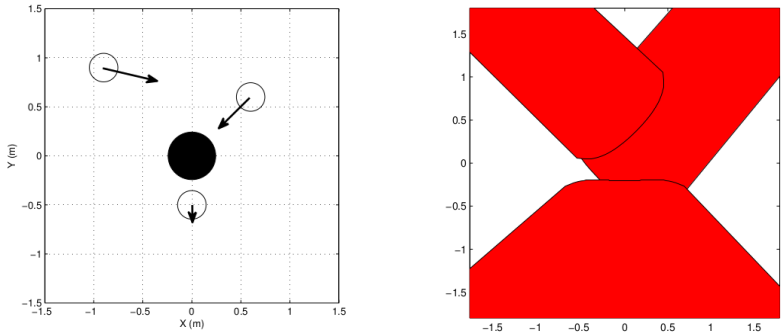


Figure 2.2: *"Relative Collision Cone"* CC(A,B) [7].

On more recent works, similar methods are applied. In [10] the *Forbidden Velocity Map* was developed. The author states that a collision occurs if the object's velocity lies within a set of velocities (Figure 2.3) . These velocities are defined considering that the object will apply it's maximum break power in order to avoid a collision with a moving obstacle and takes into account the obstacles' velocity, distance and shape. Wilkie et al. [11] adapts the VO concept taking into account the fact that a car-like robot evading



(a) Representation of three objects with velocity vectors

(b) Forbidden velocity map

Figure 2.3: Forbidden velocity map example [10].

maneuver is not rectilinear due to kinematic constraints, *i.e.* when a car turns, its path is curved and this curvature might be in the mentioned Velocity Obstacle Collision Cone. In Berg et al. [12] the Acceleration-Velocity Obstacle principle is presented. Based on the VO principle, this method chooses the velocity vector which obeys the robot's acceleration constraints. This work allows to overcome problems in cases where a velocity cannot be reached instantaneously, by applying a valid acceleration proportional to the difference between the new and the current velocity. This paper presents the possibility of using the algorithm by two robots in the same workspace, as they try to change places while avoiding a collision. Gyenes et al's [13] collision avoidance approach uses the same VO principle as in Fiorini et al. [7], however this work differs in the velocity vector selection by choosing one focused on the safety of the robot. The author claims that by choosing a collision-free velocity vector that provides a faster solution to reach the destination, the robot's safety is compromised. This is due to the fact that the faster path is usually closer to the obstacle.

2.3 Ego-motion

The knowledge of the car's ego-motion allows to retrieve the detected targets real position and velocity. This is due to the fact that since the LIDAR sensors are moving along with the ATLASCAR2, the data collected will be affected by this movement, imposing a false velocity on the targets [14, 15]. To counteract this false velocity, the LIDAR data must be converted to a global reference frame.

Commonly used sensors such as Global Position System (GPS), Inertial Measuring Units (IMU) and also wheel encoders can be used to calculate the ego-motion. However, this type of equipment, besides having poor accuracy, can behave inaccurately in some environments. GPS relies on line of sight to satellites, making it useless in confined environments such as road tunnels. Wheel encoders rely on wheel turn which, in rough terrain, can fail due to wheel slip.

Due to the constraints explained above, several different approaches were developed using cameras [16] or LIDAR sensors [15]. The principle in these approaches is to determine the spatial distance between two frames to then define the vehicle's own motion.

Using camera data, also known as Visual Odometry, as in Abiel Aguilar-González et al. [16], the process starts by selecting one feature of an image, then the pixels of that specific feature are tracked using a dynamic search template which informs the feature matching algorithm. Afterwards, by feeding the lookup table with the search parameters of two consecutive image frames, a preliminary ego-motion results is presented. This preliminary result is then refined by a post-processing step (Figure 2.4). Even though using Visual Odometry may result in great accuracy, the computational demand and processing speed is also larger when compared to other approaches such as using LIDAR sensors.

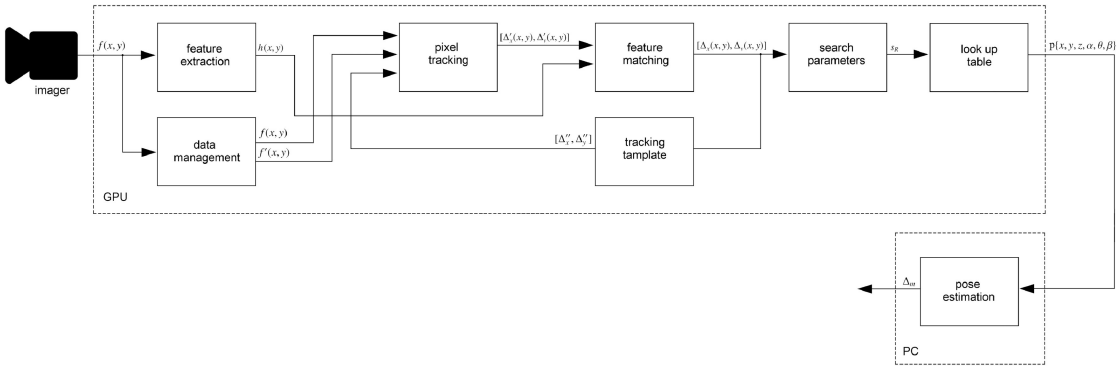


Figure 2.4: A Visual Odometry set up [16].

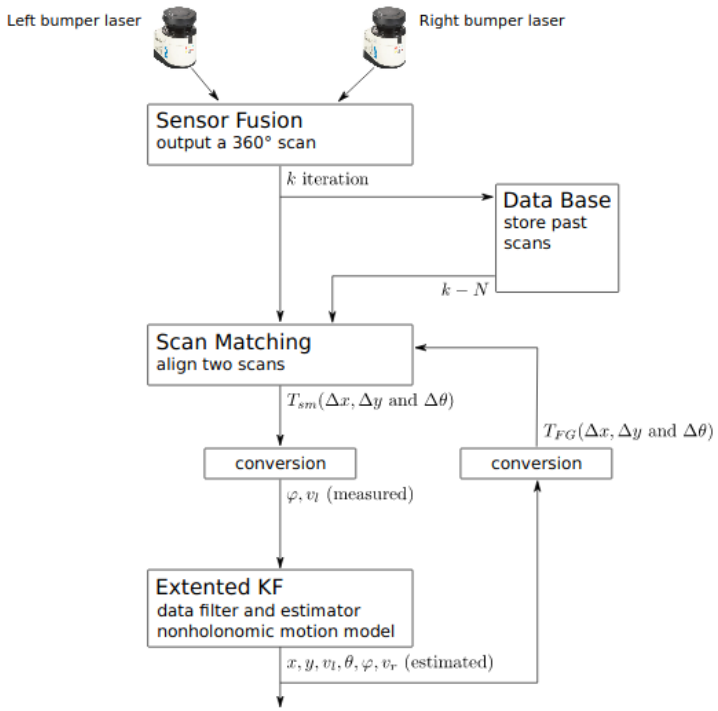


Figure 2.5: A LIDAR based ego-motion estimation setup [15].

In the case of ego-motion estimation using LIDAR, the principle is similar and cal-

culates the displacement between consecutive planar laser scans, aided by the car's odometry [15] (Figure 2.5). In this work, two LIDAR sensors were used to perceive the surrounding environment. In order to only use one set of data, a sensor fusion technique was formulated to merge the two LIDAR data and get an almost 360° sensing angle. The author developed a scan matching algorithm using an Extended Kalman filter with a non-holonomic motion model. Since the Kalman filter relies on measured data, a transformation between two scans is provided as a vector. The vector represents a translation in the X and Y axis, and a rotation along the Z axis of the car. By using this vector, the instantaneous turning center point of the car is determined which allows to calculate the car's ego motion, namely the steering wheel angle and velocity. If these values do not comply with the vehicle constraints, the measurement is discarded and the filter is iterated without a measurement.

2.4 Analysis

Fiorini et al.'s [9] approach can be considered as a baseline for all collision detection work using velocity data, such as this dissertation. The simplicity of the used geometric calculations is extremely useful to reduce the computational effort and therefore ease its application in real time collision calculations. In terms of ego motion definition, wheel encoders will be used despite its known flaws, however, since the velocity data is retrieved from the car's speedometer, no additional hardware is needed. LIDAR could be used however it would increase the computational effort.

Intentionally blank page.

Chapter 3

Experimental Infrastructure

This chapter describes both the software and hardware used during this dissertation. The hardware collects data used by the software to achieve the main purpose of collision detection.

3.1 Hardware

3.1.1 SICK LMS151-10100

The hardware used to detect the multiple objects around the ATLASCAR2 is the 2D LIDAR sensor SICK LMS151-10100 (Figure 3.1). A LIDAR (also known as laser range finder) emits a high-frequency laser beam in different directions which are then reflected by the objects it encounters and the reflection is captured by the device. This type of laser sensors are widely used in the field of robotics to determine objects distances. In order to do so, most LIDAR measurement sensors apply the Time of Flight (TOF) principle which measures the round trip time taken by the laser between the sensor and the object.

Figure 3.1: 2D LIDAR sensor SICK LMS151-10100.

Multiple configurations of the sensor are available from the manufacturer and the configuration used during this work is available in Table 3.1.

Currently, the ATLASCAR2 has two 2D LIDAR sensors located near the front headlights. With this setup, it is possible create a 2D representation of almost 360 degrees around the car, only missing a small portion in the back of the vehicle.

Table 3.1: Used specifications of SICK LMS151-10100.

LIDAR specifications	
Laser class	1 (IEC 60825-1:2014) EN 60825-1:2014
Minimum angle (°)	-135
Maximum angle (°)	135
Angle increment (°)	0.5
Time increment (s)	2.77×10^{-5}
Scan time (s)	0.0199
Minimum range (m)	0.0099
Maximum range (m)	20

3.1.2 ATLASCAR2

As described in chapter 1, the ATLASCAR2 [17, 18] is the main platform of the Atlas Project (Figure 3.2). It is a prototype vehicle for autonomous driving research and, for that purpose, multiple state of the art sensors and cameras are equipped to collect data. Besides the two main LIDAR sensors mentioned before, it's current inventory consists of: four optoelectronic sensors Sick DT20 Hi, one 3D LIDAR sensor, a GPS and IMU combo, one PointGrey ZBR2-PGEHD-20S4C and two PointGrey FL3-GE-28S4-C cameras.

Figure 3.2: ATLASCAR2.

The optoelectronic sensor is an optical measuring sensor that determines the distance between the fixed car frame and the ground. This sensor model can measure from 50 to 1000 millimeters with a measuring resolution of better than 1 millimeter. The sensors' main task is to determine the car's inclination by measuring the difference between the four sensors and applying trigonometric calculations to determine the inclination angles (tilt and roll).

The 3D LIDAR sensor works similarly to the LIDAR sensor used in this dissertation,

with the main difference being the ability to measure in three dimensions via its four measuring planes. This sensor can measured object distances for up to 250 meters.

The PointGrey cameras are an alternative method to perform road perception via computer vision. They are more commonly used to detect road lanes.

The GPS and IMU combo consists of a Novatel GPS-702-GG Dual-Frequency GPS Antenna and a Novatel SPAN-IGM-A1 Inertial Measurement Unit (IMU) and their task is to accurately determine the ATLASCAR2's position in the world reference frame.

Aside from the data collection hardware, the car is also equipped with a computer with Ubuntu Bionic Beaver Desktop operating system installed, two monitors for visualization purposes and a Uninterruptible Power Supply (UPS) that provides about 30 minutes of power to the on-board computer while the car is on the move.

3.2 Software

3.2.1 ROS - Robotic Operating System

ROS (Robotic Operating System) is an open-source robot software development framework created at the Stanford University in 2007, which aims to help the robotics research community on collaborative projects. With its wide variety of tools and libraries, it simplifies the robot software writing throughout many robotic platforms. The version of ROS used in this dissertation was ROS Melodic.

A ROS project is constituted by a package, *i.e* a catkin workspace directory, where all the nodes are located and the dependencies between each are defined using a `package.xml` file. A rosnode is a process that performs computation, *i.e* it's the execution of the code that the ROS programmer writes. This code can be written in many programming languages by using client libraries, being the most common Python and C++. The rosnodes communicate between each other via topic streaming, meaning that one node publishes a message in a topic, and another subscribes to it to receive the topic's message data. These messages can be a "ROS standard" or a custom one made by the user. Another way that the nodes can communicate is using a ROS service. In this case, one node requests for an information and expects a response to be given.

All these nodes and topics are controlled and managed by the ROS master which keeps track of all this information.

Rviz

`Rviz` is a 3D visualization tool used to represent a certain type of message data by subscribing to a topic. The message type used in this dissertation was laser scan, to represent the laser points of objects detected by the LIDAR sensor, and visualization message of type Marker, to represent several geometric objects such as lines, cubes and cylinders. A robot model can also be displayed as a visual scale of the surrounding environment. Other types of messages can also be subscribed and displayed by `rviz` such as camera images, `tf` data and axes, among others. An example of data representation is presented in Figure 3.3.

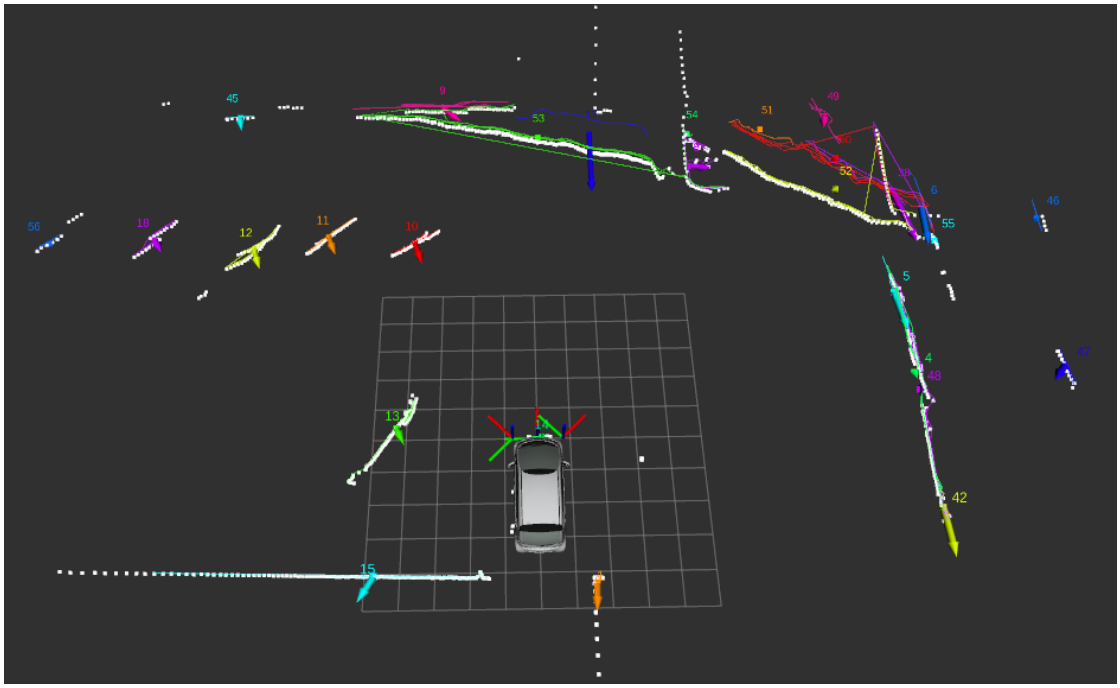


Figure 3.3: Rviz environment displaying the ATLASCAR2 robot model, the LIDAR laserscan, and several markers of the targets.

Rosbag

Rosbag is an essential tool to playback all the recorded data of every hardware of the ATLASCAR2. Since the access to the car is not always possible to test the code developed in real time, rosbag allows to record all the messages of a topic in a bagfile. In this case, laser scan messages where recorded of the LIDAR sensors to then use offline for tests.

3.2.2 ROS Packages

Simple Planar PC Generator

This package, composed by a single `simple_planar_pc_generator` node, was developed by Almeida [3]. The code, developed in C++, subscribes to the laser scan data, provided by the LIDAR sensors, and transforms it into a planar `PointCloud2` data message. This node also has the advantage of merging two LIDAR data to a single topic.

Multi Target Tracking with Global Nearest Neighbour

This package was also developed by Almeida [3] and it provides the necessary target information that will be used throughout this work. The node subscribes to the `simple_planar_pc_generator` topic and assigns each group of points, *i.e.* objects, to a target to track its position. In short, the code is divided in 4 major steps:

1. Pre-processing
2. Clustering

3. Position prediction

4. Data association

In the first step, the `PointCloud2` message is received and its points are converted into Cartesian coordinates. A noise reduction filter is also applied. The second step is responsible to add every point detected to a cluster. This cluster of points will be labeled as measurements which will then be associated to a target to be tracked. To better track the objects while they are occluded, this algorithm uses object position prediction with an adaptive Kalman filter to adjust the objects' search area accordingly. The last step associates the mentioned measurements to a target, where the Global Nearest Neighbour algorithm is applied. The cluster is associated based on the nearest distance to a target, while evaluating simultaneously every target. If there is no target within the stipulated search area, a new one is created. This process repeats at every scan received.

In the code's original version, the node publishes for every target the following information:

- Id number
- Centroid position and orientation
- Linear velocity
- Position of the first and last point, based on the angle
- Size

Colormap

Similar to matlab's Colormap, this auxiliary visual package enables adding color to rviz markers by simply associating a target id to random color. By using this package it makes it easier to add color to the different targets without specifying the RGB values to each manually.

canReceiveAndUpdateStatus

This package was developed by Figueiredo [4] and publishes the telemetry of the ATLASCAR2 namely its velocity and steering wheel angle.

Intentionally blank page.

Chapter 4

Collision Zone Detection

This chapter presents the collision calculations. In the first section a theoretical resolution of the problem is given. In the second section is presented a code overview. The third section explained some of the modifications made to the existing MTT package. The fourth section explains the collision detection procedure and, finally, in the fifth section the procedure to launch the package.

4.1 Approach

The main goal and approach of this dissertation was defined since the beginning: determine the collision zones with the Velocity Obstacles principle [9]. The VO concept allows to determine if a collision will occur based on the geometric properties of the relative velocity vector between an object, in this case the ATLASCAR2, and an obstacle.

With the knowledge of the obstacle's position and velocity relative to a reference frame, and by approximating the obstacle to a circular shape, the *Velocity Cone* can be defined with the known variables. Consider two circular objects **A** and **B**, being **A** the evading object and **B** the obstacle, with respective velocities $\mathbf{v}_A$ and $\mathbf{v}_B$; the first stage of the VO concept is to add **B** to the *Configuration Space* of **A**. The *Configuration Space* [19] defines a forbidden configuration for an object due to the presence of an obstacle. This is determined by reducing **A** to a point and adding the radius of **A** to circle **B**, resulting in the point $\hat{\mathbf{A}}$ representing the object and the circle $\hat{\mathbf{B}}$ representing the obstacle.

To allow the use of geometric calculations to determine the event of a collision, the *Velocity Space* is determined. This is done by adding the velocity vectors to the *Configuration Space* explained above. By doing so, the velocity vectors of the object and obstacle are represented in its corresponding centroid.

Afterwards, the *Collision Cone* $CC_{A,B}$ is represented by two lines tangent to $\hat{\mathbf{B}}$ (γ_r and γ_f) with apex in $\hat{\mathbf{A}}$ (Figure 4.1). A collision between the object-obstacle pair will occur if the relative velocity vector $\mathbf{v}_{A,B}$, lies within the tangent lines γ_r and γ_f.

This method allows to determine the potential ATLASCAR2 collisions with static obstacles or with dynamic obstacles with linear trajectories.

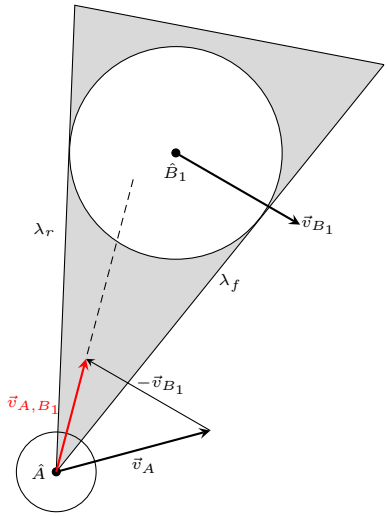


Figure 4.1: *"Collision Cone"* $CC_{A,B}$ represent by the gray area. Adapted from [9].

4.2 Code Overview

To collect data from the ATLASCAR2 surroundings, 2D LIDAR sensors were used, allowing to detect obstacles in an almost 360 degree angle around the car. To perform the detection and tracking of the obstacles the Multi Target Tracking (MTT) algorithm [3] was used. This ROS package uses the LIDAR data to assign the measured obstacle to a target using the Global Nearest Neighbour (GNN) approach. The target is then tracked throughout its lifetime. The work developed by Almeida performs particularly well with occluded obstacles by using an adaptive Kalman filter to estimate the obstacles' position. The GNN approach associates the measured data to the closest target predicted position. This data association has some conditions that need to be met in order to assign the measured data to a target. These conditions, as well as some others, are defined in a *yaml* file in order to make it easier to adjust the values without changing the code (Listing 4.1).

Listing 4.1: MTT's yaml file.

```
%YAML:1.0

   clustering_distance: 0.5
   exclusion_zone_A: 0.0
   exclusion_zone_B: 0.0

   max_missing_iterations: 20
   max_ellipse_axis: 0.6
   min_ellipse_axis: 0.2

   size_factor: 0.1
   not_found_factor: 0.1
```

The `MTT` package contains two main nodes, the `Simple_Planar_PC_Generator` and the `GNN` node. The latter publishes several target information such as position and velocity, needed by the VO. However, the package available was outdated and a migration to the ROS version Melodic was needed before applying it in this work.

The ROS package itself has additional developments by the author which were not needed; so, for the initial step, a cleanup of unessential code was done. Due to the way ROS framework is built, all the code dependencies were corrected in order to compile only the code needed for this dissertation.

Afterwards, with the information of the multiple targets detected by the `MTT` published as a ROS message, the VO principle is applied. This method is divided in three main tasks: target approximation to a circle, collision detection and representation.

4.3 GNN node Improvements

In order to achieve the end goal of this work, some minor modifications to the `MTT` package, specifically the `GNN` node, were needed. However, firstly it is important to understand what type of information it gathers. The `GNN` node provides two major types of information, a visual representation and the targets data. The visual data is represented on `Rviz` (Figure 4.2) and consists of the target id number, a square in the targets centroids, the scaled velocity vector and an ellipse representing the search area of the target. This information is coloured using the `Colormap` auxiliary package, which allows to add different colors based on the target id number. The node also publishes another topic to represent the measured obstacles centroids with a square.

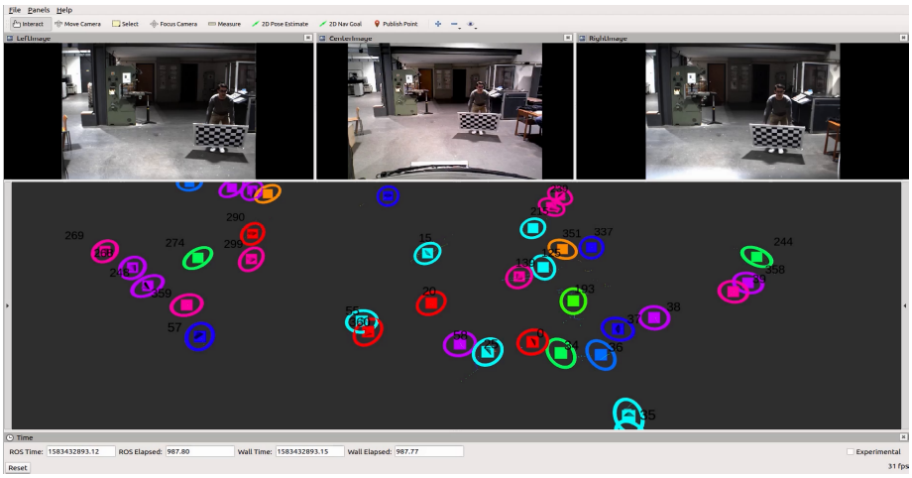


Figure 4.2: Displayed MTT visualization data on Rviz.

To avoid the excessive amount of data transfer by publishing every target data individually, the GNN node uses two different ROS custom messages. ROS custom messages allow the programmer to create a unique type of message by combining preexisting types of standard messages to a single `.msg` file. In this case two `.msg` files were created, the `TargetList.msg` and the `Target.msg`. The `TargetList` message is essentially an array of `Target` messages which display all the targets' information in a single message to be published. The original `Target.msg` consists of the following data:

- Header - A ROS standard message which states the time stamp of the message and the coordinate frame;
- ID - a `uint32` variable of the number of the tracking target;

- Pose - A geometry message of type `Pose` which consists of two geometry messages of type `Point`, informing the position, and of type `Quarternion`, informing the orientation of the centroid of the target;
- Initial Pose - A geometry message of type `Point` regarding the position of the first point of a target;
- Final Pose - Same as the Initial Pose stating the position but of the last point of a target;
- Velocity - A geometry message of type `Twist` regarding both the linear and angular velocity of the target. It is important to note that only the linear velocity is stated by the code.

Only the `TargetList` message is available to be subscribed to, on the `/targets` topic.

4.3.1 Target Information Update

To achieve a better perception of the target, especially in cases were multiple targets are close to each other, a line representation was added. This line connecting every point of a target was easily done since the code already saved all the points of a new target in a grid. This is the first step in the `GNN` code when data is received from the LIDAR. The `PointCloud2` points received are initially in polar coordinates so a transformation into Cartesian coordinates is made. Since the initial points are sorted by angle (from smallest to biggest) the Cartesian coordinates points are added to a grid in the same order. Additional data, such as distance and angle of each point, are also available in this grid.

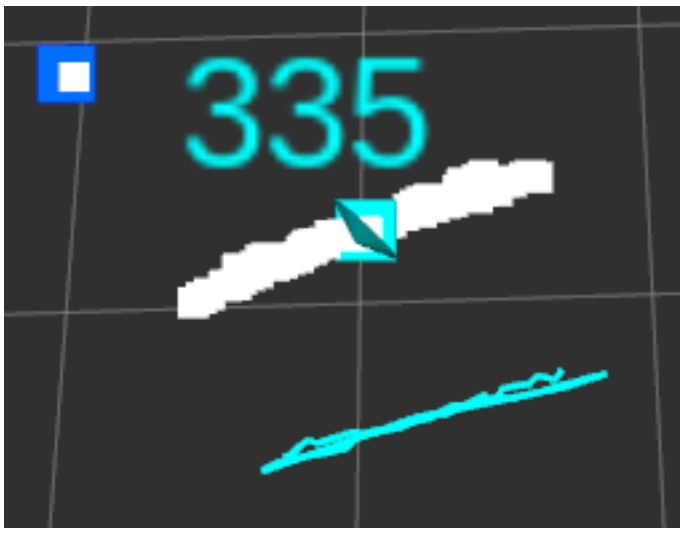


Figure 4.3: Representation errors due to old point data. The target first detection is represented by the cyan line. The target then moved, however the line position was not updated.

However, as stated, this point data was only added when a new target is detected, meaning that the tracking process did not updated the points. This led to multiple

errors in displaying the targets (Figure 4.3) as well as the Initial and Final Pose data which uses this information.

So, in order to constantly update these points, a single line was added in the data association function of the code (Listing 4.2). This line of code simply replaces the targets points with the measured points, every time a measurement is associated with an existing target.

Listing 4.2: Line of code added to update the target points

```
1 target->points = measurement->points
```

With this point information, a `Line Strip Marker` was added, representing a straight line between each point. Some previously displayed information irrelevant for this work was discarded in order to declutter the workspace, resulting in the final representation shown in Figure 4.4.

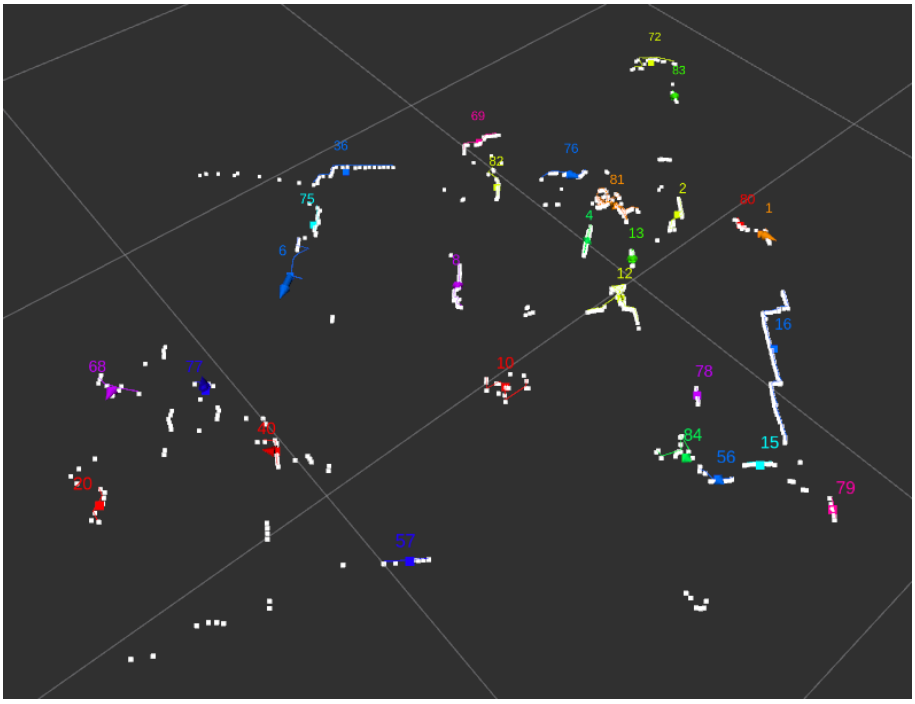


Figure 4.4: Final representation, displaying the target ID, the velocity vector, and a line connecting all points of every target.

4.3.2 Target Reduction

In order to reduce the amount of targets detected by the `GNN` node, some restrictions were added. These restrictions determine the minimum size and minimum number of points that a target must have in order to be accounted for. This strategy allows to disregard small detected obstacles which possibly relate to unfiltered sensor noise, and do not pose a serious threat to the ATLASCAR2. These restrictions were added to the existing `yaml` file, allowing to tweak the values easily. To limit the number of target calculations these restrictions were added in the data association process, only allowing to match a

measurement with a target if these conditions are met. By doing so, this measurement information is still kept, but the calculations and representations are disregarded. These restrictions allowed to further declutter `Rviz`, especially in cases where the surrounding vegetation is detected, as well as reduce the computing time of both the `GNN` and the `VO` nodes.

4.4 Collision Calculation

4.4.1 ATLASCAR2 Frames

Since the ATLASCAR2 has multiple devices installed throughout the car's body, the data collected from different devices are in different coordinate frames. These frames are all connected to the main frame through transformations matrices. The matrices indicate the rotation and translation between the main frame and every other frame. The transformation of every frame is published by the ATLASCAR2 core package in the form of a `tf` tree (Figure 4.6), using the ROS package `tf`. The ATLASCAR2 core package initializes all the sensors and cameras and publishes its data in several topics, as well as the previously mentioned `tf` tree. These frames can be displayed on `Rviz` and the most relevant ones for this work are displayed in Figure 4.5.

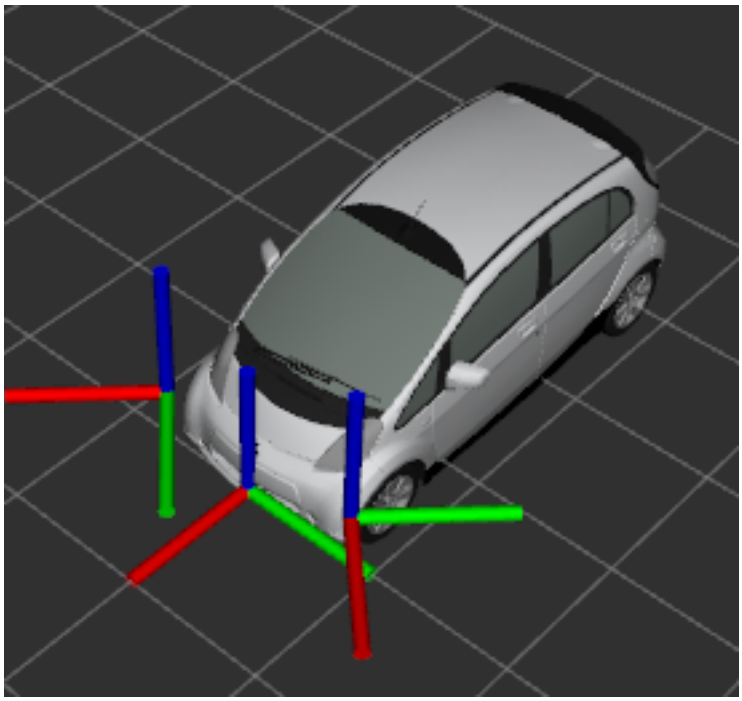


Figure 4.5: The ATLASCAR2 relevant frames; From left to right: *Right_laser*, *base_link*, *left_laser*.

The Figure shows three frames, all with different orientations; the different colours represent the x (red), y (green) and z (blue) axles. The two LIDARs are located in the left and right front bumpers of the car, and the data collected are in the `Left_laser` and `Right_laser` coordinate frame, respectively. In the front of the car, in the center, lies the `base_link` frame. This frame was used as the reference frame for all the collision

calculations, since it is, approximately, the first colliding point of the car. This frame also benefits from having the same orientation as the car instant velocity vector.

The `MTT` package receives the left and right LIDAR data and merges both in one frame (either left or right). This process is done by the `Simple_Planar_PC_Generator` node which publishes the merged laser scans as a filtered `PointCloud2` message. Since the `GNN` node uses this data, all the targets information are related to one of those reference frames. In order to transform the relevant data from the `GNN` node to the `base_link` several `Transform` functions from the `tf` package were used to transform each `Point` from the GNN node published message.

4.4.2 Circle Approximation

To apply the VO principle, all the targets must be approximated to a circle. Since in most cases the detected obstacles are only represented by a line of points, the solution chosen was to set this line as the circle diameter. Using the information given by the `GNN` node regarding the initial and final points of the target, the distance between them was calculated. Considering that the radius of the evading object, *i.e.* the ATLASCAR2, must be added to every target circle, the same method was applied. In this case, the width of the car was used as the approximated circle diameter of the ATLASCAR2. The position of the circle was set to the centroid point, given by the `GNN` node. The `GNN` node defines the centroid based on the point distribution of the target. This definition means that if the target's points are not well balanced, some of them may lie outside of the circle.

Reliability of the circle approximation is limited by the information given by the LIDARs and the `GNN` node. The full size of the obstacles cannot be detected in some situations, reducing the collision calculations reliability and possibly jeopardizing the ATLASCAR2 and its surroundings. However, by using the same approximation for the ATLASCAR2, the car-on-car collision prediction is more accurate than a car-on-truck collision.

4.4.3 Velocity Cone Calculation

The next step in the collision calculation is the definition of the velocity cone. This process defines the relative velocities that would result in a collision. The VO principle states that the points that define the cone are the centroid of the evading object and the two tangent points to the target's circle. The centroid of the ATLASCAR2 was set to the origin of the `base_link`, since it is the potential contact point in case of a collision. The two tangent points were calculated using geometric equations. Using the example in Figure 4.7, being P the target, O the `base_link` coordinate frame, T_1 and T_2 the tangent points and r the radius of the circle.

Both $\overline{PO}$ and r are known or measurable variables. The tangent angles θ_1 and θ_2 are calculated using Equations 4.1, 4.2, 4.3 and 4.4, using $n = 1, 2$, respectively.

$$\overline{PO}^2 = r^2 + \overline{OT_n}^2 \tag{4.1}$$

$$\alpha = \arcsin \frac{r}{\overline{PO}} \tag{4.2}$$

$$\beta = \arctan \frac{P_y}{P_x} \tag{4.3}$$

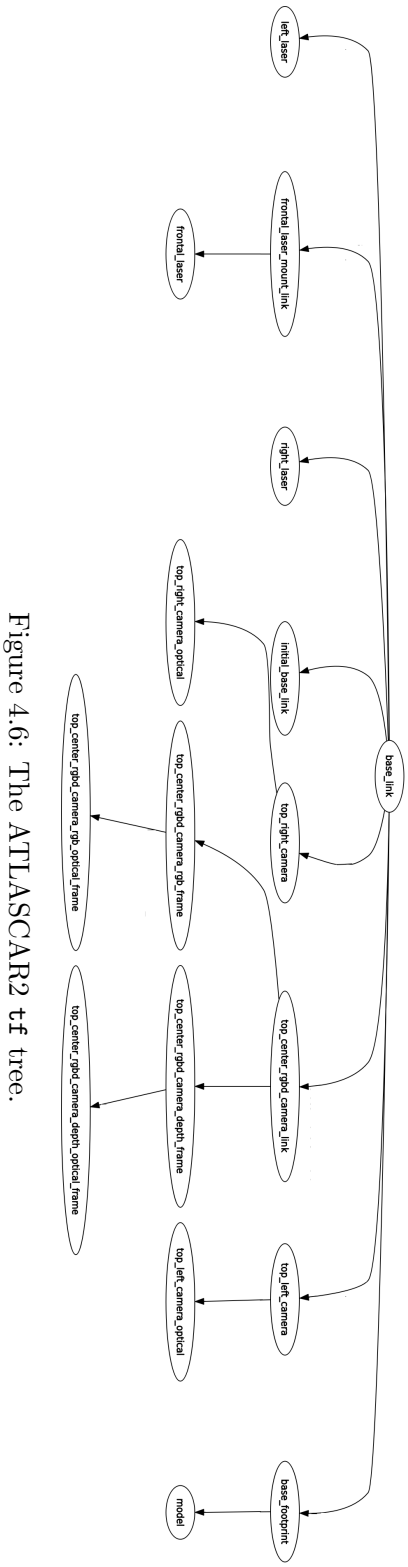


Figure 4.6: The ATLASCAR2 `tf` tree.

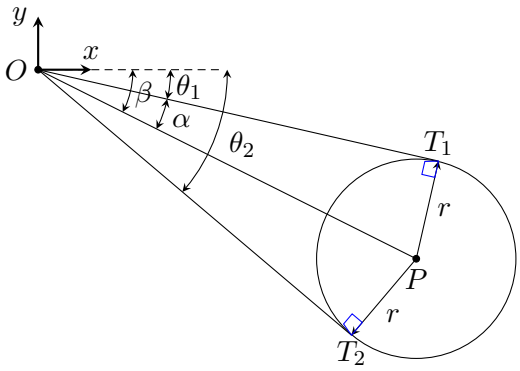


Figure 4.7: Velocity Cone Schematics.

$$\begin{aligned} \theta_1 &= \beta - \alpha \\ \theta_2 &= \beta + \alpha \end{aligned} \tag{4.4}$$

Equation 4.4 calculates two angles (θ_1 and θ_2) representing the angles of the two tangent lines of the circle. To calculate the tangent points for the line representation, Equation 4.5 can be used to determine the x and y coordinate.

$$\begin{aligned} T_{n,x} &= \overline{OT_n} \times \cos\theta_n \\ T_{n,y} &= \overline{OT_n} \times \sin\theta_n \end{aligned} \tag{4.5}$$

Where:

$n = 1,\ 2$

4.4.4 Velocity Definition

To determine if a collision is imminent, the approach defined in this dissertation makes use of the velocity data of the targets. Considering that the ATLASCAR2 is gathering data while on the move, the perceived data of the targets is corrupted with the car's own ego-motion. The only velocity that can be calculated using the raw data from the LIDARs is in fact the relative velocity between the car and the target. The target real velocity cannot be perceived by the moving sensor. These velocity errors without the ego-motion compensation led to additional studies explained further in the upcoming chapter.

4.4.5 Collision Detection

Assuming the velocity given by the `GNN` node is correct, the collision detection makes use of this vector to determine if it lies within the previously defined Velocity Cone. To do so, angle comparisons are made. Firstly, the angle of the velocity vector is calculated based on the x and y coordinates of the linear velocity. Being the previously calculated θ_1 and θ_2 the angles of the tangent lines to the circle, *i.e.* the VO cone, and $\angle V_v$ the angle of the velocity vector. If $\theta_1 < \angle V_v < \theta_2$ a collision will occur.

When a collision is detected, a message is published on the `/collision` topic containing the colliding target ID and the Expected Time of Colision (ETC) if all the

conditions are kept. The ETC is calculated using Equation 4.6 with the minimum distance of the target, *i.e.* the distance to the closest target point, and the modulus of the target velocity.

$$ETC = \frac{distance}{velocity} \tag{4.6}$$

A visual representation of the Velocity Cone is also displayed on `Rviz`. However, to reduce the amount of visual data, this representation is restricted and is only displayed if an expected time of collision parameter is met. This parameter allows to disregard targets further away or with minimal relative velocity, only representing vital information.

4.5 Launching Node Procedure

To ease the initialization of the developed package, launch files were created. The launch files allow to run all the nodes with a simple terminal command. To start the collision detection package, first all the needed ATLASCAR2 sensors are initialized, by using the previously developed `atlascar2_bringup.launch` file. Afterwards, by running the developed `vo.launch` file (Listing 4.3), it initializes the `Simple_Planar_PC_Generator` node, the `GNN` node, the `VO` node and, finally, the `Rviz` node. The `Rviz` node, however, is only initialized if defined by the user, as it uses a Boolean argument from the launch command (`show:=0` or `show:=1`). To launch the visualization on `Rviz`, another launch file (`view_vcones.launch`) was developed with all the needed topics already subscribed to at start. This, in fact, is the file that the main `vo.launch` will initiate if the `show:=1` condition is met. This condition was developed to ease the debug, since if any of the other nodes were to be shutdown to recompile, Rviz needed to be closed as well. By having two separate launch files the code recompilation was possible without closing Rviz.

Listing 4.3: `vo.launch` file which launches all the needed nodes and `Rviz`.

```
<?xml version="1.0"?>
<launch>
   <!-- This starts the Velocity Obstacles with the GNN
   and simple planar pc generator-->

   <remap from="/laserscan1" to="/left_laser/laserscan"/>
   <remap from="/laserscan2" to="/right_laser/laserscan"/>

   <remap from="/tracking_frame" to="/left_laser"/>
   <remap from="/pc_out" to="/laser/points"/>

   <node name="planar_pc" pkg="mtt" type="simple_planar_pc_generator">
      <param name="output_frequency" value="100.0"/>
      <param name="perpendicular_treshold" value="0.15"/>
      <param name="wait_for_laser_1" value="true"/>
      <param name="wait_for_laser_2" value="true"/>
   </node>

   <remap from="/points" to="/laser/points"/>

   <node name="gnn" pkg="mtt" type="gnn" output="screen">
```

```
22        <param name="parameters" value="package://mtt/src/gnn.yaml"/>
23        <param name="perpendicular_treshold" value="0.15"/>
24    </node>
25
26    <node name="vo" pkg="velocityobstacle" type="vo.py" output="screen">
27    </node>
28
29    <!-- Launch rviz with the propper parameters for loading
30             the atlascar rviz configuration file if argument SHOW is 1-->
31    <include file="$(find velocityobstacle)/launch/view_vcones.launch"
32                                              if="$(arg show)"/>
33
34 </launch>
```

Intentionally blank page.

Chapter 5

Ego and Target Motion Interaction

The main goal of this chapter is to study the impact of the ATLASCAR2 ego motion on the predicted velocity given by the `GNN` node. To do so, an auxiliary ROS node is proposed to calculate and publish some important ATLASCAR2 data such as the instantaneous turning center point, the car's angular velocity and a short term path displayed on `Rviz`, among others. This node allows to determine when the car is turning and add additional information about the car's path.

5.1 ATLASCAR2 Short Term Path

As stated in the previous chapter, the data collected by moving LIDAR sensors can be unreliable to determine the velocity of the targets using the `MTT` package. This miss perception is higher when either the ATLASCAR2 or the obstacle are turning, since the angular velocity is not accounted for on the current `MTT` package.

This issue can be overcome by implementing an ego motion estimation algorithm such as the one developed by Almeida [15]. This algorithm detects the ATLASCAR2 motion using LIDAR data which allows to change the moving coordinate system of the objects to a global reference frame. Currently all the detected object points are related to the moving frame of the ATLASCAR2, this means that the car is static and the points move according to the relative velocity between them and the ATLASCAR2 own velocity. By changing the frame of the points to a global reference frame, the detected points would move only according to their true velocity.

In order to determine the current ATLASCAR2 status, a ROS node was developed in Python language. This `Short_Term_Path` node verifies if the car is turning and determines its instantaneous turning center point, its angular velocity and other important data. The node also displays on `Rviz` the short term path of the car, *i.e.*, the trajectory of the ATLASCAR2 in the near future if the velocity and steering attitudes remain constant.

To calculate this, the node subscribes to the ATLASCAR2 velocity and steering wheel angle data, provided by the `NominalData` topic from the package developed by Figueiredo [4].

5.1.1 Instantaneous Turning Center Point

By analysing the car's velocity and steering wheel attitude, it is possible to determine its instantaneous turning center point. Most of the commercial four-wheel vehicles uses the Ackermann steering geometry principle or a variation from it [20], which allows the vehicle to curve without tire slip. This principle states that when a vehicle is performing a curved path, all of its wheel axles are arranged so that the wheels all face to the same center point. Since the rear wheels of the vehicle are not allowed to rotate, the turning center point is located at the intersection of a line extension of the rear axle and a line perpendicular to either of the front wheels. Using this principle the two front wheels rotate at different angles, being the inner wheel angle greater than the outer wheel angle.

To calculate the instantaneous turning center point, first a relationship between the steering wheel angle and the front wheels angle is needed. According to the car's manual, the maximum wheel angle for the outer and inner wheel is 38° and 45°, respectively. The node made by Figueiredo [4] publishes the steering wheel angle from $-635°$ to $635°$ (being the angle positive when turning left). To determine this relationship, the steering ratio was calculated with the known maximum steering wheel angle and both the inner and outer wheel maximum angle (Equation 5.1). This steering ratio is then used to determine both front wheel angles based on the given steering wheel angle by adjusting the previous equation.

$$steering\ ratio = \frac{max\ steering\ wheel\ angle}{max\ wheel\ angle} \tag{5.1}$$

To calculate the turning center point only one wheel angle is needed, in this case the inner wheel was used for the rest of the calculations. The representation of the following terms can be seen in Figure 5.1 and 5.2 , being L the car's wheelbase, d the car's width, and y the car's front overhang.

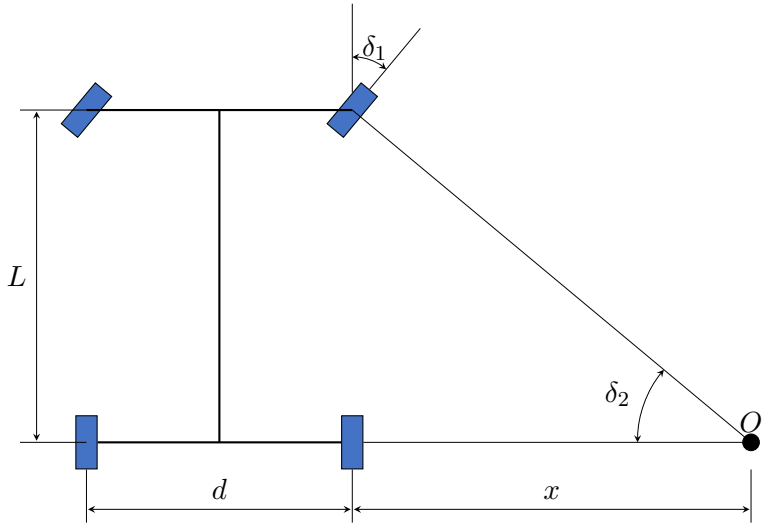


Figure 5.1: Schematics of the *Ackermann steering geometry.*

According to trigonometric rules, both angles (δ_1 and δ_2) are equal, so the distance (x) between the car's rear wheel and the turning center point can be calculated using Equation 5.2. This value is then used to determine the distance between the turning

center point to the front of the car (C in Figure 5.2), using Equation 5.3 to determine angle α and then Equation 5.4.

The Point C in Figure 5.2 is located at the front of the car and is an approximation to the `base_link` frame origin of the ATLASCAR2. All calculations and data acquisition are made relative to this frame.

$$\tan \delta_2 = \frac{L}{x} \tag{5.2}$$

$$\tan \alpha = \frac{L + y}{\frac{d}{2} + x} \tag{5.3}$$

$$\sin \alpha = \frac{L + y}{\overline{OC}} \tag{5.4}$$

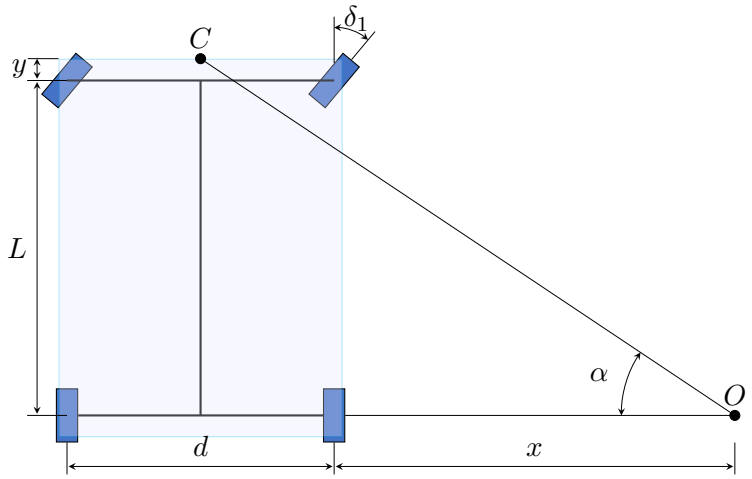


Figure 5.2: Schematics of the *Ackermann steering geometry*.

A line between the ATLASCAR2 rear axle and the instantaneous turning center point is displayed on `Rviz` (Figure 5.3).

5.1.2 Angular Velocity

The ATLASCAR2 angular velocity is calculated based on the distance between the `base_link` and the instantaneous turning center point and the current car velocity by applying Equation 5.5.

$$\omega = \frac{v}{r} \tag{5.5}$$

5.1.3 Instantaneous Velocity

The ATLASCAR2 instantaneous velocity vector was calculated based on the angle between the rear axle and the `base_link` origin (α on Figure 5.2). A representation of this vector is also displayed on `Rviz` by an arrow which allows to see the direction where the car is heading (Figure 5.3).

5.1.4 Path Representation

With the instantaneous turning center point defined, a representation of the ATLASCAR2 short term path is displayed on `Rviz` (Figure 5.3). This path takes into account the wheel angle and the car velocity to display a circular arc with its center on the turning center point and with its radius as the distance from the turning center point to the `base_link` origin. The starting angle is α (in Equation 5.3) and the end angle varies considering the car's angular velocity multiplied by a scale factor. If the $\alpha = 0$ then the path as the same length as the velocity vector.

The `Rviz` marker displays the path that the ATLASCAR2 will take at every scan if all conditions remain constant.

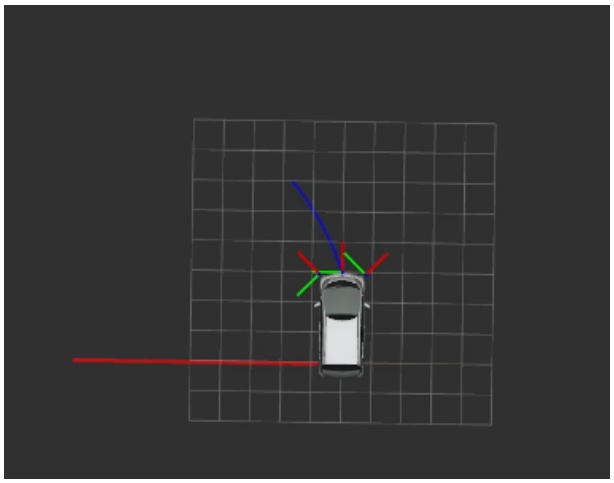


Figure 5.3: Representation of the ATLASCAR2 Short Term Path on `Rviz`.

5.1.5 Node Topics

Gathering all the information explained before, the node publishes a message on the `/Telemetry` topic containing the following data:

- The ROS header
- The current direction of the ATLASCAR2 - Left, right or forward
- The outer wheel angle
- The inner wheel angle
- The angle between the rear axle and the *base_link* origin
- A geometry message of type Twist containing the linear and angular velocity
- A geometry message of type Point informing the instantaneous turning center point
- The distance between the *base_link* origin and the instantaneous turning center point

This information allows other subscriber nodes to know the current status of the ATLASCAR2 at approximately the same rate of the `/NominalData` topic, which is currently at 1 kHz.

5.2 Problem Description

The detected velocity errors of the `GNN` targets are due to the false movement of the obstacles perceived by the moving LIDARs. Since the collected data is relative to the car, so is the calculated target velocity. The car movement will impose an apparent velocity, both linear and angular, on the target due to its own ego motion. This apparent velocity will be added to the targets own velocity, resulting in a wrong relative velocity between the ATLASCAR2 and the target. When the ATLASCAR2 is performing a linear trajectory all of the static target points are moving in the opposite direction of the car, with the same velocity. However, when the ATLASCAR2 is turning, the static target points rotate, imposing an even greater apparent velocity. The most extreme case is when the car is performing a 90° turn, which, depending on the distance of the target, its velocity can wrongly reach extremely high velocities. In addition, the target points 90° rotation also cause problems of target association to the `GNN` node.

Being the angular velocity the key issue, this problem can be divided into four separate instances of velocity errors of the `GNN` node:

- The ATLASCAR2 is turning and the obstacle is static
- The ATLASCAR2 is turning and the obstacle is moving straight
- The ATLASCAR2 is static and the obstacle is turning
- Both the ATLASCAR2 and the obstacle are turning

This study is only focused on the first two instances since, with the current setup, the detection of an obstacle angular velocity is not viable. This, however, proves to be an important matter for further investigation and in the near future this perception will certainly be facilitated with the implementation of Vehicle to Vehicle communication, where cars share their status to other agents on the road. The assessment of when the obstacle is turning is also useful since the current version of the `VO` is only viable for rectilinear trajectories.

5.2.1 Static and Dynamic Targets

Since the LIDARs are moving along with the ATLASCAR2, the collected data, the velocity in particular, is considered to be relative to the car and is always being affected with its movement. This means that a known static object, in the LIDAR's perspective, will move according to the ATLASCAR2 velocity. The target velocity is then calculated based on this imposed movement to the target's points, by the `GNN` node. For a linear trajectory the car velocity is replicated by the target, since it is the only one moving. However, when the car is turning the same does not apply.

When the ATLASCAR2 is performing a turn it rotates around the previously calculated instantaneous turning center point with an angular velocity. In the LIDAR's perspective it is the target which rotates around this point. In theory the target angular

velocity is the same as the car's, calculated by the `Short_Term_Path` node, however the linear velocity is not, as the target may be at a different distance from the turning center compared to the ATLASCAR2 (Equation 5.5). The difference in the distance to the instantaneous turning center point determines how severe the influence of this imposed velocity on the target is. For a static target placed exactly in the turning center point its velocity is considered to be null, since according to Equation 5.5 for $r = 0$,$v = 0$ as well.

On the other hand, when a dynamic target is rotating around the same turning center point at the same distance to it as the ATLASCAR2, its velocity is also perceived as null. This is due to the fact that the target is moving at the same velocity as the ATLASCAR2 (Equation 5.5 being $r_{ATLASCAR2} = r_{target}$) with the target's points constantly at the same distance to the LIDARs, therefore not moving (apparently). In contrast to the previously mentioned cased, this perceived velocity is in fact the relative velocity between the car and the target.

For better understanding, Figure 5.4 presents an example of the ATLASCAR2 performing a right turn. The ATLASCAR2 has velocity (v_c) and a target T is initially detected at a distance j, represented by T_i. The path of the car is represented by an arc P_{car} with its center on the instantaneous turning center point O and radius R. Since only the target T will be moving relative to the ATLASCAR2 frame, its path is represented by the arc P_{target} with the same center O but instead with radius D. The car's path is clockwise, so the LIDAR perceived target path is the opposite (counter clockwise). Both O and T_i coordinates are known, or measurable, and used to determine the distance D using the *Euclidean Distance* equation. The angular velocity of the ATLASCAR2 is calculated using Equation 5.5. Since both the target and the car share the same angular velocity, using again Equation 5.5, now with the distance D and the ATLASCAR2's angular velocity, the relative velocity between the target and the ATLASCAR2 (v_{tc}) is obtained. As the target is further away from the turning center point its linear velocity is greater than the car's. This means that the theoretical velocity of the target (v_t) is the difference between v_{tc} and the car velocity (v_c) as in Equation 5.6.

$$v_t = v_{tc} - v_c \tag{5.6}$$

The velocity calculated by the `GNN` is expected to be equal to the target relative velocity ($v_{GNN} = v_{tc}$). If not, it can be assumed that the target has velocity of its own : $v_t = v_{GNN} - v_c$.

Apart from the apparent velocity errors some additional issues may rise due to the `GNN` architecture. The code updates the target's points at every scan; this update creates some issues in the clustering process, particularly if a point is close to the clustering distance of a target. This means that the number of points of a target can change at every scan due to measurement noise. When a new point is added to a target its centroid can shift, especially if the new target point is further away from other target points. This is due to the fact that the centroid of the target is calculated based on point density, meaning that the centroid is located in an area where more points are. This shift in the target centroid is perceived as movement of the target by the `GNN`. Even when the ATLASCAR2 is static this issue itself causes errors in the `GNN` velocity. The `GNN` node also uses the Kalman filter with a constant velocity motion model, this means that when the ATLASCAR2 is accelerating or decelerating its tracking and velocity detection performance decreases and therefore the velocity prediction is more prone

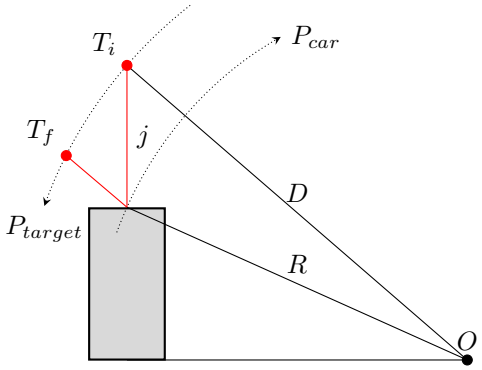


Figure 5.4: Example of a static target T while the ATLASCAR2 is performing a right turn.

to errors, the same happens from the centroid shift with its rapid change of position therefore an extreme acceleration. An implementation of a motion model which better represents road environment users can potentially improve performance in the first case, as in the latter case this implementation can improve the performance however it is not the desirable solution.

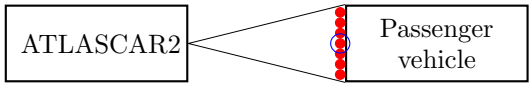
5.3 Refinement of ETC using target and contextual properties

The VO principle predicts collision between two objects based on the relative velocity between them. The Expected Time of Collision (ETC) is therefore based on this velocity and the distance between the two colliding objects. The definition of a collision is based on the approximation of the objects to a circle, as explained in the previous chapter. This method, however, is very sensitive to the correct detection of the colliding target. Additional external properties, such as the weather or road maintenance also plays a big role on ETC.

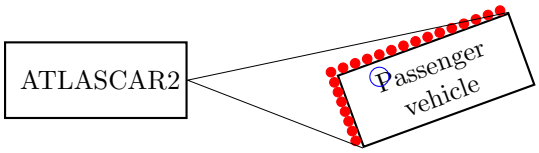
5.3.1 Target properties

In a road environment, the usage of LIDAR sensors alone to detect other vehicles travelling on the same lane can be insufficient to describe the full dimensions of the possible colliding vehicle. A vehicle travelling in front of the ATLASCAR2 is described by the LIDAR as a set of points with which can be calculated the vehicle's width (Figure 5.5). This width information is similar to both a passenger vehicle and a heavy truck which leads to the same circle approximation. These two types of vehicles, however, have different volumes and may lead to a incorrect collision detection.

The same case occurs in the distinction between a bicycle and a motorcycle, so an additional characterization algorithm is necessary in these cases. To do so, a more appropriate hardware is needed such as a 3D LIDAR sensor, or even using cameras and computer vision algorithms which allows to perceive the front vehicle's height. With the information of the height of the vehicle, a proper distinction between a passenger vehicle and a truck is possible, adding more information of the obstacle. This information



(a) Detection vehicle travelling in the same lane in front of the ATLASCAR2.



(b) Detection of same type of vehicle, but instead with added point data.

Figure 5.5: Detection of passenger vehicles. The front vehicle's LIDAR detected points are represented in red and its centroid in blue. As displayed, by adding additional points to the target its centroid shifted greatly, adding false movement to the target. The added point data also increases the target calculated width resulting in different collision calculations for the same type of vehicle.

can be used to develop a bounding box of the detected obstacle, which is continuously being updated as target points increase. This bounding box can then be compared to a database of road vehicles, allowing to achieve a better characterization of the vehicle true or approximated dimensions.

Vehicle characterization also allows to determine the risk to the passengers on board of the ATLASCAR2 in case of a possible collision. A larger vehicle has less maneuverability which decreases the probability of a truck avoiding the collision, therefore increasing the accuracy of the VO collision detection, assuming the circle approximation is correct. A smaller vehicle, on the other hand, is more unpredictable due to its high maneuverability and high acceleration. This is also the case for a pedestrian, where the rapid change of direction and extreme unpredictability makes the collision detection unstable, putting in risk all road users.

Knowing the obstacles constraints allows to adjust the ETC based on target maneuverability and predictability.

In the unfortunate case where a collision cannot be avoided and multiple road users are at risk, a good distinction between them can be advantageous by determining the "safest" collision for all. A greater ETC can be related to a slower colliding velocity and therefore a more "safe" collision. So, if the ATLASCAR2 is allowed to change its course to collide with other road users, the ETC paired with a colliding hierarchy allows to determine which is the chosen colliding object.

5.3.2 Contextual properties

As stated, the ETC is based on the relative velocity and distance between objects. This, however, can be further refined based on other aspects of the surrounding environment. The weather plays a big role on road safety. Road users change their behaviour based on the conditions of the weather, and it is commonly advised to slow down when driving in the rain, for example. This is due to loss of tire adherence to the road which increases the risk of vehicle collision. The same happens in poorly maintained roads. These

proprieties should be accounted for when calculating the collision risk, and the ETC can change since the brake distance is expected to be higher.

As seen, the collision detection and ETC calculation is very sensitive to external properties. The maximum safety can only be achieved if all of these are accounted for. Human beings constantly change their driving behaviour based on the surrounding environment and are always assessing their risk based on numerous variables. An Advanced Driver-Assistance System (ADAS) must be able to keep track of all these variable in order to provide a safer journey to its user.

Intentionally blank page.

Chapter 6

Tests and Results

This chapter presents the experiments made to demonstrate the reliability of the collision detection algorithm developed, and to access the influence of the apparent velocity on the velocity detected by the `GNN` node. Figure 6.1 shows the satellite view of the parking lot where the experiments took place at Aveiro University. These experiments involved the ATLASCAR2, a passenger vehicle portraying a dynamic target and pedestrians representing a static target. Therefore the following chapter is divided into two sections explaining the experiments for each target profile. Some of the following experiments are available on youtube[1], being the dynamic target portrayed by a bicycle.

Figure 6.1: Satellite view of the parking lot where the experiments where conducted.

6.1 Static Target

Three types of experiments were conducted using static pedestrians: imminent front collision, after turn collision and rotating around two pedestrians. The first two, as the name implies, were aimed to evaluate the collision detection algorithm, while the latter one evaluates the influence of the apparent velocity.

[1] `https://youtu.be/dtRU35QLUhk`

6.1.1 Front Collision

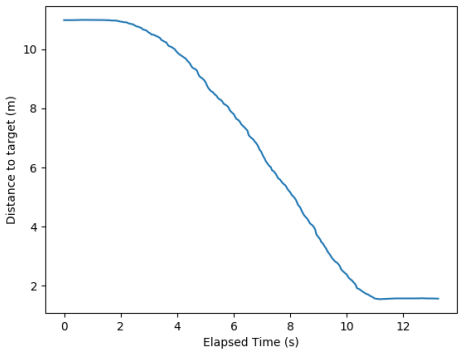
In this experiment a static pedestrian was placed at approximately 12 meters away from the ATLASCAR2 which drove at approximately 6 km/h towards the target. The initial setup and `Rviz` visualization is presented in Figure 6.2. This experiment allows to determine the correct collision detection as soon as the car accelerates, as indicated by the collision cone representation in the figure. During the experiment, the code gathers data and publishes on the `collisions` topic which is then used to evaluate the accuracy of the ETC by analyzing the graphics in Figure 6.3.

Figure 6.2: Front collision setup on the left and on the right the `Rviz` visualization of the VO cone (the blue lines) as well as the enlarged target circle .

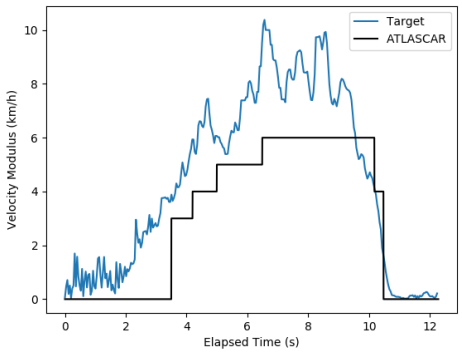
Graphic 6.3(a) displays the distance detected by the `GNN` node throughout the duration of the experiment. The graphic shows that after the initial acceleration the target distance is decreasing at an approximately constant rate throughout the majority of the experiment, though in the final section the car reduced its speed when within a few meter to the pedestrian, in order to avoid an extreme deceleration.

Graphic 6.3(b), displays the velocity modulus of the target in blue as well as the ATLASCAR2 velocity. In this case, since the target is static, the velocities of both the target and the ATLASCAR2 are expected to be the same. Despite the low resolution of the ATLASCAR2 velocity data, the graphic shows that the target velocity follows the accelerations and deceleration of the car, however, when the ATLASCAR2 velocity stabilizes at 6 km/h the same does not happen with the target velocity, increasing the difference between the two. With a more in depth analysis, it is possible to denote more point oscillation in the first few seconds of the data. This was when the target was first detected and the ATLASCAR2 is not moving. The velocity oscillation is due to the centroid shift, which makes the target appear to be moving despite that its detected points are not (see subsection 5.2.1). The average velocity of the target in this time period where the ATLASCAR2 is static is 0.73 km/h. As this occurs when there is no movement of both the ATLASCAR2 and the pedestrian, it can be assumed that the centroid shifts throughout the duration of the experiment, which could explain the overall velocity difference. Throughout the experiment, the target velocity average is 4.4 km/h as for the ATLASCAR2 its velocity average is 2.94 km/h, representing a significant

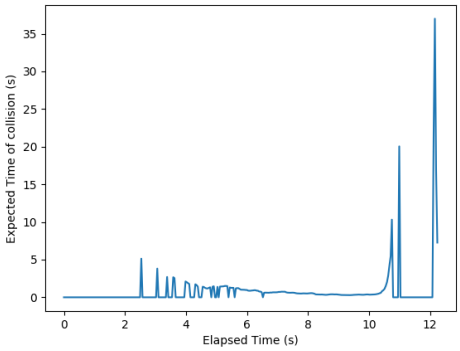
difference between the two.



(a) Target distance over time.



(b) Target velocity as detected by the GNN (in blue) and ATLASCAR2 velocity (in black) during the experiment.



(c) Target ETC over time, being 0 when a collision is not detected.

Figure 6.3: Charts of distance, velocity modulus and ETC data collected in a frontal collision with a static target.

The plot 6.3(c) displays the ETC over time, being 0 when a collision is not detected. Since the ETC is calculated based on the target distance and velocity, the represented data is in line with the distance and velocity charts, being the ETC higher in the end of the experiment as the ATLASCAR2 velocity is lower. By counting the non-zero values and dividing by the number of data collected throughout the experiment, the percentage of time when a collision was detected can be measured, which in this case resulted in approximately 54%. This value however takes into account the initial part of the experiment when the ATLASCAR2 was not moving, thus no risk of collision. By only accounting for the ($t > 2s$) time period the percentage is close to 65%. This value may be influenced by the centroid shift which causes the target to move sideways relative to the car, in the car's y axis (see subsection 4.4.1 for the car's coordinate frame), and therefore the relative velocity vector lies outside the VO cone. As the velocity decreases, this lateral shift error increases since the target's lateral velocity (velocity in the y axis)

is higher compared to the velocity in the car's x axis, thus the miss detection in the final stage of the experiment.

Overall the graphics demonstrate the known vulnerabilities of the *VO* node and its high dependency on the accuracy of the *GNN* velocity data. Despite the car accelerations throughout the experiment, the Kalman filter constant velocity motion model was able to track the target during the car's course and detect an average velocity approximated to the car's velocity.

6.1.2 After Turn Collision

The after turn collision experiment main goal was not only to study the collision detection accuracy but also the influence of the apparent velocity mentioned in the previous chapters. The setup consisted of a static pedestrian placed in the end of a 90° left turn as represented in Figure 6.4. The car's velocity profile is the same as the previous experiment of approximately 6 km/h. The collected data is represented in the graphics of Figure 6.5.

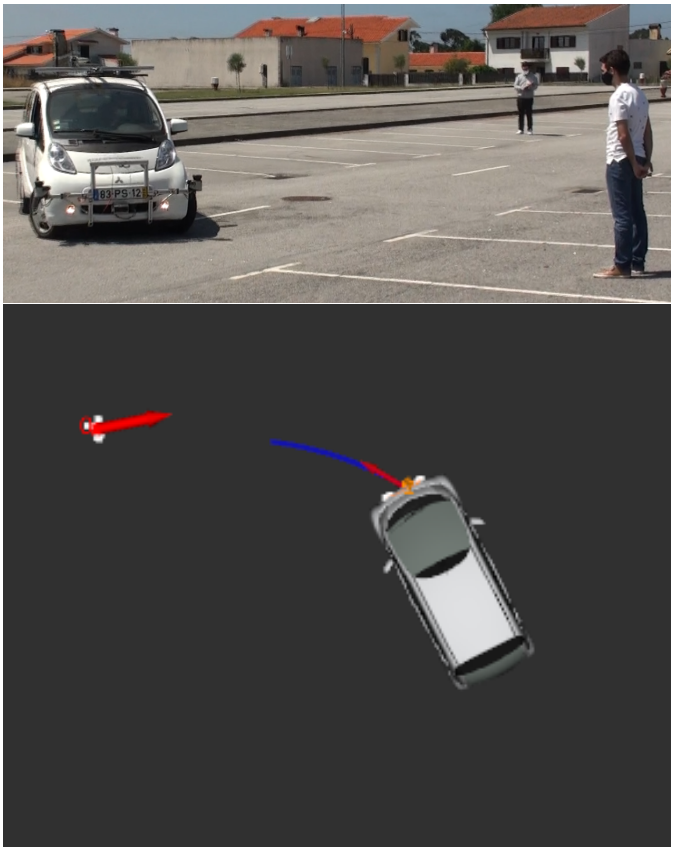
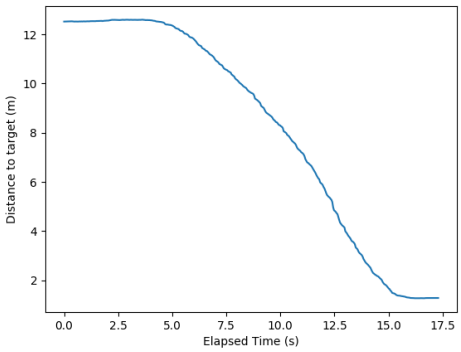


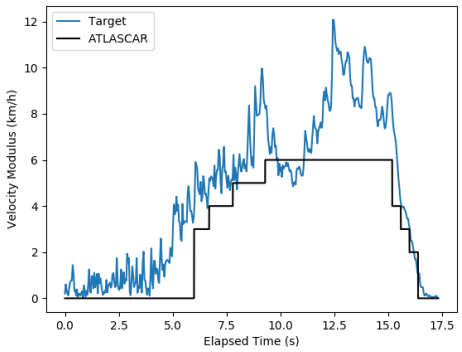
Figure 6.4: Image of the experiment of the After Turn Collision experiment on the top; On the bottom the `Rviz` representation of the target to be collided with in red with id number 0, the ATLASCAR2 short term path in blue and the instantaneous velocity vector in red.

Figure 6.5(a) shows the target distance and, similarly to the previous experiment, the distance is decreasing at an approximately constant rate after the initial acceleration.

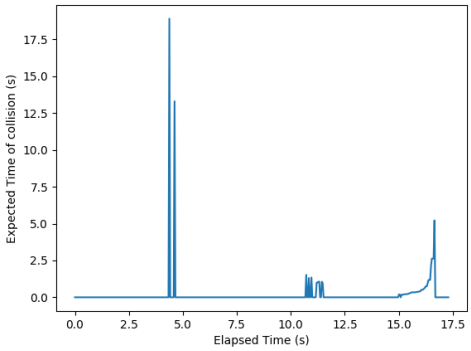
The velocity data is displayed on chart 6.5(b). The initial acceleration is clear until around the 6 second mark which after that the previously mentioned point oscillation is shown, this time however reaching a maximum velocity of 12.29 km/h at 12.44 seconds in the experiment, which translates to 4.96 meters away from the target. The ATLASCAR2 velocity data is similar as before stabilizing at 6 km/h. The overall target velocity average is 4.55 km/h mainly due to the initial static period, since the velocity average after this is 6.05 km/h. The ATLASCAR2 average velocity is 3.05 km/h in the full duration of the experiment and 4.67 km/h in same non-zero velocity period. This data again concludes the same error magnitude as the previous experiment.



(a) Target distance over time.



(b) Target velocity as detected by the GNN (in blue) and ATLASCAR2 velocity (in black) during the experiment.



(c) Target ETC over time.

Figure 6.5: Graphic representation of target distance, velocity and ETC data collected in a collision with a static target in the end of a 90° left turn.

In the matter of ETC calculation, the graphic 6.5(c) clearly shows the low 12% accuracy of the VO node. This conclusion was expected as the current VO principle is only to be applied in linear trajectories, and in this case it only detects collision in the final meters from the target.

6.1.3 Rotating Around Two Pedestrians

The main goal of the following experiment's was to study the effects of the apparent velocity since both static pedestrians did not face any risk of collision. The setup consisted of a static pedestrian placed in the car's turning center point and another placed outside the turn, while the car performed two 360° rotations around them. Its initial setup is displayed in Figure 6.6.

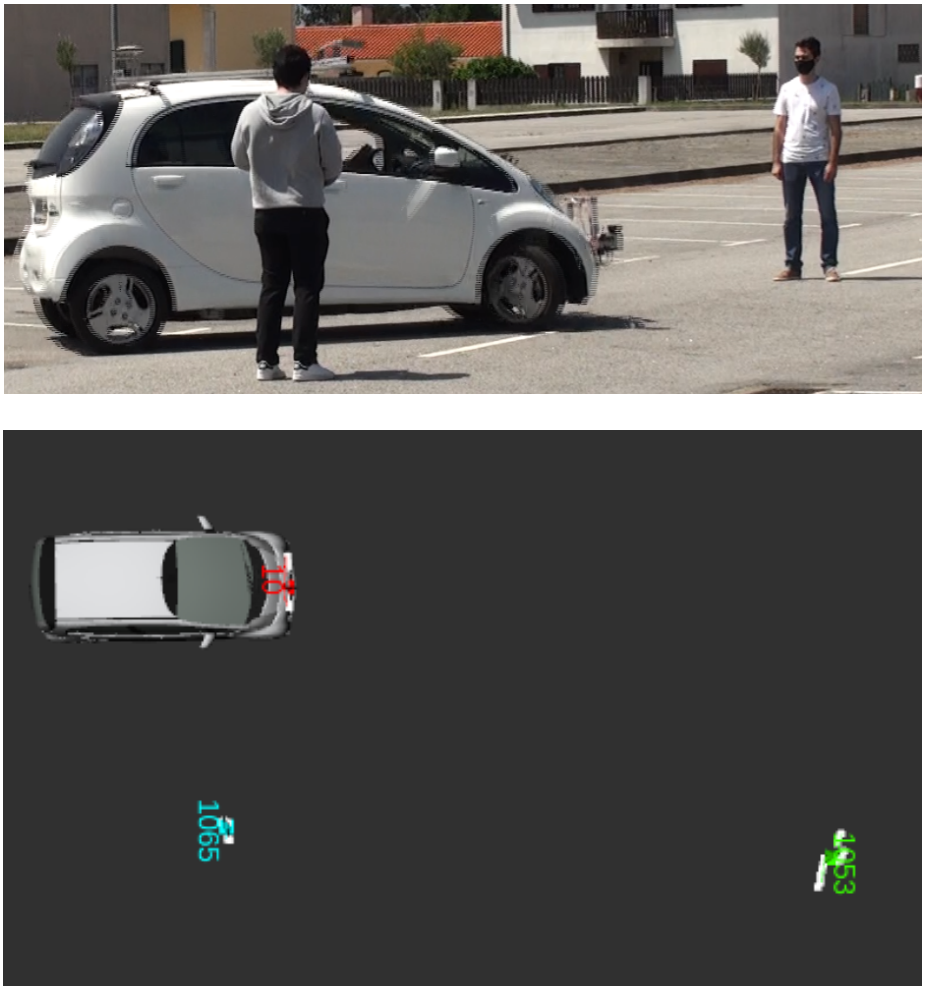


Figure 6.6: On the top an image of the Rotating Around Two Pedestrians experiment; On the bottom the `Rviz` representation . The target placed in the turning center point is represented in cyan with id 1065 and the pedestrian placed outside the turn in green with id number 1053.

ATLASCAR2 velocity was set at about 6 km/h. The collected data is divided into two sets of plots: the charts of Figure 6.8 represent the data related to the target placed on the car's turning center point (target id 1065 in 6.6), while the charts of Figure 6.9 display the data of the target placed outside the turn (target id 1053 in Figure 6.6). The latter however is divided into three sets of data, as the `GNN` node was not able to successfully track the pedestrian during the full duration of the experiment. When the ATLASCAR2 was facing away from the target the background measurements, such as

vegetation and the road curb, would move and embrace the pedestrian's target points making it impossible to distinguish it from the remaining points (Figure 6.7). This was due to target being close the LIDAR detection angle edge which caused the pedestrian id to be lost and then be assigned a new one when its points are found by the LIDAR. These sets of data were then merged to be displayed in a single graphic.

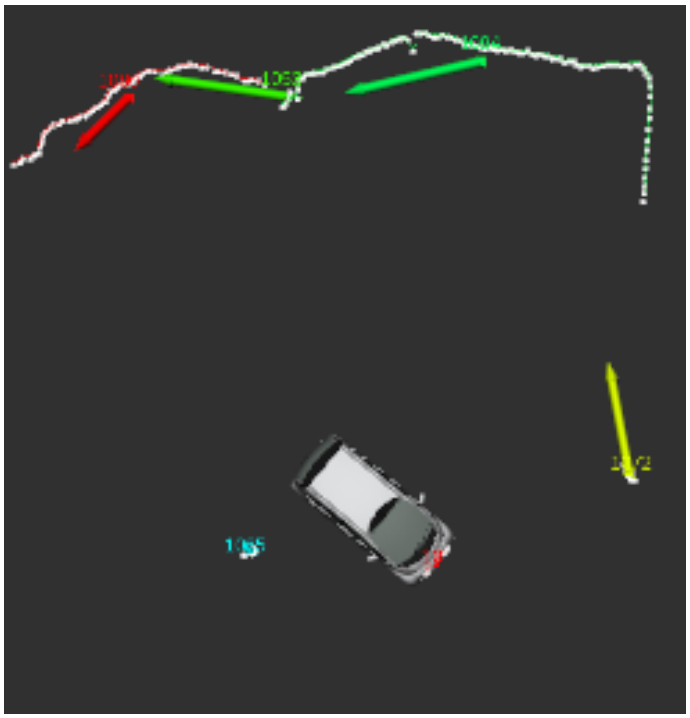
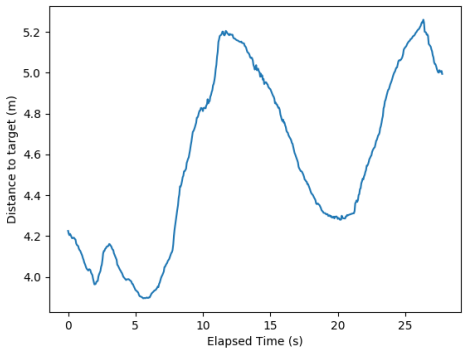


Figure 6.7: `Rviz` representation of the Rotating Around Two Pedestrians experiment error. The target placed in the turning center point is still being tracked with the same id 1065, however the target outside th8 turn (id number 1053) is being embraced by the background measurements, causing the target to merge with the remaining points and therefore lose the pedestrian tracking.

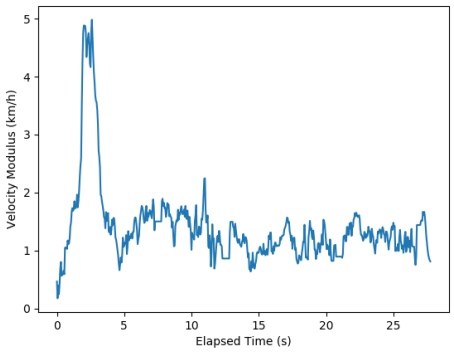
Following the previous presentations, plot 6.8(a) presents the distance of the target placed in the turning center point. Due to its placement, the target does not move, which confirms the statements of subsection 5.2.1, hence the small distance oscillation in the graphic. The initial distance values ($t < 5s$) are different due to the pedestrian adjusting its placement. The average distance is approximately 4.5 meters, which is the stated minimum turning distance of the car. Relative to the target velocity (graphic 6.8(b)) the pedestrian position adjustment is also detected in the same time period, otherwise it oscillates around the 1.42 km/h mark. As previously stated, this velocity should be perceived as 0 since the target points are not moving, these values must again be due to the centroid shift. Despite the target not being in risk of collision, the node detects it around 5% of the duration of the experiment. Due to its high expected time of collision, the same conclusion as before is acceptable.

Regarding the pedestrian placed on the outer side of the turn, graphic 6.9(a) shows the expected oscillation in the target distance denoting the data set switch in the points of highest distance to the car. Apart from that, the remainder of the data is in line

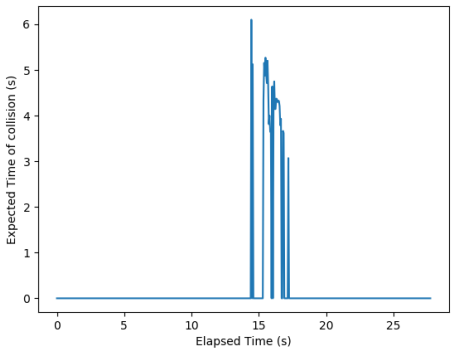
with what is expected. In terms of velocity, graphic 6.9(b) represents the magnitude of the apparent velocity influence on the target velocity. In comparison with the previous velocity graphic, the placement of the target when the ATLASCAR2 is turning is related to the inaccuracy of the GNN target velocity. In this case, the average target velocity is 19.87 km/h with a maximum of 30.55 km/h at the 21.25 second mark, or at 9.10 meters away from the car. These values are far from the car's real velocity, which indicates the extreme influence of the apparent velocity. Despite not confirmed, it is believed that the further the initial distance of the target to the car, the higher its detected velocity it would be.



(a) Target distance over time.



(b) Target velocity as detected by the GNN.



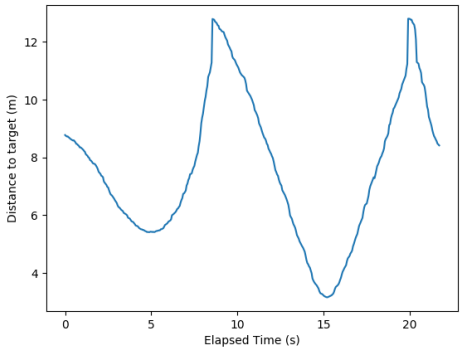
(c) Target ETC over time.

Figure 6.8: Graphic representation of distance, velocity and ETC data collected while rotating around a static target placed on the car's turning center point.

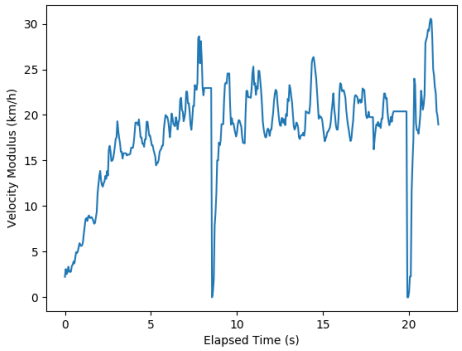
On the other hand, graphic 6.9(c) shows that during the experiment there was no collision detected which, in this case, is a good result as initially there was no risk to the pedestrian.

6.2 Dynamic Target

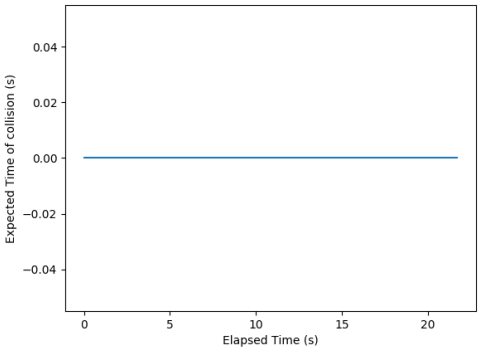
This section describes the experiments made with a moving target. Throughout all of the following experiments an effort was made to stabilize the target velocity to around



(a) Target distance over time.



(b) Target velocity as detected by the GNN.



(c) Target ETC over time.

Figure 6.9: Graphic representation of distance, velocity and ETC data collected while rotating around a static target placed on the outer side of the turn.

8 km/h. Even with some deviations from this velocity goal, this is the value considered for the target velocity while discussing the results. The main goal of the following experiments is to access the apparent target velocity while the ATLASCAR2 is performing linear and circular trajectories. Despite the fact that two vehicles were not intended to simulate a collision, the study of the ETC is still valid to understand how the architecture behaves with dynamic targets.

6.2.1 Pass By

In the following experiment both ATLASCAR2 and the passenger vehicle perform linear trajectories passing by each other. The evolution of the detected points throughout the experiment is shown in Figure 6.10. Initially, due to the GNN clustering architecture, the front of the incoming vehicle is detect by two sets of points, therefore two targets with its respective id (first image in Figure 6.10). As the target gets closer to the car, these points are merged into only one target still representing the front of the car (second image in Figure 6.10). When the passenger vehicle is side by side with the ATLASCAR2 the detected points now represent the side of the car (third image in

Figure 6.10). As the target drives away from the ATLASCAR2 the LIDAR now detects the back of the passenger vehicle (fourth image in Figure 6.10). This change of detected points throughout the experiment makes the target centroid move, in this case backwards thus affecting the GNN velocity data. Since no additional id was created throughout the experiment, the graphics in Figure 6.11 represent the data from the remaining target after the merge.

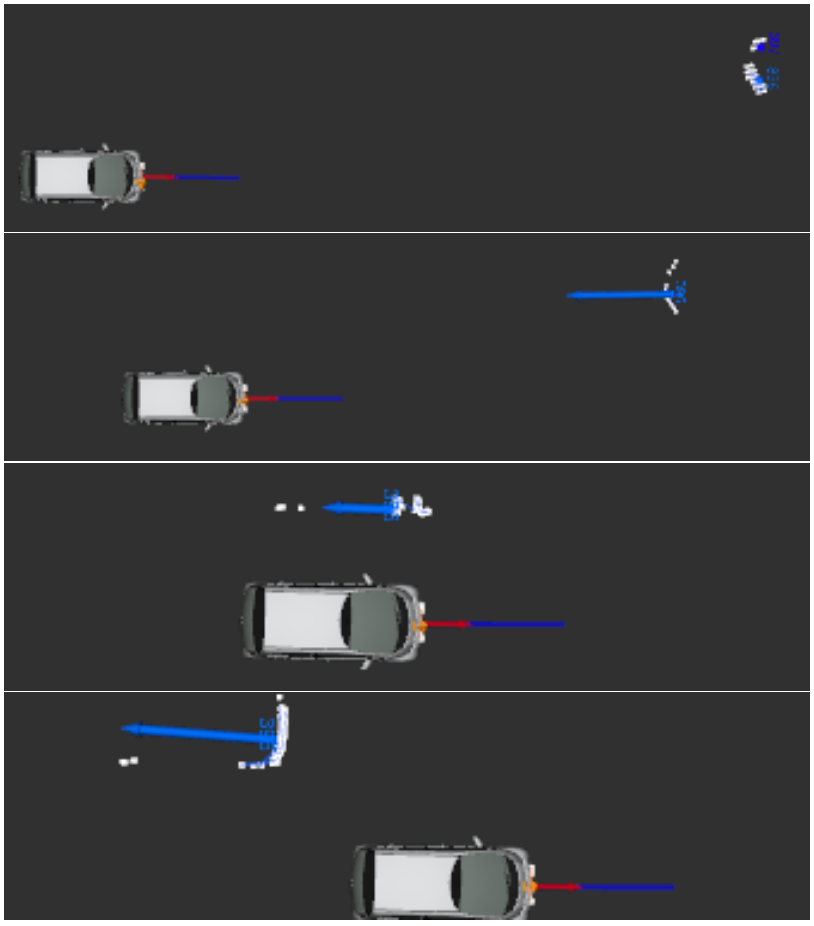
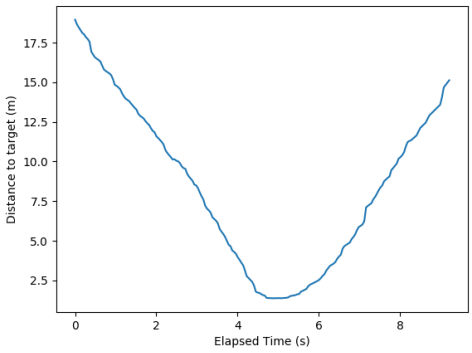


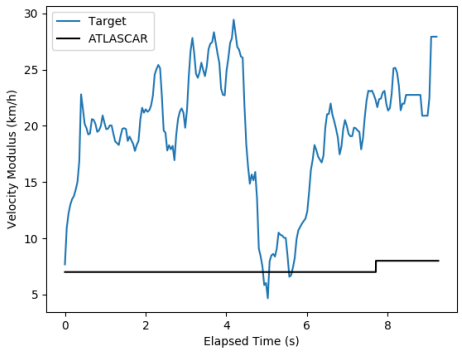
Figure 6.10: Rviz representation of the "Pass By" experiment. Initially the GNN clustering architecture identified the incoming vehicle as two sets of points representing the front of the vehicle, therefore two targets with respective ids and colors (cyan and blue). Afterwards one of the id is removed when the target is closer to the ATLASCAR2. When both vehicles are side by side, the LIDAR now only detects the side of the passenger vehicle. As the target drives away from the ATLASCAR2, the side of the passenger vehicle is no longer detected, but rather its trunk. This change in detected points makes the centroid move, in this case backwards, affecting the GNN velocity data.

As expected, graphic 6.11(a) accurately displays the distance of the target initially decreasing and then, after crossing the ATLASCAR2, increasing. The target and ATLASCAR2 velocity is displayed on graphic 6.11(b). Since the target is moving, the displayed data informs the relative velocity between the two. As the pair is moving in opposite directions (the angle between the two velocity vectors is approximately 180°),

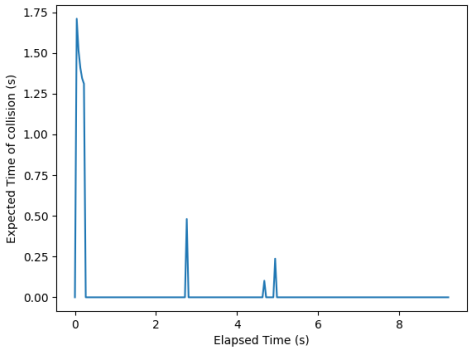
the ATLASCAR2 velocity is added to the target real velocity, as in subsection 5.2.1, thus the `GNN` target average velocity of 19.46 km/h when the ATLASCAR2 average velocity was 7.16 km/h. By subtracting these average velocities the data averages at 12.3 km/h, which is more approximate to the target's real velocity. By analysing the graphic it is clear that the target reaches its lowest velocity when closest to the ATLASCAR2. This is due to the change of detected points, previously mentioned. Apart from this, the average velocities error seems to increase for the dynamic target when compared to the static. Graphic 6.11(c) shows that the `VO` node detected a collision in several occasions throughout the experiment. This must again be due to the centroid shift of the target, as there was never a collision risk between the two and these detection are when a velocity spike occurs. Overall, the displayed data relates to, in this case, an approximately 4% inaccuracy.



(a) Target distance over time.



(b) Target velocity as detected by the `GNN` (in blue) and ATLASCAR2 velocity (in black) during the experiment.



(c) Target ETC over time.

Figure 6.11: Graphic representation of distance, velocity and ETC data collected while passing by a dynamic target moving in the opposite direction.

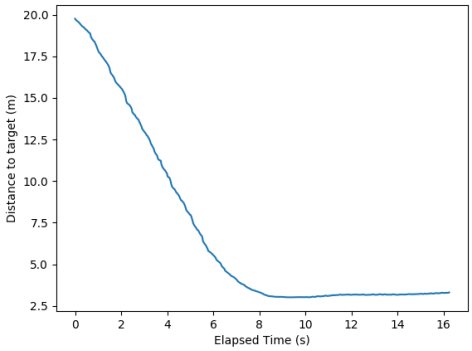
6.2.2 Intersection

The following experiments, divided into two separate road situations, simulate an intersection. Firstly, the ATLASCAR2 merges into a road where the passenger vehicle travels, to then drive side by side with the detected target. In the second experiment, the ATLASCAR2 merges again into the road where the passenger vehicle travels, this time driving in opposite directions. Its aim is to study the apparent velocity inflicted into the target in both situations. Initially, the collision detection is expected to be unreliable since, as stated previously, the developed architecture is not suited for circular trajectories.

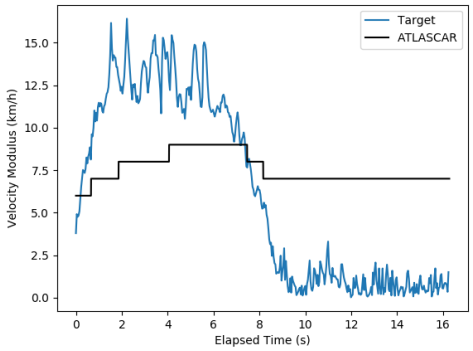
Same direction

As mentioned previously, Figure 6.12 displays the experiment where the ATLASCAR2 merges into a intersection. Figure 6.13 displays the graphics of the experiment where the ATLASCAR2 merges the road to then travel side by side with the target. As expected the distance displayed on graphic 6.13(a) shows the distance decreasing to stabilize at the closest distance to the target of 3 meters. Graphic 6.13(b) clearly denotes two situations regarding the velocity. As the ATLASCAR2 is moving perpendicular to the target, the `GNN` detected target velocity is higher; afterwards, as both travel side by side, the target velocity decreases to approximately 0 km/h. This a good example of the subtraction of both velocities vectors, since in the two cases they lie at different angles (> 0 in the first case and 0 in the last). The overall `GNN` average velocity was 6.24 km/h while for the ATLASCAR2 its average velocity was 7.55 km/h. The plot 6.13(c) displays the collision detected in the beginning of the experiment; according to this both vehicles would collide if the ATLASCAR2 did not change its course, which, despite not being tested, is a good assumption.

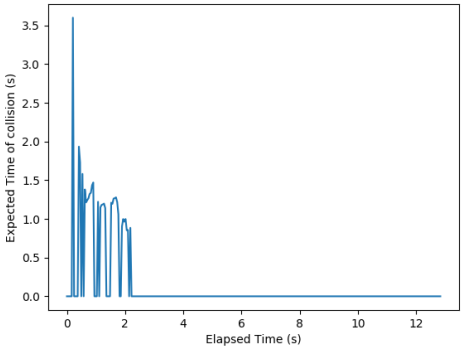
Figure 6.12: On the left an image of the experiment, in this case with a bicycle portraying the dynamic target; On the right the `Rviz` representation of the data collected using a passenger vehicle. The target is represented in pink with id 279, the ATLASCAR2 short term path in blue and the instantaneous velocity vector in red.



(a) Target distance over time.



(b) Target velocity as detected by the GNN (in blue) and ATLASCAR2 velocity (in black) during the experiment.



(c) Target ETC over time.

Figure 6.13: Graphic representation of distance, velocity and ETC data collected while the ATLASCAR2 merged into a road where a vehicle was traveling (in the same direction) after which driving side by side with the target.

Opposite direction

Regarding the second experiment, still studying the behaviour of the vehicles in an intersection, Figure 6.14 exemplifies the motion of the ATLASCAR2 as well as the Rviz representation. Charts 6.15 displays the data when the ATLASCAR2 merges into the opposite lane of the passenger vehicle. Graphic 6.15(a) presents the expected values for the distance of the target. Plot 6.15(b), in contrast with the previous intersection situation, shows the velocity to be always higher than the ATLASCAR2. Even though the difference is not as clear, the graphic still depicts two different velocity profiles. Initially, when the ATLASCAR2 is perpendicular to the target, its velocity is similar as previous experiment. On the other hand, when the ATLASCAR2 is driving away from the passenger car, its velocity increases, similar to the "drive by" experiment. The results show an overall GNN target velocity average of 17.11 km/h, and the ATLASCAR2 velocity average of 6.55 km/h. Analogous to the previous experiment, graphic 6.15(b) informs of an eventual collision if the ATLASCAR2 continued to drive forward.

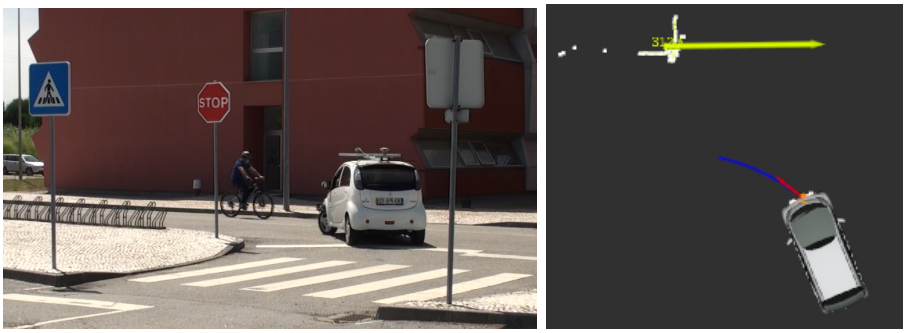
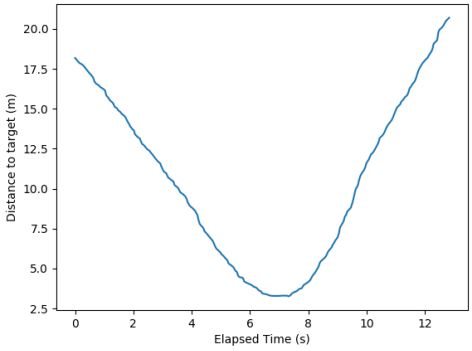
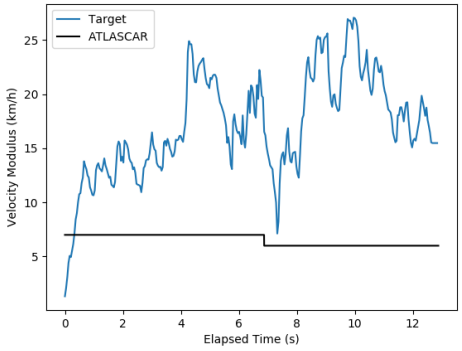


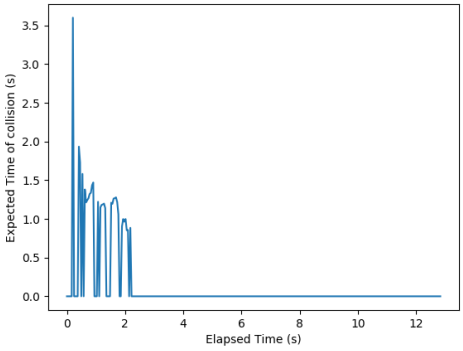
Figure 6.14: On the left an image of the experiment, in this case with a bicycle portraying the dynamic target; On the right the `Rviz` representation of the data collected using a passenger vehicle. The target is represented in yellow with id 312, the ATLASCAR2 short term path in blue and the instantaneous velocity vector in red.



(a) Target distance over time.



(b) Target velocity as detected by the `GNN` (in blue) and ATLASCAR2 velocity (in black) during the experiment.



(c) Target ETC over time.

Figure 6.15: Graphic representation of distance, velocity and ETC data collected while the ATLASCAR2 merged into a road where a vehicle was traveling (in the opposite direction).

6.3 Circle approximation

In the following segment three separate, yet similar, sideways collisions will be described. First, the ATLASCAR2 will be colliding with a stationary passenger vehicle perpendicularly placed, aiming for its middle section between the front and back doors. Afterwards, the same situation is studied, this time being the ATLASCAR2 static while the passenger vehicles performs the above described movement. Lastly, while the ATLASCAR2 is static and still perpendicular to the passenger vehicle, the latter will drive towards a collision with the ATLASCAR2 rear headlight. Its main goal is to study the collision detection algorithm of the `VO` node and the influence of the circle approximation radius on its accuracy. Despite not having velocity data from the passenger vehicle real velocity, it is assumed to be approximately 8 km/h.

6.3.1 ATLASCAR2

The initial setup of the following experiment is displayed in Figure 6.16 while the ATLASCAR2 drives towards the static passenger vehicle. As described previously the graphics displayed on 6.17 related to the first sideways collision with the moving ATLASCAR2 with the static passenger vehicle. Similar to the previous experiments, the graphics 6.17(a) and (b) depict the expected distance and velocity behaviours. Focusing on the graphic 6.17(c) displaying the ETC data, the overall good results of collision detection by the developed architecture can be seen. The node is able to detect the collision almost immediately, despite some minor miss detection afterwards, allowing the driver to safely adjust its course to avoid a collision. In terms of numbers, the collected data shows that the `VO` node detected a collision 71% of the time.

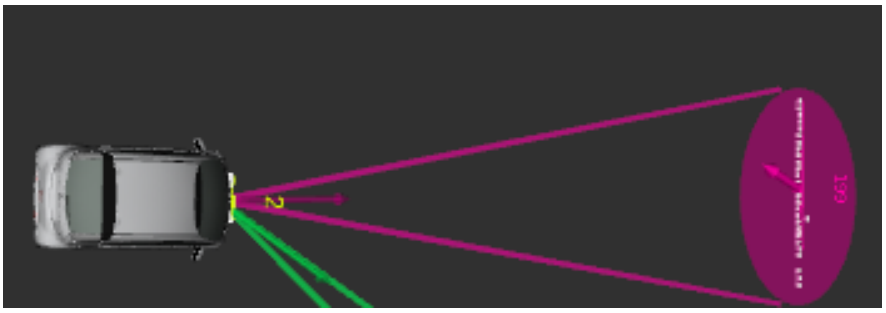
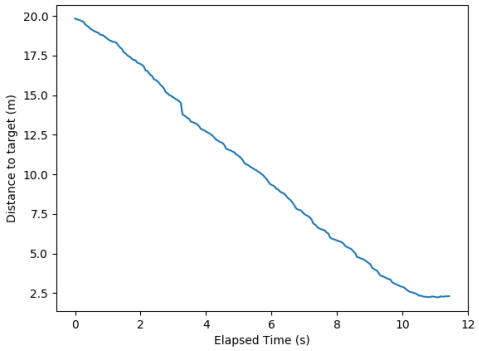


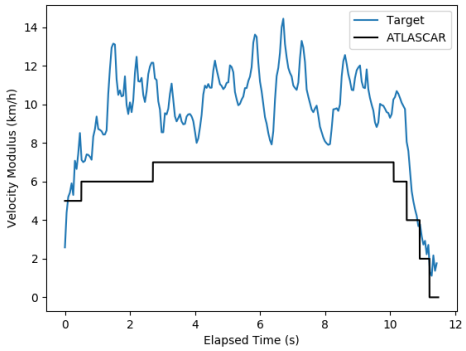
Figure 6.16: `Rviz` representation of the ATLASCAR2 sideways collision experiment with the target id 199. The collision cone can also be seen with the pink lines as well as the enlarged target circle.

6.3.2 Passenger vehicle

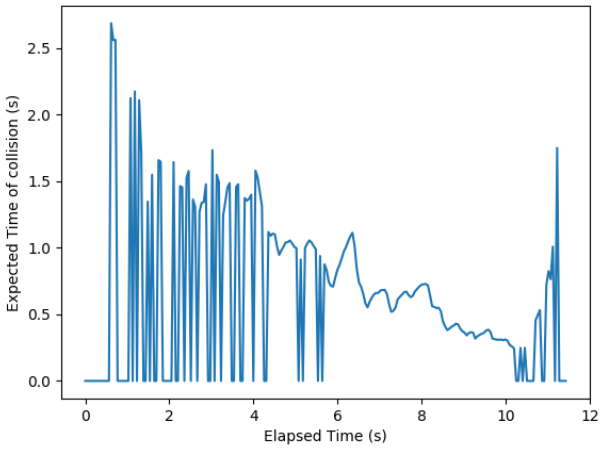
Regarding the second part of this sideways collision experiment, the graphics to be analysed are presented in Figure 6.19. Graphic 6.19(b) displays the already known static velocity of the ATLASCAR2 in black, while in blue the target velocity is considered to be a good approximation to the passenger vehicle velocity. This target velocity, despite no accurate measurement of the vehicle's real speed, is in line with the data gathered from the experiment in subsection 6.1.1, where in both situations the velocity



(a) Target distance over time.



(b) Target velocity as detected by the GNN (in blue) and ATLASCAR2 velocity (in black) during the experiment.



(c) Target ETC over time.

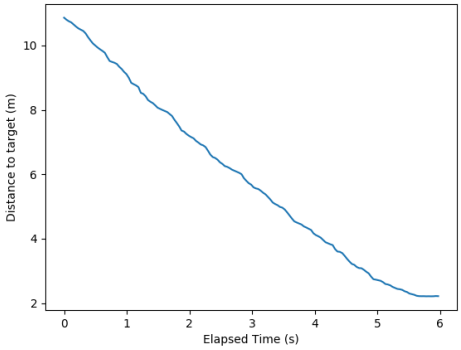
Figure 6.17: Graphic representation of distance, velocity and ETC data collected in a collision simulation between the ATLASCAR2 and a stationary passenger vehicle perpendicularly placed.

profile is approximate. The graphic displayed in 6.19(c), in contrast to the first part of this experiment, shows a much poorer collision detection, even thought the colliding vehicle movement in both is the same, translating to a 35% decrease in accuracy (36% collision detection accuracy for this case). This poor accuracy has to do with the circle approximation and the location of the VO cone apex point. As mentioned in previous chapters, the ATLASCAR2 is approximated to a circle of radius equal to its width and the VO cone apex point is locate at the front of the vehicle. Since in this case the colliding point is not at the front of the vehicle, but rather on the side of the car, the VO node does not detect a collision between the two. Additionally, the collision is not detected due the fact that the ATLASCAR2 circle radius is smaller than the distance, in the car's x axis, between the VO cone apex point and the target centroid. If the ATLASCAR2

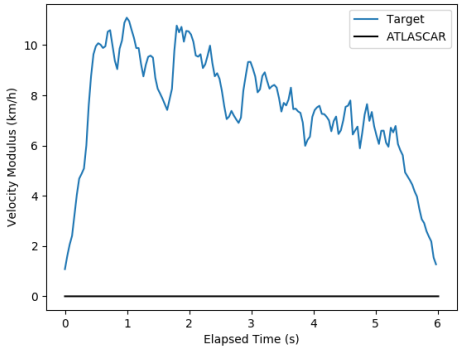
width is enlarged the collision would in fact be detected as shown in Figure 6.18.

Figure 6.18: ATLASCAR2 circle radius comparison between two situation in the approximately same position. On the left the ATLASCAR2 circle radius is 3 meters so a collision is detected. On the right its radius is 1.5 meters and a collision is not detected.

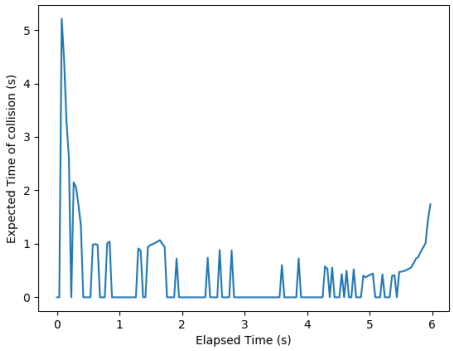
Maintaining the initial ATLASCAR2 circle width, if the passenger vehicle would aim to collide further back in the ATLASCAR2, its detection would be even lower as the graphic 6.20(c) shows with a 1% collision detection accuracy.



(a) Target distance over time.

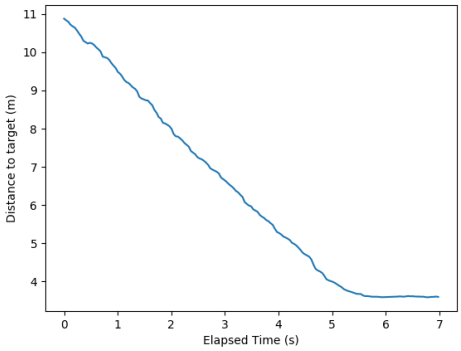


(b) Target velocity as detected by the `GNN` (in blue) and ATLASCAR2 velocity (in black) during the experiment.

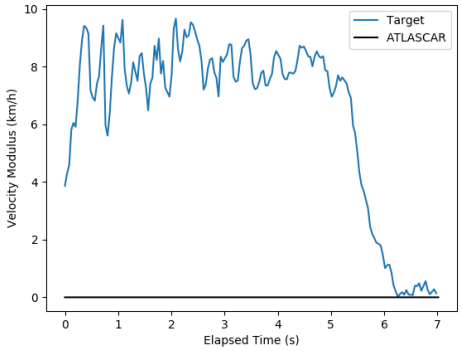


(c) Target ETC over time.

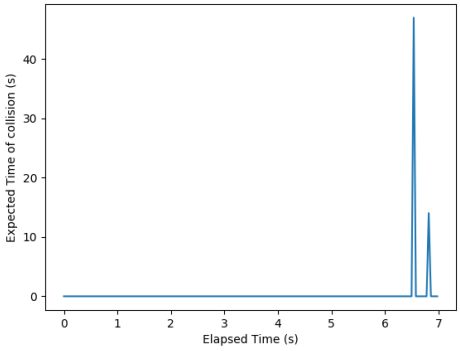
Figure 6.19: Graphic representation of distance, velocity and ETC data collected in a collision simulation between the passenger vehicle and the ATLASCAR2 placed perpendicular to it.



(a) Target distance over time.



(b) Target velocity as detected by the GNN (in blue) and ATLASCAR2 velocity (in black) during the experiment.



(c) Target ETC over time.

Figure 6.20: Graphic representation of distance, velocity and ETC data collected while the passenger vehicle drove towards a tail collision with the ATLASCAR2.

6.4 Overall Result Analysis

In conclusion, Table 6.1 shows the results of the collision detection percentage and the modulus of the difference between the average velocity of the GNN target and the ATLASCAR2, of the static target experiments. As expected, the collision detection is more accurate when the ATLASCAR2 performs a linear trajectory, as in the "Front collision" experiment, compared to the "After Turn Collision". This proves the unreliability of this architecture when applied to circular trajectories, as a collision is only detected when the ATLASCAR2 is within a few meter to the target. The average velocity difference for both experiments demonstrates the centroid shift error, since the GNN target velocity is expected to be equal to the ATLASCAR2 velocity. These errors are similar to the average target velocity when the ATLASCAR2 is static (0.73 km/h in the initial stage of the "Front Collision" experiment). Regarding the "Rotating Around Two Pedestrians" experiment, the difference between the GNN and the ATLASCAR2 velocity clearly demonstrates the inaccuracy of the GNN architecture to detect the target velocity, which

is affected based on the distance of the target to the car's turning center point. This error is due not only to the centroid shift, experienced in the inner pedestrian, but also to the apparent velocity influence on the outer pedestrian.

Table 6.1: Static Target experiment results of the collision detection percentage and the modulus of average velocity difference between the GNN target velocity and the ATLASCAR2 velocity.

Static Target		
	Collision detection (%)	|Average velocity difference| (km/h)
Front collision	65%	1.46
After turn collision	12%	1.38
Rotating around two pedestrians - Outer pedestrian	5%	5.58
Rotating around two pedestrians - Inner pedestrian	0%	12.87

Regarding the Dynamic Target experiments, Table 6.3 display the results obtained. Since no collision was expected and the motion of the ATLASCAR is similar to the previous static experiments, only the average velocity difference is to be analysed. The overall results demonstrate the influence of the angle between the ATLASCAR2 velocity vector and the GNN target velocity vector. Throughout the "Pass by" experiment both vectors were at an approximately 180° angle, thus its difference resulting in an higher velocity modulus. The average velocity however was affected by the detected passenger vehicle points, as when the incoming car passed by the ATLASCAR2 its target points were no longer related to the front of the vehicle, but rather its side and afterwards the back of the car, as it drove away. This change resulted in a momentarily decrease in the target velocity as the centroid made this movement from the front to the back of the passenger vehicle (Figure 6.10).

The "Intersection" experiment was divided in two separate scenarios, "Same direction" and "Opposite direction". These were meant to illustrate the difference in the GNN velocity as the ATLASCAR2 drove parallel to the passenger vehicle and, in the second part of the experiment, while it drove away from the passenger vehicle. Therefore, the average velocity difference clearly demonstrates the influence of the angle between the GNN target and the ATLASCAR2 velocity vectors. Initially this angle is the same in both experiments. In the "Same direction" experiment, after merging into the road, the angle was approximately 0° thus their velocity modulus were subtracted, resulting in a low average velocity difference. On the other hand, in the "Opposite direction" experiment the angle was approximately 180° resulting in a higher velocity difference, as both velocity modulus were added.

Finally, the last set of experiments test the collision detection accuracy in a sideways collision. In the first scenario the ATLASCAR2 simulated a collision with a static passenger vehicle placed perpendicular to it. In the second scenario, the passenger vehicle is the one who collides with the static ATLASCAR2. Lastly, the previous scenario is ex-

Table 6.2: Dynamic Target experiment results of the modulus of the average velocity difference between the GNN target velocity and the ATLASCAR2 velocity.

Dynamic Target	
	|Average velocity difference| (km/h)
Pass by	12.3
Intersection - Same direction	1.31
Intersection - Opposite direction	10.59

perimented, this time the passenger vehicle simulating a collision with the ATLASCAR2 rear headlight. These results demonstrate the influence of the circle approximation radius of the ATLASCAR2, as in the last two experiments, by targeting the rear end of the ATLASCAR2, the collision detection accuracy decreased 35%, resulting in an extremely dangerous 1% accuracy. This is due to the fact that the current architecture expects the ATLASCAR2 to collide front first with an object rather being collided with on the side. This error, however, can be decrease by enlarging the ATLASCAR2 circle, adding a safer margin to detect a collision.

Table 6.3: Circle approximation experiment results of the detected collisions percentage.

Circle approximation	
	Collision detection (%)
ATLASCAR2	71
Passenger vehicle	36
Passenger vehicle - rear headlight	1

Chapter 7

Conclusion and Future Work

7.1 Conclusions

Before a vehicle is prepared to drive midst the chaotic road environment with no input from the driver, it must be able to ensure maximum safety for its user, as well as everyone surrounding it. For that, a collision prediction algorithm, such as the one studied in this dissertation, must perform with the highest accuracy.

There are several different techniques to detect a collision between two objects and, in this case the proposed approach was to use LIDAR detected velocity data and target properties, such as its dimensions. Consequently, the developed architecture heavily relies on this data, which, in this dissertation, was provided by the ROS package `MTT`, specifically the `GNN` node, previously developed at Universidade de Aveiro. The code in question was refurbished to allow its execution in more recent versions of ROS and provide additional data needed for the remaining architecture. Initially, the developed ROS package was intended for detection and tracking of objects while its LIDAR is static, resulting in very good performances. However, the deployment in dynamic environments, as the LIDAR accompanied the ATLASCAR2 attached to the front bumper, led to a study of the influence of the ego motion inflicted apparent velocity, specifically while the car is turning. From this study also resulted an auxiliary ROS node capable of representing the short term path of the ATLASCAR2.

By comparing the velocity data from a known static target with the collected data from the ATLASCAR2 velocity, the study concluded that without any ego motion compensation, the `GNN` node is unable to successfully determine the target's real velocity nor the relative velocity between the target and the ATLASCAR2. Regarding dynamic targets, despite being clear the relative velocity changes based on the angle between both target and ATLASCAR2 velocity vectors, the same error is presented. A possible reason for this is the target centroid constant shift due the `GNN` point clustering architecture which would add false velocity to the target. The erroneous velocity data compromises the ETC calculations since it is purely based on distance and velocity. The mentioned centroid shift is also responsible for decreasing the collision detection accuracy, since in cases where the centroid shifts sideways it may cause the relative velocity vector to lie outside the VO cone. Despite this setback, the `VO` node behaved as expected with good collision detection performances in linear trajectories. The VO technique applied was only viable to these linear trajectories so the experiments made proved the expected poor accuracy in circular trajectories. The conducted experiments allowed also to verify the

VO node circle approximation, which proved to influence the collision detection accuracy. In this case, it was initially set that the section of the car, which would be expected to collide, was the front of the vehicle, so the cone was placed in the middle front section of the car and the car's width as the radius of the circle. With this setup the VO node was able to detect a collision between the front bumper of the ATLASCAR2 and every target on the road. However, in the case where a collision was to happen with the side of the car the node was not as accurate, decreasing as the collision point was further back in the ATLASCAR2. The behaviour is expected to be similar in cases where the colliding target dimension are not well defined thereby reducing the collision detection algorithm performance.

In conclusion, the collision detection algorithm performed as expected when accounting for the known vulnerabilities. The developed architecture was able to accurately detect collisions in linear trajectories, as in a front collision or in an intersection. On the other hand, the correct localization of where and when the collision is going to occur, with the current setup, is not reliable as it depends on the correct velocity definition. The usage of LIDAR to detect and track targets has some disadvantages, such as the limited definition of dimensions due to the lack of target information, which in certain cases could reduce the algorithm's performance. This and several other constraints, while using the current setup, were also addressed in this dissertation, proving a good start up point if addressing collision detection using LIDAR. Regarding the initial objectives of this dissertation, the migration of previously developed tools to the most recent version of ROS was accomplished, thus allowing to use them in their full capabilities. The auxiliary visualization tool to display the ATLASCAR2 short term path was also proved useful to understand the behaviour of the car in the short term. The ultimate goal to determine the collision zones can be divided into two parts: detection of the imminent collision and definition of the location of said collision. Based on the results for the front collision, the first can be considered successful, for the current setup. The definition of the location of a collision, due to the high dependency on the correct detection of the target velocity, as greater margin of improvements.

7.2 Future Work

In the pursuit of maximum road safety, collision detection architectures as the one developed in this dissertation can always improve. These improvements can be separated in different areas such as object detection and collision detection.

In the target detection algorithm, results show that an improvement is needed to better determine the exact location of the eventual collision and also to improve the ETC accuracy. The usage of LIDARs alone to perceive the ATLASCAR2 surroundings is not sufficient to get the full dimensions of the obstacles, thus reducing the collision detection accuracy. Pairing the 2D LIDARs used in this dissertation with, for example, cameras and computer vision algorithms would allow to obtain the obstacles full dimensions or even distinguish it from different categories such as pedestrians, bicycles, heavy trucks or passenger vehicles. This target characterization is essential as for heavy trucks and passenger vehicles the detected LIDAR points is similar despite their dimensions and driving behaviour is completely different, influencing the collision results. With the target dimensions fully defined it can then be tracked using a bounding box which

reduces the mentioned centroid shift error, increasing the `GNN` target velocity accuracy, therefore improving the ETC calculation. The target detection and tracking algorithm is the base of any collision detection algorithm hence the importance of improving the architecture used in this dissertation.

In the collision aspect of this work, the implementation of a additional or improved algorithm to account for circular trajectories of both the target and the ATLASCAR2 is a must. The implementation of Vehicle to Vehicle communication which allows cars to share their status with other road agents is certainly going to facilitate this process to determine when a target is turning and its short term path, to then act accordingly. Additionally, the current setup allowed to point out several occasions where the circle approximation, of both the colliding target and the ATLASCAR2, reduced the collision detection, thus jeopardizing road users. The mentioned bounding box implementation would aid this circle approximation, as all dimensions of the target can now be accounted for.

Finally, contextual properties such as weather or road condition can also be accounted for when calculating the ETC. These properties change drivers behaviours and should also play a role in the autonomous vehicle calculations. Increasing the ATLASCAR2 circle approximation based on these factors can be considered as a first step in providing a safer journey in these conditions. By doing so, the collision calculations would have a safer margin, allowing the ATLASCAR2 to change its course if an obstacle was to pass too close to the car.

Intentionally blank page.

References

[1] *J3016B: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles - SAE International*. URL: https://www.sae.org/standards/content/j3016_201806/ (visited on 02/27/2020).

[2] M. Mohanan and A. Salgoankar. "A survey of robotic motion planning in dynamic environments". In: *Robotics and Autonomous Systems* 100 (Feb. 2018), pp. 171–185. DOI: 10.1016/j.robot.2017.10.011.

[3] J. Almeida. "Target tracking using laser range finder with occlusion". MA thesis. Universidade de Aveiro, 2010.

[4] D. Figueiredo. "Controlo Remoto para a Operação e Condução do ATLASCAR2". MA thesis. Universidade de Aveiro, 2020.

[5] R. Azevedo. "Sensor Fusion of LASER and Vision in ActivePedestrian Detection". MA thesis. Universidade de Aveiro, 2014.

[6] N. Silva. "Semi-Automatic Labelling and Tracking of Targetsfor Autonomous Driving". MA thesis. Universidade de Aveiro, 2018.

[7] P. Fiorini and Z. Shiller. "Motion planning in dynamic environments using the relative velocity paradigm". In: *Proceedings IEEE International Conference on Robotics and Automation*. May 1993, pp. 560–565. DOI: 10.1109/ROBOT.1993.292038.

[8] P. Fiorini and Z. Shiller. "Time optimal trajectory planning in dynamic environments". In: *Proceedings of IEEE International Conference on Robotics and Automation*. Vol. 2. Apr. 1996, pp. 1553–1558. DOI: 10.1109/ROBOT.1996.506925.

[9] P. Fiorini and Z. Shiller. "Motion Planning in Dynamic Environments Using Velocity Obstacles". In: *The International Journal of Robotics Research* 17.7 (July 1998), pp. 760–772. DOI: 10.1177/027836499801700706.

[10] B. Damas and J. Santos-Victor. "Avoiding moving obstacles: the forbidden velocity map". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2009, pp. 4393–4398. DOI: 10.1109/IROS.2009.5354210.

[11] D. Wilkie, J. van den Berg, and D. Manocha. "Generalized velocity obstacles". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2009, pp. 5573–5578. DOI: 10.1109/IROS.2009.5354175.

[12] J. van den Berg, J. Snape, S. Guy, and D. Manocha. "Reciprocal collision avoidance with acceleration-velocity obstacles". In: *IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 3475–3482. DOI: 10.1109/ICRA.2011.5980408.

[13] Z. Gyenes and E. Szadeczky-Kardoss. "Motion planning for mobile robots using the safety velocity obstacles method". In: *19th International Carpathian Control Conference (ICCC)*. May 2018, pp. 389–394. DOI: 10.1109/CarpathianCC.2018.8473397.

[14] M. Rapp, M. Barjenbruch, M. Hahn, J. Dickmann, and K. Dietmayer. "Probabilistic ego-motion estimation using multiple automotive radar sensors". In: *Robotics and Autonomous Systems* 89 (Mar. 2017), pp. 136–146. DOI: 10.1016/j.robot.2016.11.009.

[15] J. Almeida. "Active Tracking of Dynamic Multivariate Agents using Vectorial Range Data". PhD thesis. Universidade de Aveiro, 2016.

[16] A. Aguilar-González, M. Arias-Estrada, F. Berry, and J. Osuna-Coutiño. "The fastest visual ego-motion algorithm in the west". In: *Microprocessors and Microsystems* 67 (June 2019), pp. 103–116. DOI: 10.1016/j.micpro.2019.03.005.

[17] V. Santos, J. Almeida, E. Ávila, D. Gameiro, M. Oliveira, R. Pascoal, R. Sabino, and P. Stein. "ATLASCAR - technologies for a computer assisted driving system on board a common automobile". In: *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, Sept. 2010, pp. 1421–1427. DOI: 10.1109/ITSC.2010.5625031.

[18] V. Santos. "ATLASCAR: A Sample of the Quests and Concerns for Autonomous Cars". In: *Informatics in Control, Automation and Robotics*. Springer International Publishing, 2020, pp. 355–375. DOI: 10.1007/978-3-030-11292-9_18.

[19] T. Lozano-Perez. "Spatial Planning: A Configuration Space Approach". In: *IEEE Transactions on Computers* C-32.2 (Feb. 1983), pp. 108–120. DOI: 10.1109/TC.1983.1676196.

[20] M. Abe. *Vehicle Handling Dynamics - Theory and Application*. 2nd ed. Butterworth-Heinemann, 2015.

Appendix A

Instructions to Install and Run the Developed Packages

The packages developed in this dissertation were created in the ROS Melodic version. To download this framework follow the instructions on `https://wiki.ros.org/melodic/Installation/Ubuntu`. As of right now, additional versions of ROS were made available, however these were not tested and compatibility issues may rise.

After setting up the catkin workspace by following the initial ROS tutorials available in `https://wiki.ros.org/ROS/Tutorials`, github clones are necessary to download the needed packages. The auxiliary packages (`Colormap` and `canReceiveandUpdateStatus`) are available on `https://github.com/lardemua`. The core `VO`, `GNN` and `Short_Term_Path` nodes are available on `https://github.com/ruipcosta`, however a branch of this repository is expected be made to LARDEMUA. All the packages must be inside the `/src` folder within the previously made catkin directory.

To compile, within the catkin directory, do *catkin_make*.

To launch the ATLASCAR2 LIDARs, first start up the car, afterwards, in the Atlas machine, launch the sensor by running `roslaunch atlas2_bringup bringup.launch`. Check if the LIDAR points are being represented on `Rviz`. Afterwards, if the data is to be recorded, in another terminal run `roslaunch atlas2_bringup record_sensor_data.launch`.

To execute the full collision detection architecture (`VO` and `GNN`) run `roslaunch velocityobstacle vo.launch show:=1`; if `Rviz` is not intended to initiate, substitute the "1" to "0" in `show:=1`.

To run the ATLASCAR2 `Short_Term_Path` node, the `canReceiveandUpdateStatus` node must be initiated to publish the *NominalData* topic. This node however requires hardware outside of this dissertation topic. If the topic is being published, simply run the command `rosrun velocityobstacle ShortTermPath.py` to begin the node.