Universidade de Aveiro
2024

Rafael Filipe Bastos Oliveira

Mapas de risco baseados em visão e LiDAR para condução autónoma e apoio à condução no ATLASCAR2

Risk maps based on vision and LiDAR for autonomous driving and driving assistance in ATLASCAR2

Universidade de Aveiro
2024

Rafael Filipe Bastos Oliveira

Mapas de risco baseados em visão e LiDAR para condução autónoma e apoio à condução no ATLASCAR2

Risk maps based on vision and LiDAR for autonomous driving and driving assistance in ATLASCAR2

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado com Agregação, do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Esta dissertação foi desenvolvida 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 José Paulo Oliveira Santos
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee

Doutor Hugo Filipe Costelha de Castro
Professor Coordenador da *Escola Superior de Tecnologia e Gestão - Instituto Politécnico de Leiria*

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

Agradecimentos / Acknowledgements

Em primeiro lugar, gostaria de agradecer à minha família por acreditarem sempre em mim e me proporcinarem as condições necessárias para alcançar este objetivo.

Ao Prof. Doutor Vítor Santos, agradeço pela orientação e ajuda, especialmente pela paixão e entusiasmo que me transmitiu ao longo do semestre. Obrigado pela sua disponibilidade e por me ajudar a manter o foco no que realmente importava.

Ao Eng. Rui pela sua ajuda, sem ela, não teria tido oportunidade de dar umas voltas no ATLASCAR2! Agradeço também ao Gonçalo, meu colega do lado, pelos ensinamentos de C++, pela companhia e pela disponibilidade para ajudar desde o início. À restante malta do LAR, em especial ao Pedro, Manuel e Joel, que sempre se mostrarm disponíveis para trocar ideias e ajudar.

Agradeço ao Zé pela constante motivação e dicas de Python. Agradeço também ao Ricardo pela companhia e pelos momentos de desabafo.

Por fim, gostaria de agradecer aos meus amigos de Lamego pela sua existência singular e apoio imensurável.

Keywords Obstacle detection, Obstacle classification, Object tracking, Point Cloud segmentation, Sensor calibration, Panorama stitching, Collision risk, Occupancy prediction.

Abstract

Traffic accidents are a major global health concern, causing millions of deaths and injuries every year. This highlights the urgency of addressing and enhance road safety. ADAS and Autonomous driving are emergent technologies that have the potential to significantly reduce accidents by leveraging efficient perception systems.

This dissertation opens the door for autonomous driving and ADAS within the ATLASCAR2 by enhancing its perception capabilities through the installation of a 3D LiDAR and the development of a new framework named Risk Maps that analyzes and transmits collision risks between the ATLASCAR2 and other road agents, namely pedestrians and vehicles.

Risk Maps are based on the obstacle's instantaneous properties, namely speed, acceleration, size, nature and orientation as well as the ATLASCAR2's speed. This involves addressing concepts like obstacle detection and classification, point cloud segmentation, sensor calibration, image stitching, and the combination of camera and LiDAR data.

This dissertation showcases the challenges encountered during the creation of Risk Maps and the decisions made to overcome them. It tests the proposed solution in both simulated and real world environments, addressing common urban scenarios such as roundabouts, junctions, crosswalks, and lane passing. The collision risk is transmitted before the potential collisions, thereby enhancing the Risk Maps ability to provide actionable insights for improving road safety.

Palavras-chave

Deteção de obstáculos, Classifcação de obstáculos, Rastreamento de objectos, Segmentação de nuvens de pontos, Calibração de sensores, Imagem panorama, Risco de colisão, Previsão de ocupação.

Resumo

Os acidentes rodoviários são uma das principais preocupações na saúde a nível mundial, que causa milhões de mortes e lesões graves todos os anos. Estes factos destacam a urgência de abordar e melhorar a segurança na estrada. Os sistemas de auxílio ao condutor (ADAS) e a condução autonoma são tecnologias que têm vindo a crescer consideravelmente nos últimos anos, e possuem o potencial de reduzir significativamente os acidentes na estrada através do uso de sistemas de perceção eficientes.

Esta dissertação abre as portas para a condução autonóma e ADAS no ATLASCAR2, melhorando as suas capacidades de perceção através da instalação de um sensor LiDAR 3D, e do desenvolvimento de uma nova infraestrutura, designada "Mapas de Risco", que analisa e transmite o risco de colisão entre o ATLASCAR2 e outros participantes na cena, tais como peões e veículos.

Os "Mapas de Risco" são baseados nas propriedades instantâneas dos obstáculos, nomeadamente a sua velocidade, aceleração, dimensões, natureza e orientação, e também na velocidade do ATLASCAR2. A obtenção destas propriedades envolve conceitos tais como, deteção e classificação de objetos, segmentação de nuvens de pontos, calibração de sensores, criação de panoramas e combinação da informação das camaras com a do LiDAR.

Esta dissertação demonstra os desafios encontrados ao longo da criação dos "Mapas de Risco" e as decisões tomadas para os ultrapassar. A solução proposta é testada tanto em ambientes de simulação como em ambientes da vida real, abordando cenários comuns em zonas urbanas tais como rotundas, passadeiras, cruzamentos e passagens por veículos na via contrária. O risco de colisão é transmitido antes de uma potencial colisão, demonstrando assim a capacidade dos "Mapas de Risco" de fornecer informações úteis para melhorar a segurança rodoviária.

Contents

1 Introduction **1**
- 1.1 ATLAS project . . . 1
- 1.2 Problem description . . . 2
- 1.3 Objectives . . . 2
- 1.4 Document structure . . . 3

2 State of the Art **5**
- 2.1 ATLASCAR2 . . . 5
- 2.2 Obstacle detection with 3D LiDAR . . . 5
- 2.3 Obstacle detection and classification with cameras . . . 8
- 2.4 Calibration . . . 12
- 2.5 Combination of LiDAR with Camera . . . 12
- 2.6 Collision risk . . . 13
  - 2.6.1 CMCDOT (Conditional Monte Carlo Dense Occupancy Tracker) . . . 14
  - 2.6.2 Occupancy Maps . . . 15
  - 2.6.3 HARA . . . 16
  - 2.6.4 Risk Repulsion . . . 17
  - 2.6.5 SSM . . . 17
  - 2.6.6 Bayesian Network . . . 17
  - 2.6.7 BConvLSTM (Bayesian Convolutional Long Short Term Memory) . . . 18
  - 2.6.8 Situation-Aware Dynamic Risk Assessment (SINADRA) . . . 18
  - 2.6.9 Risk assessment from V2V framework . . . 20
- 2.7 Collision damage . . . 20
- 2.8 Summary . . . 23

3 Experimental infrastructure **25**
- 3.1 Hardware . . . 25
  - 3.1.1 ATLASCAR2 . . . 25
  - 3.1.2 Asus ROG Strix G15 G512LI-70AT5PB1 laptop . . . 27
- 3.2 Software . . . 28
  - 3.2.1 Robot Operating System . . . 28
  - 3.2.2 ROS Visualization . . . 28
  - 3.2.3 Gazebo Simulator . . . 29
- 3.3 3D LiDAR implementation and setup in the ATLASCAR2's infrastructure 29

4 Proposed solution **33**
4.1 Collision Risk method . . . 33
4.2 Perception . . . 39
4.2.1 LiDAR . . . 39
4.2.2 Camera . . . 41
4.2.3 Camera-LiDAR data combination . . . 45
4.3 Risk Maps . . . 49

5 Tests and Results **59**
5.1 Simulation scenarios . . . 59
5.1.1 Junction scenario . . . 60
5.1.2 Opposite lane pass scenario . . . 62
5.1.3 Lane collision scenario . . . 63
5.2 Crosswalk approach scenario . . . 64
5.3 Software configuration for real data collection . . . 66
5.3.1 Calibration . . . 66
5.3.2 Stitch . . . 70
5.3.3 LiDAR object detection and clustering . . . 71
5.3.4 Sensor synchronization . . . 72
5.3.5 ATLASCAR2 velocity . . . 72
5.4 Real world scenarios . . . 73
5.4.1 Roundabout approach scenario . . . 73
5.4.2 Opposite lane pass scenario . . . 74
5.4.3 Junction approach scenario . . . 77
5.4.4 Crosswalk approach scenario . . . 78
5.5 Summary . . . 79

6 Conclusions and future work **83**
6.1 Conclusions . . . 83
6.2 Future work . . . 85

References **91**

A Temporal Occupancy Maps **93**

List of Tables

2.1 Classification of SSM indicators 18
2.2 Relationship between the accident data and the collision location 21
2.3 Example of the ORFS calculation for location P_0 22

3.1 Point Grey FL3-GE-28S4-C specifications 26
3.2 Logitech C270 HD specifications 26
3.3 Asus ROG Strix G15 G512LI-70AT5PB1 laptop's specifications 28
3.4 Velodyne Puck VLP-16 specifications 30
3.5 Point Grey and USB camera proposed parameters 32

5.1 Charuco properties 67
5.2 Initial and calibrated positions of the three sensors frames with relation to the `base_link` frame (Figure 5.15) 70

Intentionally blank page.

List of Figures

2.1 Frame from the KITTI dataset 6
2.2 Algorithm that converts a dense point cloud D into a ground model G . . 6
2.3 LiDAR obstacle segmentation 8
2.4 LiDAR clustering parameters 9
2.5 Inference application architecture 10
2.6 YOLOPv2's car detection, road segmentation and lane marking 11
2.7 Key point detection and matching 11
2.8 ATOM architecture 13
2.9 LiDAR and RGB camera data combination. The green bounding boxes indicate objects identified by the object classifier, while the orange points represent LiDAR clusters. 14
2.10 Classification of risk assessment methods by approach 14
2.11 Probabilistic distributions for vehicles (top) and pedestrians (bottom) . . 15
2.12 Urban scene with predicted probabilistic distributions of pedestrians . . . 16
2.13 The HARA method of risk assessment 17
2.14 Architecture of the Deep Predictive model 19
2.15 SINADRA'S architecture 19
2.16 Collision risk algorithm based on V2V communication 20
2.17 Horizontal collision location classification 22
2.18 Example of P_1 collision between ego vehicle (red) and other vehicle (blue) 23

3.1 ATLASCAR2 25
3.2 Camera Point Grey FL3-GE-28S4-C 26
3.3 Logitech C270 HD 27
3.4 Asus ROG Strix G15 G512LI-70AT5PB1 28
3.5 Schematic example of the use of ROS 29
3.6 Example of an RViz interface used for real data collection with the ATLASCAR2 30
3.7 Example of a scenario in Gazebo simulator 31
3.8 Velodyne Puck VLP-16 31
3.9 Velodyne and cameras setup in the ATLASCAR2 32

4.1 Grid map in Rviz with ATLASCAR2 robot model 34
4.2 Car probabilistic distribution prediction. The brown bounding box indicates the current position, while the black arrow shows the orientation . . 35
4.3 Pedestrian probabilistic distribution prediction. The brown bounding box indicates the current position, while the black arrow shows the orientation 36
4.4 ATLASCAR2's trajectory 37

4.5 ATLASCAR2's future occupancy . . . 37
4.6 Stopped obstacles model (Bb stands for Bounding box) . . . 38
4.7 Stopped probabilistic distribution of a car, with the bounding box illustrated in blue . . . 38
4.8 ATLASCAR2's distribution intercepting another car's distribution for a 2 s time horizon . . . 39
4.9 Risk Maps ROS framework showcasing the connections between nodes (round shapes) and topics (rectangular shapes) . . . 40
4.10 Proposed LiDAR clustering parameters . . . 41
4.11 Example of Lidar Obstacle Detector object segmentation: Pedestrian on the left, car on the right. . . . 42
4.12 Changes in cameras setup and their respective field of view . . . 42
4.13 Setup for the panorama stitching method . . . 44
4.14 Panorama image . . . 44
4.15 Comparison of person's display between left camera image and panorama image when resized to 640×640 . . . 45
4.16 Division of the panorama image into two images represented by the red and green rectangle . . . 46
4.17 Comparison of person's display between panorama image and red rectangle panorama image when resized to 640×640 . . . 46
4.18 YOLOv7 obstacle detection and classification on the panorama image . . . 47
4.19 Corners derived bounding box (orange), cluster points derived bounding box (green), projected 3D bounding box corners (blue) and cluster points (red) . . . 49
4.20 LiDAR's and YOLOv7's bounding boxes combination through IoU . . . 49
4.21 Example of the Temporal Occupancy Map for the ATLASCAR2 with speed of $6\,\mathrm{m\,s^{-1}}$ . . . 55
4.22 Temporal Occupancy Map resulting from the union of the Occupancy Maps for 1 s, 2 s and 3 s for the ATLASCAR2 and another car . . . 56
4.23 Comparison between the Temporal Occupancy Maps and the Risk Maps in the first 2 frames . . . 57
4.24 Comparison between the Temporal Occupancy Maps and the Risk Maps in the last 2 frames . . . 58

5.1 Junction scenario . . . 60
5.2 Junction scenario risk evolution . . . 60
5.3 Representative frames of the junction scenario . . . 61
5.4 Opposite lane pass scenario . . . 62
5.5 Opposite lane pass scenario risk evolution . . . 62
5.6 Four representative frames of the opposite lane pass scenario . . . 63
5.7 Lane collision scenario . . . 63
5.8 Lane collision scenario risk evolution . . . 64
5.9 Representative frames of the lane collision scenario . . . 65
5.10 Crosswalk approach scenario . . . 65
5.11 Crosswalk approach scenario risk evolution . . . 65
5.12 Four representative frames of the crosswalk approach scenario . . . 66
5.13 ROS camera calibration interface showing the chessboard's corner detection 67

5.14 Four collections illustrative of the Intrinsic calibration process 68
5.15 Transformation tree of the ATLASCAR2's sensors 68
5.16 ATOM camera and LiDAR labeling result 69
5.17 LiDAR semi-automatic labeling corrected 69
5.18 ATOM calibration results 70
5.19 Panorama stitching setup to find the homography matrix 71
5.20 Real panorama image result 71
5.21 Comparison between the odometry encoder and the car speedometer velocity values for a duration of 10 s. 72
5.22 Real data collection path in Aveiro and scenario locations numbered from 1 to 5 73
5.23 Roundabout approach scenario risk evolution 74
5.24 Four representative frames of the roundabout approach scenario 75
5.25 Wide opposite lane pass scenario risk evolution 75
5.26 Three representative frames of the wide opposite lane pass scenario 76
5.27 Tight opposite lane pass scenario risk evolution 76
5.28 Three representative frames of the tight opposite lane pass scenario 77
5.29 Junction approach scenario risk evolution 77
5.30 Representative frames of the junction approach scenario 78
5.31 Crosswalk approach scenario risk evolution 79
5.32 Representative frames of the crosswalk approach scenario 80

A.1 Correspondent Temporal Occupancy Maps frames of the junction scenario 93
A.2 Correspondent Temporal Occupancy Maps frames of the opposite lane pass scenario 94
A.3 Correspondent Temporal Occupancy Maps frames of the lane collision scenario 95
A.4 Correspondent Temporal Occupancy Maps frames of the crosswalk approach scenario 96
A.5 Correspondent Temporal Occupancy Maps frames of the roundabout approach scenario 97
A.6 Correspondent Temporal Occupancy Maps of the wide opposite lane pass scenario 98
A.7 Correspondent Temporal Occupancy Maps frames of the tight opposite lane pass scenario 98
A.8 Correspondent Temporal Occupancy Maps frames of the junction approach scenario 99
A.9 Correspondent Temporal Occupancy Maps frames of the crosswalk approach scenario 100

Intentionally blank page.

List of Algorithms

1 Tracking IoU calculation 41
2 Panorama Stitching Algorithm 43
3 Cluster-to-BoundingBox Association 47
4 LiDAR Cluster Points Projection onto Panorama Image 48
5 IoU Calculation for Bounding Box Matching 50
6 ID Mapping 52
7 Assignment of Obstacle Properties to Object List 53
8 Time Horizon Calculation 54

Intentionally blank page.

List of Acronyms

ADAS Advanced Driver Assistance Systems

LiDAR Light Detection and Ranging

LAR Laboratory for Automation and Robotics

GPS Global Positioning System

IMU Inertial Measurement Unit

RANSAC RANdom SAmple Consensus

LLR Log-Likelihood Ratio

ROS Robot Operating System

PCA Principal Component Analysis

IoU Intersection over Union

ATOM Atomic Transformations Optimization Method

AV Autonomous Vehicles

HARA Hazard Analysis and Risk Assessment

TTC Time to Collision

ODD Operational Design Domain

V2V Vehicle to Vehicle

MPC Model Predictive Control

PCSI Potential Crash Severity Index

ORFS Odds Ratio for Fatal and Severe

CAN Controller Area Network

Intentionally blank page.

Chapter 1

Introduction

The growth in autonomous driving and Advanced Driver Assistance Systems (ADAS) has revolutionized the industry, promising enhanced safety. The core section within these technologies is the perception which brings the ability to perceive the driving environment accurately. Among the sensors utilized, cameras and Light Detection and Ranging (LiDAR) are the most popular, each offering unique advantages and complementing each other in creating a comprehensive understanding of the surroundings.

Vision systems, such as cameras, are the sensors most similar to human vision, which provide high-resolution images with color and texture information, enabling qualitative classification of the road obstacles. On the other hand, LiDAR systems use laser pulses to measure distances to objects, which, depending on the sensor capacity, can create 3D maps of the environment. These systems complement the camera's qualitative information very well by providing quantitative information about the obstacles, namely their position and size in space.

The integration and combination of vision and LiDAR data enables the creation of road maps, which can be very useful for autonomous driving and ADAS. These maps represent the driving environment, including road obstacles and other participants, such as pedestrians and drivers. Creating reliable maps involve several complex tasks, such as object detection and classification, LiDAR segmentation and clustering, camera and LiDAR data combination.

This dissertation contributes to the ATLAS project [1] by enhancing perception capabilities. It involves implementing a 3D LiDAR in one of the ATLAS project prototypes, the ATLASCAR2, to improve its environmental perception. This enhancement allows for the creation of detailed maps that represent the vehicle's surroundings. Therefore, the work presented in this dissertation can advance autonomous driving and ADAS within the ATLAS project [1].

1.1 ATLAS project

The ATLAS project [1] started in the Laboratory for Automation and Robotics (LAR) at the department of Mechanical Engineering of the University of Aveiro. The objective of this project is to develop and enable the proliferation of advanced sensing and active systems, designed for implementation in automobiles and similar platforms. The project started with autonomous navigation in controlled scenarios, but has been

evolving to deal with real road scenarios in the past years. In this context, the prototype vehicle used for current experiments is the ATLASCAR2, which is described in Section 3.1.1.

1.2 Problem description

Every year, around 1.3 million lives are tragically lost due to road traffic accidents, and an additional 20 to 50 million people endure non-fatal injuries, often resulting in long-term disabilities. The impact of these incidents extends beyond individual suffering, causing substantial economic burdens on affected individuals, their families, and entire nations [2]. Autonomous cars emerge as a potential solution to this problem. By leveraging advanced sensor technologies, artificial intelligence, and real-time data analysis, autonomous vehicles aim to mitigate the risks associated with human errors in driving.

Nowadays, there are several companies investing in autonomous cars such as Tesla, General Motors and Waymo [3], that have good solutions but still far from the ideal. In October 2023, General Motors had to remove all of their 950 autonomous cars from the streets of California because one of their Cruise vehicles critically damaged someone. This person was hit by another car and landed on the path of the Cruise vehicle which made it stop at first, but then in order to get out of the traffic it pulled over and dragged the person about six meters forward [4]. This emphasizes the urgent need to improve current autonomous vehicles or ADAS abilities of handling hazard situations for the sake of road safety.

A key aspect to enhance these systems is to effectively assess and mitigate the collision risk. Collision risk refers to the probability of the ego vehicle being involved in a collision with another road agent namely, vehicles and pedestrians. In order to support drivers, whether human or machine, in detecting potential collisions, it is essential to formalize and represent this risk. One way to achieve this, is by creating a sort of map that displays the collision risk, along with some indicators or descriptors to grade it. This is how the Risk Maps concept was developed.

The use of Risk Maps empowers both drivers and autonomous vehicles to make informed, preemptive decisions. This includes adjusting speeds, changing lanes, or choosing alternative routes to avert potential collisions with obstacles. The main goal is to enhance situational awareness by providing real-time information about the surrounding environment.

1.3 Objectives

The work developed within the scope of this dissertation has two primary objectives. The first is to improve the perception power of the ATLASCAR2 by installing a 3D LiDAR in its infrastructure. The second is to develop a concept to enhance ATLASCAR2's ability to perceive hazard events, by creating the Risk Maps. To accomplish these objectives, the work is divided into the following sub goals:

- Understand the best location in the car to install the LiDAR alongside the existing cameras;
- Find a solution to detect and classify road obstacles;

- Find a solution to assess and display the collision risk;

1.4 Document structure

This dissertation comprises six chapters:

- Chapter 1: Presents the dissertation by providing an introduction of the work, an overview on the background problem and the thesis objectives.
- Chapter 2: Presents the state of the art on the main concepts of the dissertation and solutions to fulfill the objectives.
- Chapter 3: Describes the hardware and software utilized for the dissertation, and also the concretization of the first primary objective.
- Chapter 4: Describes the development of the proposed solution, the Risk Maps. It provides an overview of the challenges faced and the decisions that were made.
- Chapter 5: Presents the tests done on the developed solution in both simulated and real world scenarios. Additionally, it provides an overview on extra software configurations needed for the real world data collection.
- Chapter 6: Presents the conclusions of this dissertation and suggestions for future works.

Intentionally blank page.

Chapter 2

State of the Art

This chapter focuses on recent developments and methodologies on the various areas that this thesis relies on. It addresses topics like LiDAR and camera perception, sensor calibration, collision risk and damage, which are essential for autonomous systems. Thus, it represents the exploration phase of the work that provided an understanding on the various subjects and the discovery of essential tools.

2.1 ATLASCAR2

Fellow students and professors at the LAR at the University of Aveiro have developed projects over the years using the ATLASCAR2 for various case studies. This work is stored in a GitHub repository [5], which currently offers several packages and features useful for working with the ATLASCAR2 in both real world and simulation environments. Additionally, it has a detailed guide for setting up the car and a list of the available hardware, namely:

- 2 RGB cameras
- 2 2D LiDAR
- Global Positioning System (GPS) + Inertial Measurement Unit (IMU) system
- Odometry unit

A more detailed overview of the experimental infrastructure will be addressed in in Chapter 3.

2.2 Obstacle detection with 3D LiDAR

Since one of the goals is to install and use a 3D LiDAR in the ATLASCAR2, there's a need to understand how to process its data in order to detect potential obstacles on the road. In [6] the authors used a Velodyne LiDAR [7] combined with a GPS/IMU localization system to create a framework for ground surface estimation and static/moving obstacle detection shown in Figure 2.1.

The proposed approach is divided in two major steps: (1) piece-wise surface fitting algorithm to estimate a finite set of multiple surfaces that fit the road and surrounding area; (2) 3D voxel-based representation for obstacle modeling. For the first part, the point-clouds given by the sensor are multiplied by transformation matrices to go from

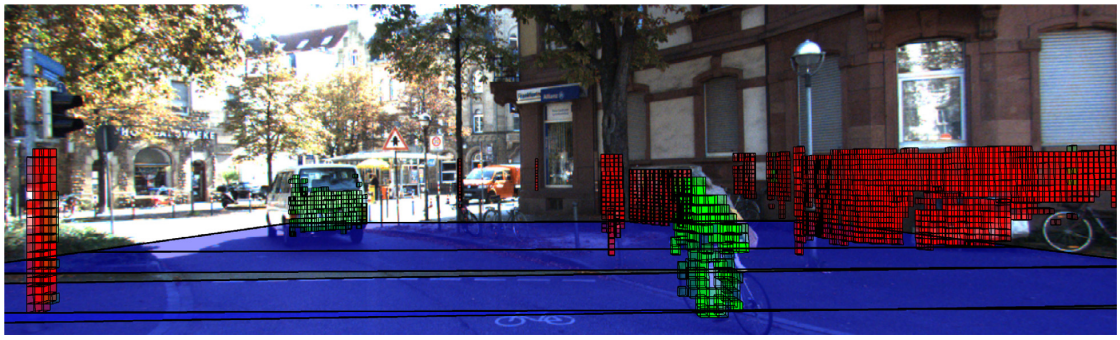


Figure 2.1: Frame from the KITTI dataset showing the estimated ground surface in blue, dynamic obstacles in green and static ones in red [6]

the ego-vehicle's to the world coordinate system. Then, a 'Box Grid Filter' is applied which breaks the space into voxels speeding up the point-cloud's registration. The post-processed point clouds (D) are transformed back into the car's coordinate system and go through the algorithm in Figure 2.2 which is composed by four tasks: (1) Slicing, (2) Gating, (3) Plane fitting with RANdom SAmple Consensus (RANSAC) and (4) Validation.

Algorithm Piecewise Ground Surface Estimation.

Input: Dense Point Cloud: D
Output: Ground Model: $G = \{G_1, \ldots, G_N\}$
for slice $k = 1$ to N **do**
  $S_k \leftarrow$ Slice (D)
  $\dot{S}_k \leftarrow$ Gate (S_k)
  $G_k \leftarrow$ RANSAC $(\dot{S}_k)$
end for
for slice $k = 1$ to N **do**
  if $\neg((\delta\psi_k < \tau^\circ) \wedge (\delta Z_k < \ell))$ **then**
    $G_k \leftarrow G_{k-1}$
  end if
end for

Figure 2.2: Algorithm that converts a dense point cloud D into a ground model G [6].

The resulting Ground plane points are then removed leaving only those that represent the obstacles (O). After that, a process called voxelization is applied to the dense cloud (D) and the obstacle points (O), turning the points into voxels $\bar{D}$ and $\bar{O}$ with a size of 0.1 m. This is followed by a segmentation to differentiate the moving from the stationary obstacles which starts with a simple subtraction mechanism as the dynamic objects will occupy different voxels while the stationary stay at the same ones in consecutive scans. Later, these results are refined using an analysis based on $2DCounters$, expressions (2.1) and (2.2), and the Log-Likelihood Ratio (LLR), equation (2.3), is used to obtain

the binary masks of the moving/stationary voxels,

$$H_s(\check{x},\check{y}) = \sum_{k=1}^{n(\check{x},\check{y})} \bar{D}(\check{x},\check{y},\check{z}) \tag{2.1}$$

$$H_d(\check{x},\check{y}) = \sum_{k=1}^{m(\check{x},\check{y})} \bar{O}(\check{x},\check{y},\check{z}) \tag{2.2}$$

where $(\check{x},\check{y},\check{z})$ is the location of a cell in a voxel grid. H_s and H_d are the computed static and dynamic counters, $n(\check{x},\check{y})$ and $m(\check{x},\check{y})$ indicate the number of voxels in the column/bar of $(\check{x},\check{y})$ in $\bar{D}$ and $\bar{O}$, respectively,

$$R(\check{x},\check{y}) = \log \frac{\max\{H_d(\check{x},\check{y}),\varepsilon\}}{\max\{H_s(\check{x},\check{y}),\varepsilon\}} \tag{2.3}$$

where ε is set to 1 to prevent division by zero or taking the log of zero. By applying a threshold on $R(\check{x};\check{y})$, 2D binary masks of the stationary and moving voxels can be obtained using the following expressions:

$$B_d(\check{x},\check{y}) = \begin{cases} 1 & \text{if } R(\check{x},\check{y}) > T_d \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

$$B_S(\check{x},\check{y}) = \begin{cases} 1 & \text{if } R(\check{x},\check{y}) < T_S \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where T_s and T_d are the thresholds used to compute the 2D decision masks for detecting the most reliable stationary and moving voxels. The static (B_s) and dynamic (B_d) binary 2D masks are applied to all levels of $\bar{D}$ and $\bar{O}$ voxel grids to generate voxels labeled as stationary (V_S) or moving (V_M).

A similar open source software, developed within Robot Operating System (ROS) by the author in [8], also provides tools to process 3D LiDAR data. The application named Lidar Obstacle Detector, works by taking the raw point cloud ROS topic and the segmentation thresholds defined by the user as the input. Then, it outputs the ground and obstacle point cloud, and the 3D bounding boxes of the clustered objects which contain the information of their centroid and dimensions. The object clusters are determined using an Euclidean clustering algorithm which calculates the Euclidean distance between neighboring points. If the distance falls within a threshold, the points are considered part of the same cluster. Subsequently, the 3D bounding boxes are designed using the minimum and maximum points (x, y, z) within each cluster, with the centroid being the mean value between these points. Figure 2.3 shows the resulting bounding boxes.

The thresholds and features of the application can be defined by the user in real time with $rqt\_reconfigure$ [9] which is a graphical interface in ROS to view and edit node parameters. This interface is illustrated in Figure 2.4:

A brief explanation of each parameter is presented next:

use_pca_box is an option to build bounding boxes using Principal Component Analysis (PCA) algorithm to estimate the orientation. Every point's z coordinate of the cluster is changed to be the same as the centroid's z coordinate forming an XY

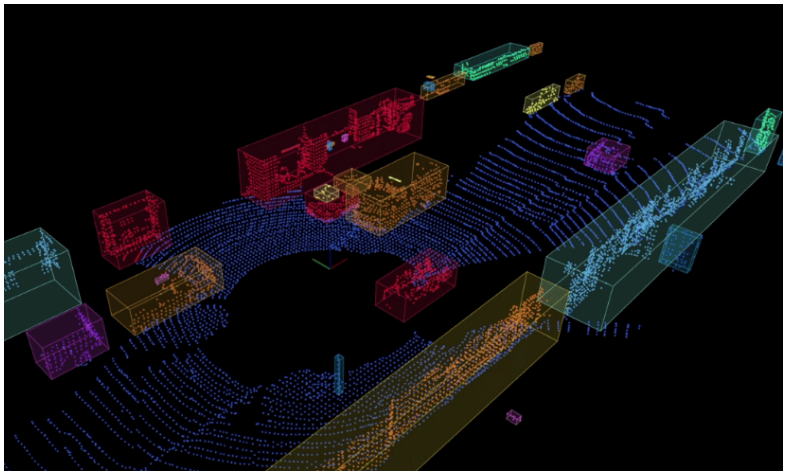


Figure 2.3: LiDAR obstacle segmentation [8]

plane. Then, the PCA algorithm computes the principal directions of the cluster. Finally the points are projected back to their original z position but rotated along the PCA frame.

use_tracking is an option to enable bounding box tracking. The tracking consists of comparing bounding boxes in consecutive frames. The metrics being compared is the euclidean distance between the centroids and the Intersection over Union (IoU) which have their respective thresholds set in the last two parameters: **displacement_threshold** and **iou_threshold**.

voxel_grid_size is the size of the voxels used in the Voxel Grid Downsampling algorithm. This algorithm reduces the number of points within a point cloud by dividing it into equally sized cubic voxels. A smaller voxel grid size results in denser sampling of the point cloud, preserving more details but requiring more computational resources.

roi is the region of interest in the LiDAR's frame where bounding boxes are created.

ground_threshold represents the maximum euclidean distance from a point to the estimated ground plane for it to be considered an inlier. Points within this threshold distance are classified as belonging to the ground plane, while those beyond it are considered outliers.

cluster_threshold , as said above, represents the maximum euclidean distance allowed between neighboring points for them to be considered part of the same cluster.

cluster_size defines the minimum and the maximum number of points required for a group of points to be considered a cluster.

2.3 Obstacle detection and classification with cameras

LiDAR provides accurate distance measurements of the obstacles, but struggles or fails to identify the nature or properties of those. The integration of cameras provides detailed visual information (e.g.: color, shape, texture, ...), which is valuable for obstacle recognition. The next paragraph addresses a method of processing ATLASCAR2's camera data.

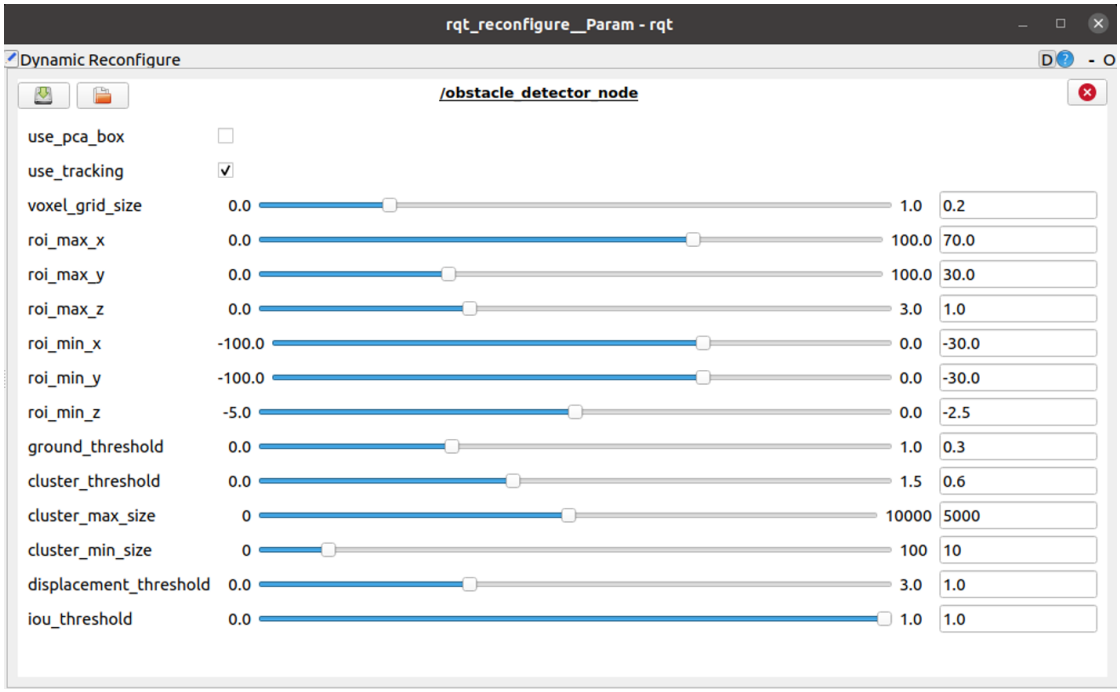


Figure 2.4: LiDAR clustering parameters

In a recent LAR's project [10, 11], Ribeiro developed a modular application with the purpose of inferring real data provided by the ATLASCAR2 through multiple deep-learning models. It was built within the ROS framework to facilitate the communication since the car's software framework is also ROS.

The application architecture is illustrated in Figure 2.5, where the yellow block represents the deep-learning model.

The architecture is designed with four main blocks, namely the Receiver, the Data sender, the Inference node and the Input arguments:

- The Data sender block is assigned to read image or video data and send it to the Inference node block.
- The Input arguments block send the Model file path, the Module and the Model loader names to the Inference node block.
- The Inference node block receives the information from the blocks mentioned above and outputs the inference results to the Receiver block. This block is divided in two parts, the Inference solution and the Inference manager.
  - The Inference solution receives the Model information from the Module and Model loader and outputs the inference result. The first one is responsible for handling the Model's input requirements in terms of Data format, and adapt the Model's output to fit in the Inference manager. The Model loader loads the Model an returns a variable with the Model, the Model's framework (eg.: PyTorch) and precision (eg.: float32).

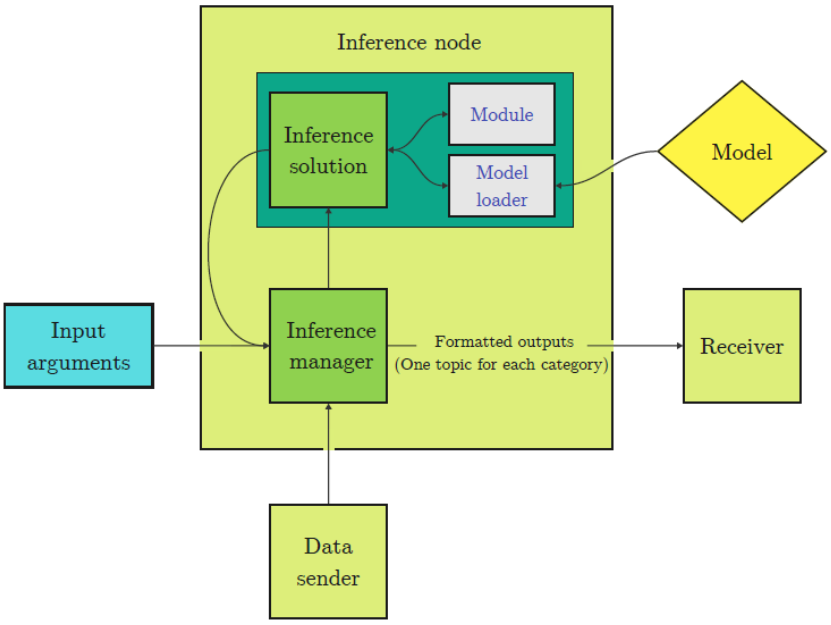


Figure 2.5: Inference application architecture [10]

  - The Inference manager receives the data from the Data sender as well as the inference and the Model's parameters, and outputs the inference solution in a standardized format to the Data receiver block.

- The Receiver block acquire the inference results and display them.

As Ribeiro's main objective was to compare multi-tasked with multiple single-task networks, he designed another block to evaluate the performance of various deep learning model's in three categories: car detection, road segmentation and lane marking. The results led to the conclusion that multi-task models are better than using multiple single-task models. This is attributed to the slower inference speed and difficulty in synchronization associated with the latter. The best single-task and multi-task model for car detection is YOLOv8 [12] and YOLOPv2 [13], respectively. Figure 2.6 shows the results of the YOLOPv2 in day-time (top images) and night-time (bottom images) scene.

On the other hand, panorama image stitching, while not directly related to object detection or classification, can be very useful as it enhances the overall perception of the scene. The author in [14] developed an application to merge two video streams into a single panoramic stream. The process is based on OpenCV's stitching module, which offers several built-in functions. Since it involves a video stream and the cameras are stationary, the homography is calculated only once from the initial frames. The process of obtaining the homography involves identifying key points or features in both frames and then matching those that are similar. An example of this correspondence is illustrated in Figure 2.7. Furthermore, the homography is applied to the right camera frames allowing them to transition to a different plane in order to promote a seamless integration with the left camera frames.

Figure 2.6: YOLOPv2's car detection, road segmentation and lane marking [13]

Figure 2.7: Key point detection and matching [14]

2.4 Calibration

Both LiDAR and camera provide complementary data that enables a more robust, accurate and versatile perception system which is necessary for building the risk maps. In this context, calibration of these sensors is essential, aiming to improve the precision of the geometric relationship between them.

A powerful calibration tool named Atomic Transformations Optimization Method (ATOM) [15] was developed in LAR. This framework supports the calibration of complex robotic systems with multiple sensors and of different modalities. For example, it allows the simultaneous calibration of LiDAR, RGB, and RGB-D cameras in the same run. Additionally, it works for different calibration scenarios such as sensor to sensor, sensor in motion and sensor to coordinate frame. The software architecture is represented in Figure 2.8:

To calibrate sensors using ATOM, a ROS bag file containing recorded data from the sensors, as well as transformation and joint messages, is essential. Additionally, the xacro/URDF file of the robotic system, which contains its configuration, is required. Moreover, the calibration configuration file which contains information about the sensors and the calibration pattern must be defined. ATOM offers a functionality that allows the user to set an estimation of the sensors initial pose through interactive markers in RViz, but this step can be skipped if the sensor's pose described in the *urdf* or *xacro* file is accurate enough. The next step is data collection which is where the user saves various collections with different pattern or sensor poses from the bag file. After that, there's a labeling process that involves annotating the detected portions of the calibration pattern. The labeling can be manual, semi-automatic and fully automatic depending on the sensor modality. Afterwards, ATOM allows the user to play the dataset with the corrected collections to ensure that the labeling process was successful. Finally, the calibration step which, in a summarized way, adjusts the sensors with relation to the pattern in order to minimize the reprojection error. Subsequently, the optimized robot description file is generated, ready to use.

2.5 Combination of LiDAR with Camera

There are two main approaches to combine LiDAR data with RGB camera data. It is either to project the LiDAR 3D points into the camera image, or to project the image pixels into the 3D point cloud. The biggest advantage of this approach is the ability to classify objects detected by the LiDAR in the 3D space. Depending on the LiDAR's resolution, an estimation of the object's nature can be made. This estimation can then be refined or supplemented by classifiers operating on camera images. In [17] the authors use the technique of projecting LiDAR's points into the camera image for object classification. The process starts by calibrating the camera intrinsic parameters and then both camera's and LiDAR's extrinsic parameters. Subsequently, the LiDAR's points are mapped into the image using these parameters. Figure 2.9 shows the result of this union:

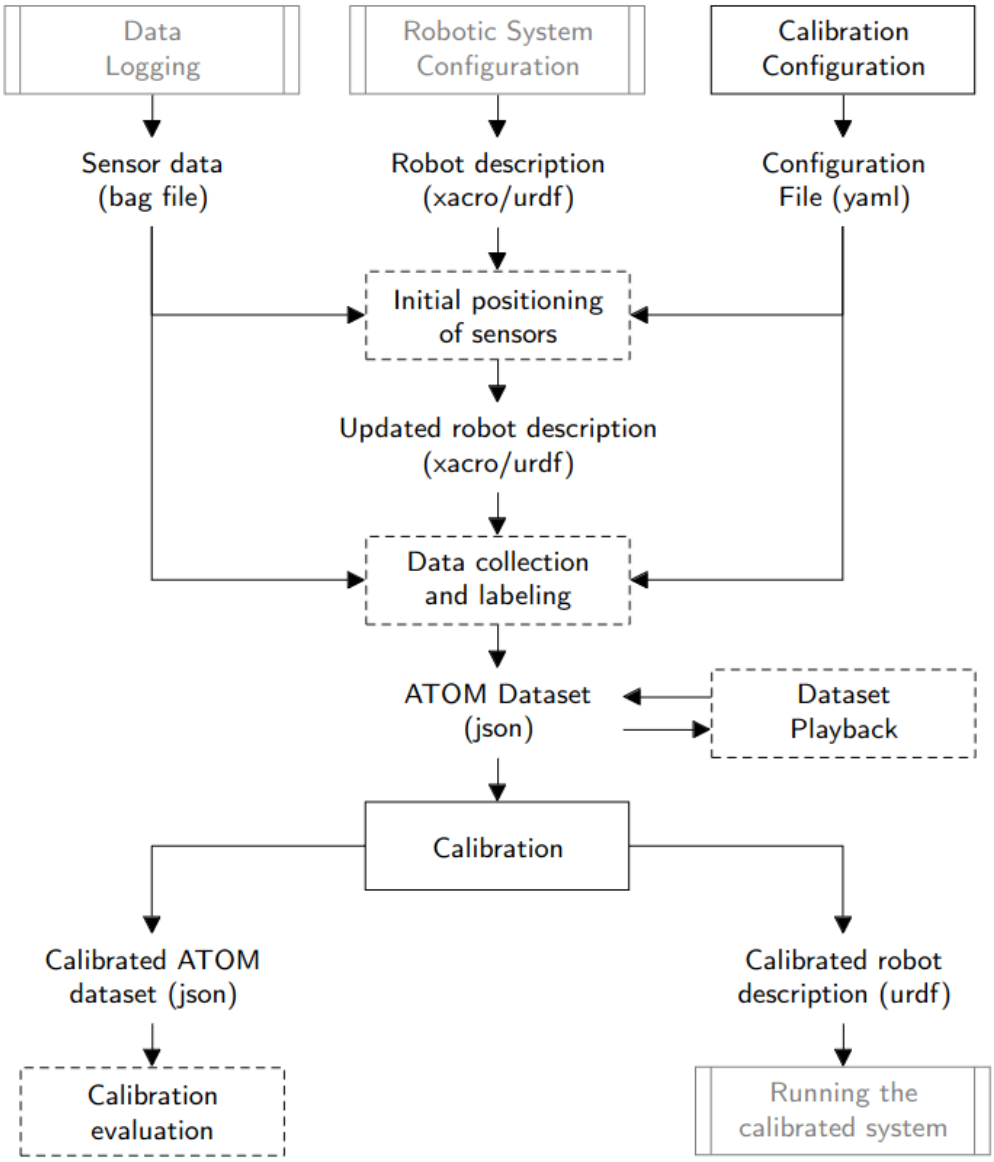


Figure 2.8: ATOM architecture [16]

2.6 Collision risk

Collision risk is paramount in the autonomous driving world as it allows Autonomous Vehicles (AV) or drivers to avoid potentially dangerous situations by giving information of a future event based on the actual state of every participant in the surrounding scene. As the environment complexity increases, the more difficult it is to assess an effective risk metric due to the increased number of interactions involved [18] and, consequently a more sophisticated methodology is needed.

Figure 2.10 presents the most used and researched methods to determine the collision risk [19], [18]s.

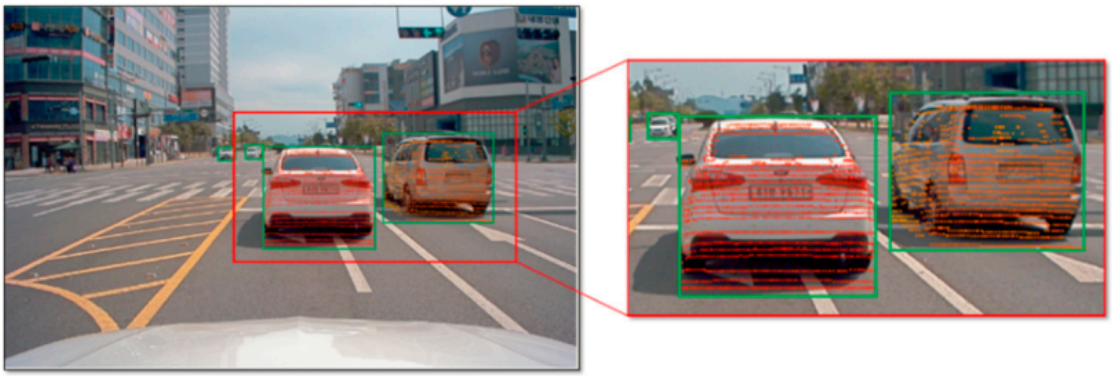


Figure 2.9: LiDAR and RGB camera data combination. The green bounding boxes indicate objects identified by the object classifier, while the orange points represent LiDAR clusters. [17]

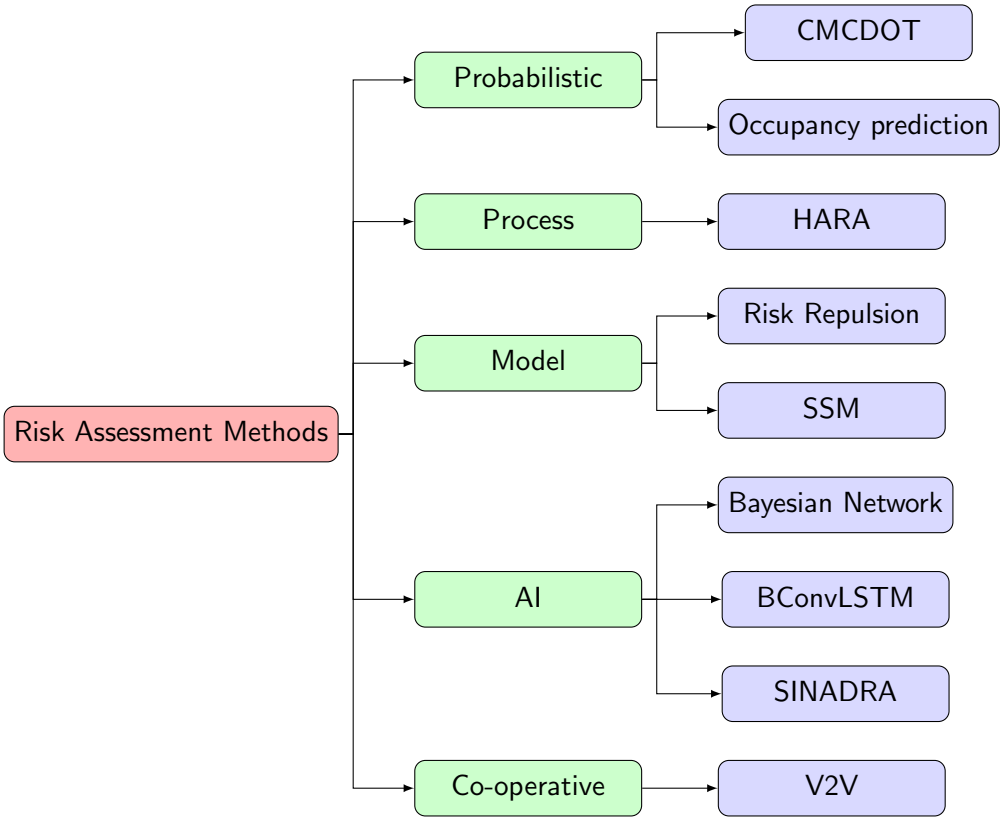


Figure 2.10: Classification of risk assessment methods by approach, according to [18, 19]

2.6.1 CMCDOT (Conditional Monte Carlo Dense Occupancy Tracker)

This method outputs the probability of collision risk between the vehicle and a identified object in the scene [20]. It is structured with four main steps. In the first step (state representation) the algorithm models the state distributions of cells in a grid where each is potentially occupied by a static object (s), dynamic object (d), empty (e) or undefined (u). The second step (Prediction) consists of state prediction for each of

the mentioned above (s, d, e, u). After the model is projected in time the particle vectors are transformed (rotated and translated) in order to be expressed in the new frames. The third step (Evaluation) evaluates the updated distributions of the states in each cell of the grid. The fourth step (Particle re-sampling) is the dynamic part sampling which generates new particles for the newly dynamic parts to reassign the right amount of particles per cell. The collision risk occurs when grids are overlapped in the future prediction of the vehicle movement and the object.

2.6.2 Occupancy Maps

In [21] the authors proposed an open source method to evaluate the collision risk based on motion prediction of vehicles and pedestrians through the stochastic models: radial and angular distribution. These models account for uncertainties in sensing as the final result is a probabilistic distribution of the future heading. The distributions are different for vehicles and pedestrians as they have different motion models. These distributions are represented in Figure 2.11 where the probability goes from 0 (blue) to 1 (red):

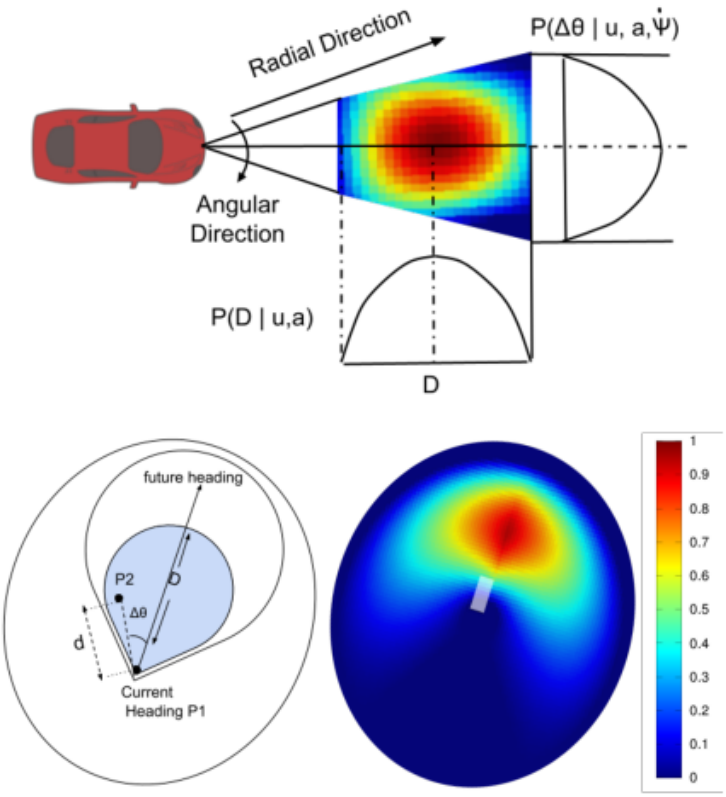


Figure 2.11: Probabilistic distributions for vehicles (top) and pedestrians (bottom) [21]

In order to compute the probabilistic distributions, information about the obstacles and the ego vehicle is needed, namely:

- Position (x, y) relative to the ego's vehicle coordinate frame
- Speed (v_x, v_y)

- Acceleration (a_x, a_y)
- Orientation angle (yaw)
- Dimensions (x, y)
- Obstacle's nature (e.g. pedestrian, car, ...)
- Ego vehicle speed (v_x, v_y)
- Time horizon (time in the future for which the prediction is made)

The ego vehicle trajectory is computed using a simple motion equation where the distance that it will cover is obtained from the product of its speed by the time horizon. This trajectory is represented in Figure 2.12 as a red rectangular shaped area.

The collision risk is obtained through the intersection of the ego's trajectory with the obstacle's distributions. The maximum value of probability intersected is the collision risk.

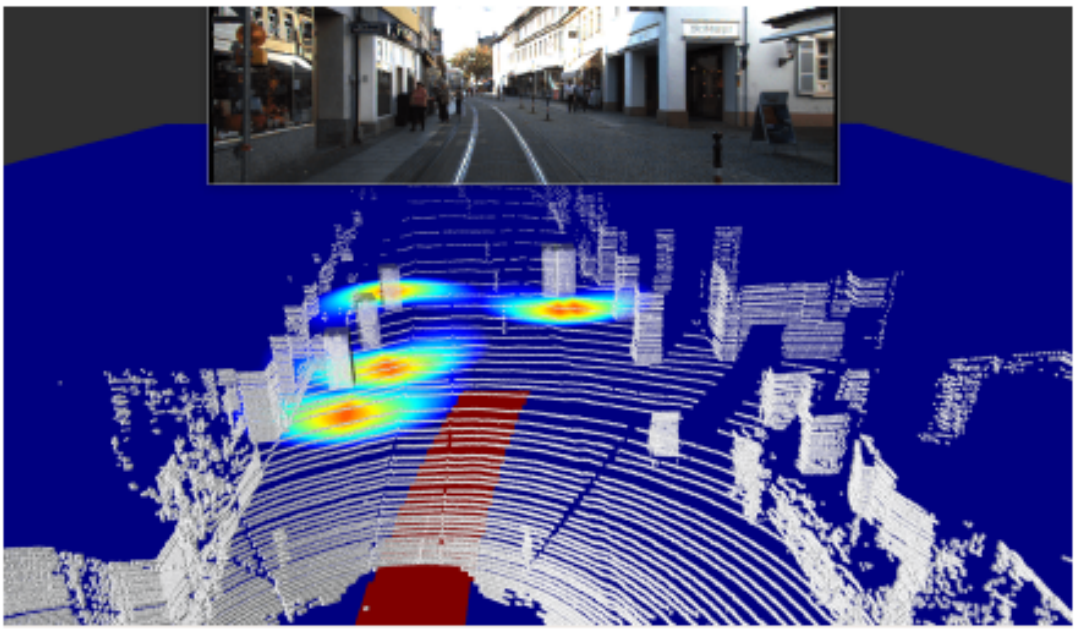


Figure 2.12: Urban scene with predicted probabilistic distributions of pedestrians [21]

2.6.3 HARA

The Hazard Analysis and Risk Assessment (HARA) methodology is normally used during the development of the vehicle where possible hazards can be identified from malfunction behaviours. Following that, the risk is calculated to evaluate the likelihood of each hazard resulting in different scenarios. From here, adjustments are made to ensure the safety goals [22]. This iterative approach is represented in Figure 2.13:

While this dissertation does not directly focus on HARA, as it does not rely on sensor data for collision risk assessment, its significance in the world of collision risk

Figure 2.13: The HARA method of risk assessment [22]

management cannot be overlooked. By proactively addressing potential hazards and their associated risks, HARA plays a crucial role in preventing and mitigating collisions.

2.6.4 Risk Repulsion

The model of Risk Repulsion is based on the field theory of traffic flow which describes traffic interactions with physical expressions [23]. In the longitudinal car-following scenario when the driver approaches the front car at high speed, a force is perceived in a way that makes the driver decelerate to keep a safe distance. The authors in [24] called this force as risk repulsion which is inversely proportional with Time to Collision (TTC) and also takes in consideration the time headway between the two cars, relative speed and safe following distance.

2.6.5 SSM

Surrogate Safety Measures is a framework of metrics that are used to measure the nearness to a collision of traffic events [25]. These are divided into four categories, namely time-based, distance-based, deceleration-based and energy-based, as shown in Table 2.1:

2.6.6 Bayesian Network

A Bayesian network is a probabilistic model that represents the causal relationship among various variables by using conditional probabilities. In [26] the authors used a Bayesian hierarchical model for real-time prediction of the collision risk at highway entrances and exits. The model has tree layers: the first layer is the vehicle physical state, that includes its kinematics and dynamic characteristics, such as position, speed and acceleration; the second layer is related to the vehicle interaction, which takes into account motion variances like changes in distance, speed and steering angle between the interacting vehicles; the third layer is used to represent the risk probability of the collision occurrence. To conclude, this model allows for real time evaluation of the collision risk of multi-vehicle interaction.

Table 2.1: Classification of SSM indicators

Category	Indicator	Definition
Time-based	TTC (Time to Collision)	Time for two cars to crash at their current speed and direction
	PET (Post-Encroachment Time)	Time interval between one party leaving and the other arriving at the conflict zone
	TET (Time Exposed Time-to-Collision)	Total time period when TTC is below the threshold
Distance-based	SDI (Stop Distance Index)	Minimum distance required to avoid collision with a front vehicle when it decelerates at the maximum deceleration rate until stops
	PICUD (Potential Index for Collision with Urgent Deceleration)	Distance between two vehicles after they complete emergency braking
	DSS (Difference of Space distance and Stopping distance)	Same as PICUD plus the friction
Deceleration-based	DRAC (Deceleration Rate to Avoid the Crash	Speed difference between two vehicles divided by TTC
	CPI (Crash Potential Index)	Extended version of DRAC that considers vehicle's maximum deceleration rate
Energy-based	Delta V	Velocity changes in vehicle trajectory before and after the collision
	CFI (Conflit Index)	Kinetic energy released during collisions is estimated by combining PET with speed, mass and relative angle

2.6.7 BConvLSTM (Bayesian Convolutional Long Short Term Memory)

This artificial intelligence approach for the collision risk assessment is based on deep learning techniques [27]. The model employed, known as the Deep Predictive model, utilizes sequences of image sensor data, vehicle state information, and driver action commands as its input variables. This information is then processed in both space and time to predict future collisions. Unlike other similar methods, this one estimates uncertainty in the predictions generated which are then used to improve the quality and accuracy of the method's output. The architecture is represented in Figure 2.14.

2.6.8 Situation-Aware Dynamic Risk Assessment (SINADRA)

SINADRA [28] is an open source solution that evaluates the collision risk in 4 steps: i) identification of the ego vehicle and its class; ii) behaviour prediction of every party in the scene; iii) transformation of the behaviours into trajectories; and, iv) calculate the risk. The first three steps are based on data sources such as research data sets, sim-

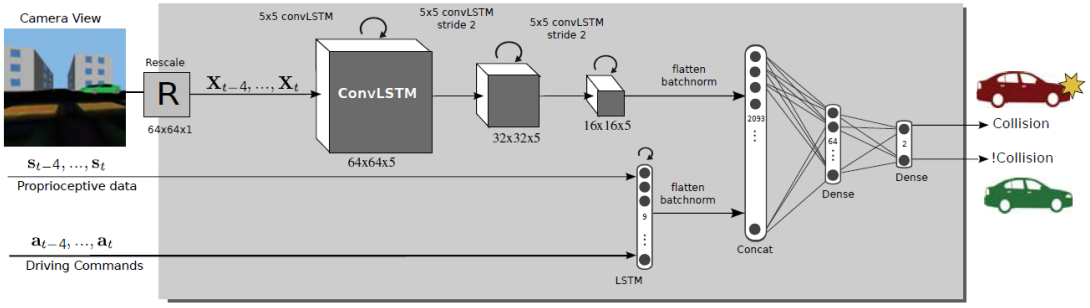


Figure 2.14: Architecture of the Deep Predictive model [27]

ulation, accident and driving behaviour research. Figure 2.15 illustrates the method's architecture. In the first step, the ego vehicle is placed in a class that the authors

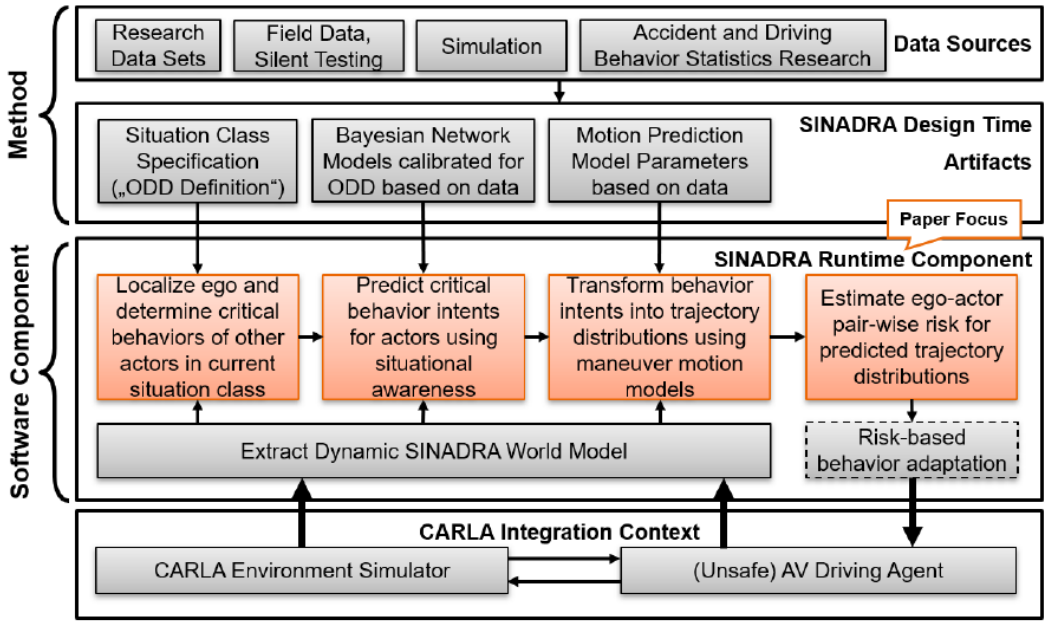


Figure 2.15: SINADRA'S architecture [28]

call Operational Design Domain (ODD). Each ODD has predefined a set of ego behaviours, other road users behaviours and environmental conditions. The second step is an application of the Bayesian network method (explained in 2.6.6) to predict behaviour intents of all the participants based on the actual ODD's behaviours and knowledge about decision-making in traffic. The third step is a generation of a trajectory based on each predicted behaviour through behaviour-specific motion models. With the predicted trajectories of both ego and other road users, the risk can then be computed with any risk assessment metric such as TTC (Table 2.1).

2.6.9 Risk assessment from V2V framework

The Vehicle to Vehicle (V2V) framework relies on wireless communication between vehicles allowing information transfer such as vehicle position, direction, speed and acceleration [29]. This information is then fused with sensor's and radar's scans providing location and motion estimation of the other vehicles. The outcome goes through Model Predictive Control (MPC) that makes predictions for both latitudinal and longitudinal aspects of the targets. Having targets movement predictions, it is possible to estimate the collision risk. The model's architecture is illustrated in Figure 2.16. An interesting feature about V2V is that a global collision risk can be assessed if all the vehicles share their perceived risk values. It is called the augmented collision risk and represents the entire scene.

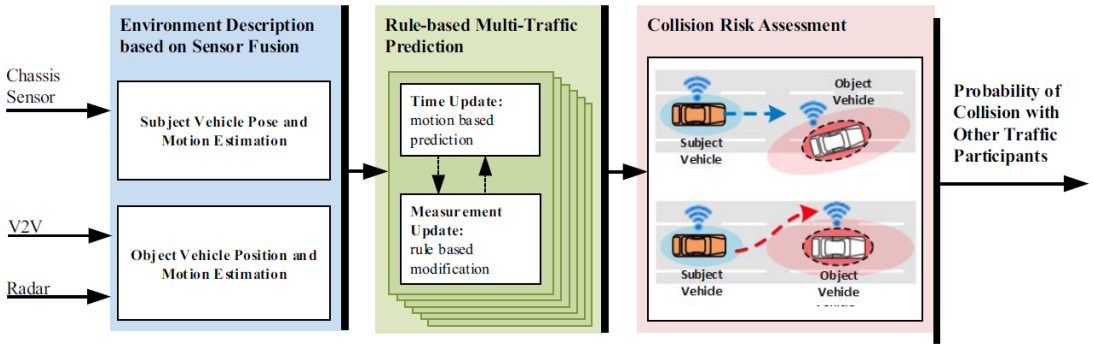


Figure 2.16: Collision risk algorithm based on V2V communication [29]

This method is particularly valuable in occluded scenarios, where the ego vehicle lacks information about potential obstacles that may be obscured from its sensors. In such situations, another vehicle from a different perspective may have access to crucial information and can communicate it to the ego vehicle, thereby preventing certain collision scenarios.

2.7 Collision damage

Similarly to collision risk assessment, potential collision damage or severity estimation is an important metric to ensure people's safety on the road. This is relevant, especially in scenarios where a collision is unavoidable, having the option to choose a trajectory that minimizes damage or reduces the risk of injury, becomes particularly valuable.

The authors in [30] address this exact situation. They developed a MPC-based motion planner that gives the trajectory with the lowest crash severity when collision is inevitable. The part of this method that has the most interest for this dissertation is the calculation of the crash severity, which is based on three factors: relative speed, relative heading angle and mass ratio between the collision vehicles.

1. Relative Speed ΔV: velocity metric to measure Potential Crash Severity Index (PCSI):

$$\text{PCSI}(\Delta V) = \frac{\Delta V}{D} \tag{2.6}$$

where, ΔV and D are the approaching velocity and the distance between the ego vehicle and the obstacle, respectively.

2. Relative Heading Angle θ: Heading angle towards each vehicle that goes from 0° to 180° in six steps: 0° – 15° 15° – 45°, 45° – 90°, 90° – 135°, 135° – 165°, and 165° – 180°. The PCSI index related to the relative angle θ is given by (2.7).

$$\text{PCSI}(\theta) = \begin{cases} 1 & \theta = 0°\&180° \\ 4 & \theta = 0° \sim 15°\&165° \sim 180° \\ 3 & \theta = 15° \sim 45°\&135° \sim 165° \\ 2 & \theta = 45° \sim 90°\&90° \sim 135° \end{cases} \tag{2.7}$$

3. Mass Ratio W_o/W : Mass ratio between the weights of the obstacle vehicle W_o and the ego vehicle W:

$$\text{PCSI}(W) = \frac{W_o}{W} \tag{2.8}$$

The total $PCSI$ is defined as:

$$\text{PCSI} = k_{\Delta v}\,\text{PCSI}(\Delta V) + k_\theta PCSI(\theta) + k_w\,\text{PCSI}(W)W \tag{2.9}$$

where $k_{\Delta v}, k_\theta$, and k_w are the weights of $PCSI$ related to the relative speed, relative heading angle, and mass ratio, respectively.

In [31] the authors also proposed a motion planning approach that outputs the trajectory associated with the minimum collision severity for the passengers of the ego vehicle. They used real accident data [32] to establish a relationship between the impact location and the severity of injuries. The considered data only includes accidents at junctions. In this scenario at least one passenger car was involved in the accident, and the collisions involved only passenger cars against other vehicles.

The impact location classification is illustrated in Figure 2.17. The relationship between the accident data and the collision location is defined in Table 2.2 for three injury metrics: fatal, severe and minor.

Table 2.2: Relationship between the accident data and the collision location

Collision Location	Description	Fatal	Severe	Minor	ORFS
B_0	Rear compartment	2	1	10	0.61
D_0	Distributed across entire side	4	4	13	1.30
F_0	Front compartment	7	25	72	0.91
L_0	1/4 from left side	0	0	6	0.0
L_1	1/3 from left side	0	0	1	0.0
P_0	All of passenger compartment	24	11	52	1.54
P_1	Passenger compartment-front seat	1	3	17	0.48
P_2	Passenger compartment-rear seat	1	0	10	0.20
R_0	1/4 from right side	0	0	6	0.0
R_1	1/3 from right side	0	0	1	0.0
Y_0	Front and passenger compartment	10	15	33	1.71
Y_1	Front compartment and front seat	7	6	32	0.83
Z_0	Rear and passenger compartment	9	6	31	1.01
Z_1	Rear compartment and rear seat	2	6	17	0.98

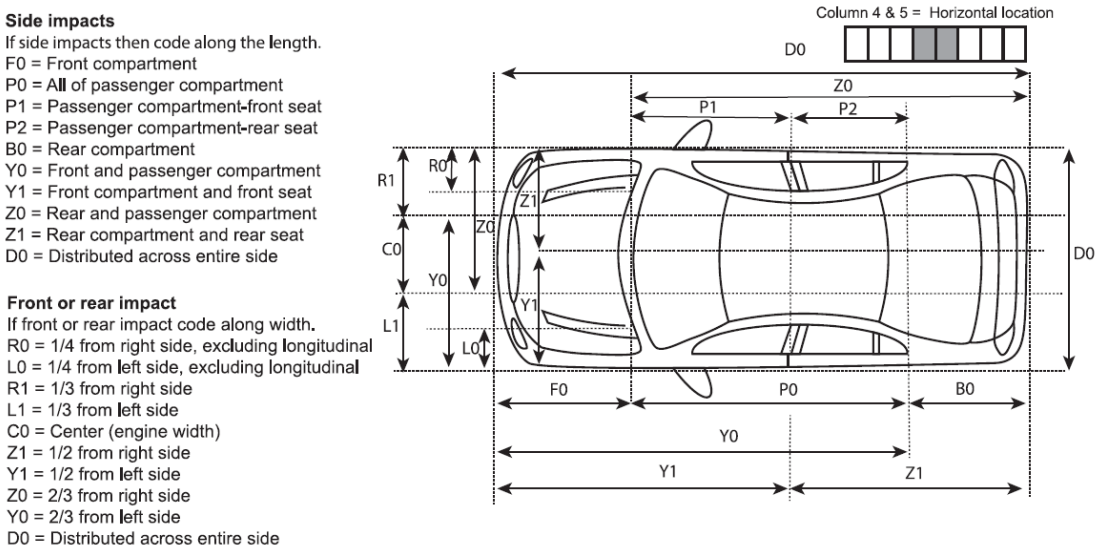


Figure 2.17: Horizontal collision location classification [31]

The Odds Ratio for Fatal and Severe (ORFS) metric which ranks collisions based on the impacted location is calculated with expression (2.10):

$$ORFS = \frac{a/b}{c/d}, \tag{2.10}$$

where a, b, c, d are specified in Table 2.3 as an example for the collision location P_0.

Table 2.3: Example of the ORFS calculation for location P_0

Collision location	Fatal and severe	Minor
P_0	a	b
All other locations	c	d

The final severity classification named by the authors as the cost function $F(P_n(t_c))$ is presented in the equation (2.11), where the cost values are derived based on ORFS values. In this context, equation (2.11), 1 and 12 represent the lowest and highest ORFS values, respectively.

To identify the impact location and assess the severity according to equation (2.11), the authors proposed a division of each vehicle into ten polygons where polygon 10 and 5 represent the front and rear of the car, respectively. The polygon division is illustrated in Figure 2.18. Finally, there is a need to determine which vehicle is the impacted one in order to evaluate the collision location. For this matter, the authors designated the polygon 10 as the impact polygon. For instance in Figure 2.18 the red car's polygon 10 is overlapping the blue car's polygon 8 meaning that the blue car is the impacted one and the collision location is blue car's polygon 8. This location corresponds to a passenger compartment-front seat (P_1) category in equation (2.11), therefore the output severity for this example is four.

$$F\left(P\left(t_c\right)\right) = \begin{cases} 12 & \text{front and passenger compartment } (Y_0) \\ 11 & \text{all of passenger compartment } (P_0) \\ 10 & \text{distributed across entire side } (D_0) \\ 9 & \text{rear and passenger compartment } (Z_0) \\ 8 & \text{rear compartment and rear seat } (Z_1) \\ 7 & \text{front compartment } (F_0) \\ 6 & \text{front compartment and front seat } (Y_1) \\ 5 & \text{rear compartment } (B_0) \\ 4 & \text{passenger compartment-front seat } (P_1) \\ 3 & \text{passenger compartment-rear seat } (P_2) \\ 2 & \text{front to front collision} \\ 1 & \text{front to rear collision} \end{cases} \tag{2.11}$$

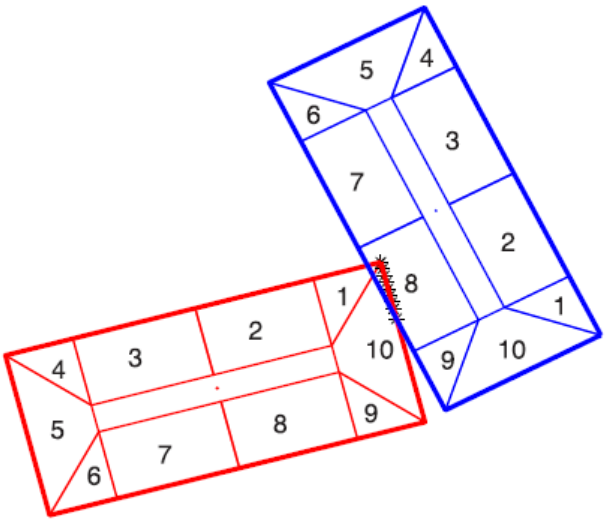


Figure 2.18: Example of P_1 collision between ego vehicle (red) and other vehicle (blue)

2.8 Summary

In this chapter, an analysis of solutions across three main fields, including perception, collision risk, and collision damage, was undertaken.

The perception field is subdivided into four domains, each requiring its own solution:

- **Obstacle detection with 3D LiDAR:** Two methods were analyzed. While both share similar features, the Lidar Obstacle Detector application stands out as it is open source, making it ideal to explore.
- **Obstacle detection and classification with cameras:** Ribeiro's application is ideal to explore because it allows to experiment various deep learning models.

- **Panorama Stitch:** Image Stitching could be very useful as it improves the field of view for the image detector and classifier.
- **Calibration:** ATOM was already used to calibrate sensors in the ATLASCAR2 before. It is ideal because it allows to calibrate various sensors with different modalities at the same time.
- **LiDAR camera combination:** The solution analyzed is useful to combine the LiDAR's data with the image's classifier data and it fits well within this dissertation.

In the collision risk field from all the methods analyzed the ones who have more interest for this dissertation are the open source methods. In this context, the Occupancy Map and the SINADRA solutions are ideal to explore.

In the collision damage field two methods were analyzed. The first method has an equation to evaluate the severity of a crash which could be a potential solution for this dissertation. As for the second method, more specific information is needed, such as the vehicle people occupancy or detailed impact location, which would require additional information and more sensor power.

Chapter 3

Experimental infrastructure

This chapter describes the tools used in this dissertation, both in hardware and software. Hardware wise, two main units were used: ROG laptop and the ATLASCAR2. The latter includes the sensors (two cameras, 3D LiDAR) and an odometry unit. Software wise, a variety of applications such as ROS, Gazebo, RViz, and others were used to process data and achieve results. Additionally, it shows the new hardware implementation and setup in the ATLASCAR2's infrastructure.

3.1 Hardware

This section provides an overview of all the hardware components used in this dissertation and their characteristics.

3.1.1 ATLASCAR2

ATLASCAR2 is a fully electric Mitsubishi i-MiEV [33] destined exclusively for investigation purposes in the Mechanical Engineering department at the Aveiro university (Figure 3.1). The vehicle is equipped with various sensors. The sensors installed in the ATLASCAR2 that have the most interest for this dissertation are described in the following subsections.

Figure 3.1: ATLASCAR2

3.1.1.1 Camera Point Grey FL3-GE-28S4-C

The Point Grey FL3-GE-28S4-C (Figure 3.2) is a 2.8 Megapixel color GigE Vision digital camera that uses a Sony ICX687 EXview HAD CCD II image sensor to deliver high resolution, high-quality images in a compact and low-cost package. At its highest resolution of 1928 × 1448, the camera runs at 15 FPS. However, it is possible to decrease its region of interest to obtain more frames per second [34]. Table 3.1 describes the main specifications of this sensor.

Figure 3.2: Camera Point Grey FL3-GE-28S4-C [34]

Table 3.1: Point Grey FL3-GE-28S4-C specifications

Specifications	
Resolution	1928 × 1448
Frame rate	15 FPS @ 1928 × 1448
Megapixels	2.8 MP
Chroma	Color
Sensor	Sony ICX687 EXview HAD CCD II
Dimensions	29 × 29 × 30 mm
Weight	38 g

3.1.1.2 Camera Logitech C270 HD

The Logitech C270 webcam [35] (Figure 3.3) is a versatile camera designed to meet various needs. Its sleek and compact design allows for easy portability, making it easy to use in various environments. Featuring a resolution of 720p HD, it delivers clear video quality. Additionally, it is a plug-and-play camera which does not require drivers or software installations, making the setup quick and easy. Table 3.2 describes the main specifications of this sensor.

Table 3.2: Logitech C270 HD specifications

Specifications	
Resolution	1280 × 960 1.2 MP
Frame rate	30 FPS @ 640 × 480
FOV	60°
Focal length	4 mm

Figure 3.3: Logitech C270 HD

3.1.1.3 Odometry unit

The odometry unit was developed for the ATLASCAR2 as part of Sara Pombinho's master's dissertation [36]. This unit comprises a RI32-0/1000ER.14KB wheel encoder [37], an Arduino UNO wifi rev2 [38] with a CAN-BUS shield [39], and a CANalyze device [40]. The Arduino counts the electrical pulses transmitted by the encoder, incrementing or decrementing a position variable depending on the rotation direction of the encoder. This position data is then sent via the CAN-BUS shield to the CANalyze device. Additionally, the Controller Area Network (CAN) messages from ATLASCAR2 are also provided to the CANalyze device. This device processes the CAN packets transforming them into ROS messages. Pombinho combined the steering angle information from the vehicle's CAN bus with the encoder's positions to compute the odometry. The odometry messages include the ATLASCAR2's speed, which is the piece of information needed from this unit for this dissertation. The ATLASCAR2's speed (V) is obtained through the following equations (3.1), (3.2), (3.3):

$$PPS = \frac{\Delta Pulses}{\Delta t} \tag{3.1}$$

$$RPS = \frac{PPS}{PPR} \tag{3.2}$$

$$V = RPS \times P \tag{3.3}$$

where PPS is the encoder rotational speed obtained by the difference between two consecutive encoder positions $\Delta Pulses$ divided by the correspondent time interval Δt. RPS is the number of rotations per second obtained by dividing the encoder rotational speed by the maximum number of rotations per second PPR. Lastly, the speed V is obtained by multiplying the number of rotations per second by the wheel's perimeter P.

3.1.2 Asus ROG Strix G15 G512LI-70AT5PB1 laptop

The Asus ROG Strix G15 G512LI-70AT5PB1 laptop [41], illustrated in Figure 3.4, serves as the cornerstone hardware element within this dissertation. It enabled the reading and processing of all the sensor data, allowing the development of the proposed solution (Chapter 4). The laptop's main characteristics are presented in Table 3.3.

Figure 3.4: Asus ROG Strix G15 G512LI-70AT5PB1

Table 3.3: Asus ROG Strix G15 G512LI-70AT5PB1 laptop's specifications

Specifications	
CPU	Intel® Core™ i7-10750H Hexa Core
GPU	NVIDIA GeForce GTX 1650 Ti
RAM	16 GB
Storage	1 TB SSD

3.2 Software

The software described in this section represents the primary tools utilized in this dissertation. Other specifics, such as ROS packages, which were also used, are detailed in Chapter 4.

3.2.1 Robot Operating System

ROS is an open-source framework [42] designed to facilitate robot software development. It provides tools, libraries, and conventions that help software developers create and manage complex robotic systems. One of the key elements of ROS is its distributed computing architecture. It utilizes a graph-like structure where nodes, which are individual software processes, communicate with each other through topics, services, and actions. This communication is facilitated by a publish-subscribe messaging system, allowing nodes to send and receive data efficiently. Figure 3.5 shows an example of the ROS communication between two nodes. The node talker publishes a message "Hello, World" through the topic `/talker` to which the node listener subscribes.

3.2.2 ROS Visualization

ROS Visualization (RViz) [43] is a ROS graphical interface that allows the user to visualize sensor data, robot models, and other information in a 3D environment. The user can configure and customize visualizations, change perspectives and overlay different types of data, helping the analysis and debug of complex robotic systems. Figure 3.6 shows an example of the RViz interface. On the left there's the display section where

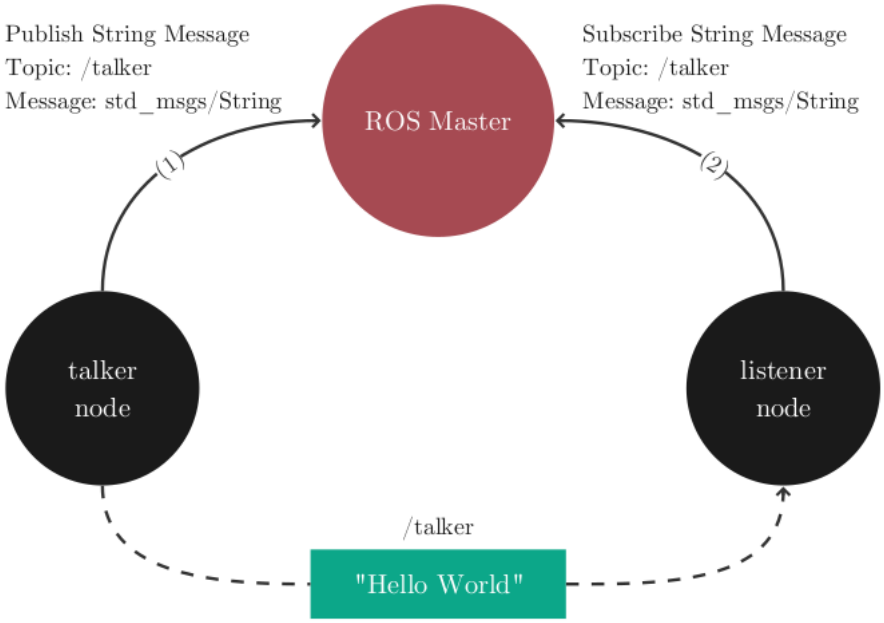


Figure 3.5: Schematic example of the use of ROS

global definitions are set and ROS topics are subscribed for visualization. In this case, there's an image, a point cloud and a robot model being displayed. On the right side, there's a section where the user can choose the view being used.

3.2.3 Gazebo Simulator

Gazebo [44] is an open-source 3D robotics simulator widely used in research, education, and industry. It provides a versatile platform for creating virtual environments, designing robots, and conducting simulations to assess the performance of robotic systems. With Gazebo, users can simulate realistic interactions between robots and their environments, including dynamic physics, sensor feedback, and environmental factors such as lighting and terrain. One of Gazebo's key strengths is its support for the integration of multiple sensors, plugins, and other tools, allowing developers to create complex robotic systems and test them in a simulated environment. Figure 3.7 shows an example of a scenario in Gazebo Simulator.

3.3 3D LiDAR implementation and setup in the ATLASCAR2's infrastructure

The 3D sensor named Velodyne LiDAR's Puck [45] is a small and compact LiDAR with sixteen channels (Figure 3.8) used across a variety of applications ranging from automotive, mapping, robotics and more. It provides surround 3D view in a form of a point cloud. The range goes up till 100 m, across a 360° horizontal field of view and a 30° vertical field of view. Moreover, it has an integrated web server for monitoring and configuration. Table 3.4 shows the LiDAR's main properties.

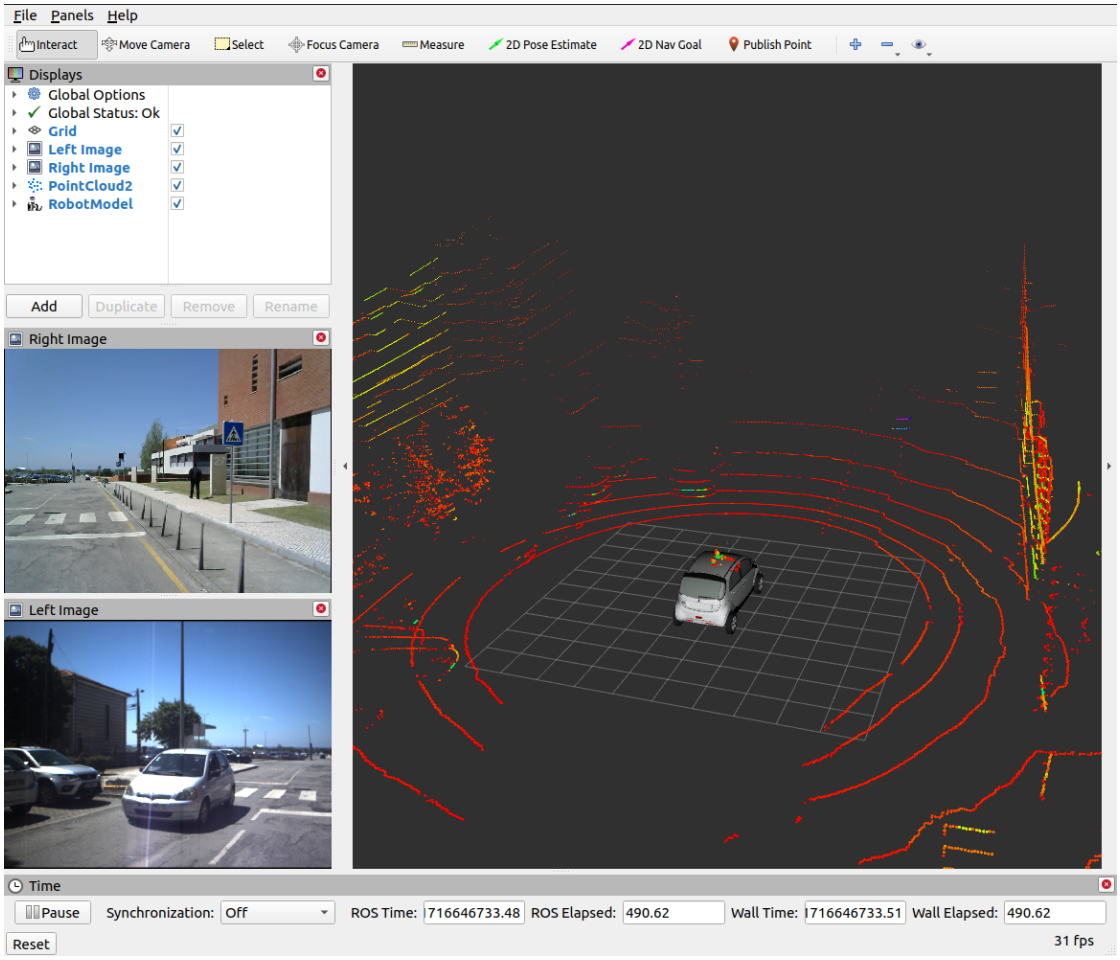


Figure 3.6: Example of an RViz interface used for real data collection with the ATLAS-CAR2

Table 3.4: Velodyne Puck VLP-16 specifications

Specifications	
Channels	16
Measurement range	100 m
Range accuracy	±3 cm
Vertical FOV	15° to −15° (30°)
Vertical angular resolution	2°
Horizontal FOV	360°
Horizontal angular resolution	0.1° – 0.4°
Rotation rate	5 Hz to 20 Hz
Operating voltage	9 – 32 V

In the setup, an important concern was to place the Velodyne LiDAR as high as possible, positioning it centrally between the two cameras without occluding its channels by the cameras position. Figure 3.9 illustrates the proposed setup, where all the sensors

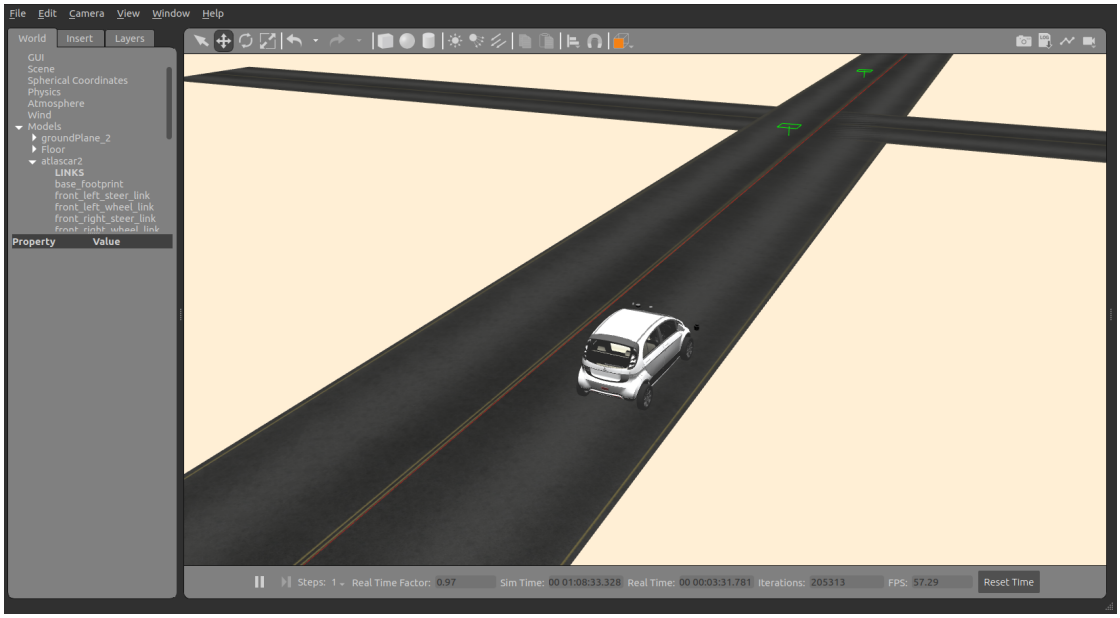


Figure 3.7: Example of a scenario in Gazebo simulator

Figure 3.8: Velodyne Puck VLP-16

were attached to an aluminium profile, which was in turn fixed within the two existing profiles on the car's roof. The length of the aluminium profile was also checked for possible scan occlusion. Both right (Point Grey) and left (USB) cameras were placed under the profile for the same purpose. Additionally, the cameras were positioned to maximize the field of view while ensuring sufficient overlap for the stitching method (Section 2.3) to function effectively.

In terms of connectivity, the Point Grey camera and the Velodyne LiDAR connect to the car roof's Ethernet switch, while the USB camera connects directly to the computer. The IPs set of the former and the laptop are:

- Point Grey camera: 169.254.0.5
- Velodyne LiDAR: 169.254.0.4
- Computer: 169.254.0.3

In terms of power supply, the Point Grey camera is connected to the 12 V supply placed near the roof switch, that comes from the electrical panel. As for the Velodyne LiDAR, a temporary solution was adopted, involving its connection to the power inverter

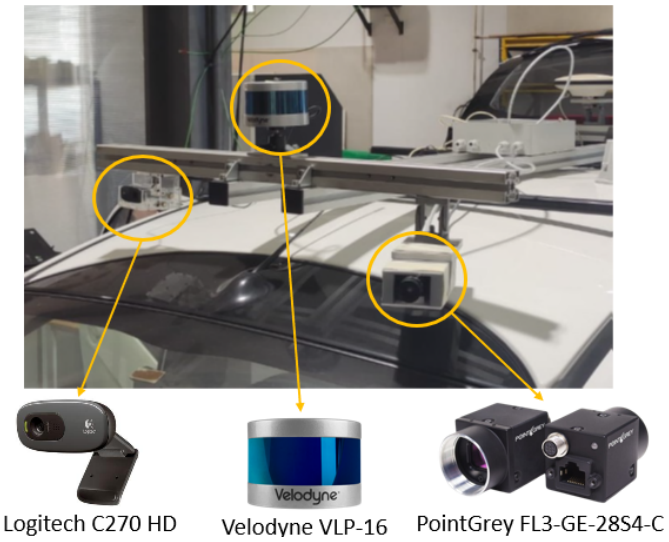


Figure 3.9: Velodyne and cameras setup in the ATLASCAR2

(12 DC – 220 AC). The optimal solution would involve directly connecting the LiDAR to the 12 V power supply, mirroring the setup of the Point Grey camera, but the lack of time made it impossible.

In terms of sensor parameters, the Velodyne LiDAR was set to the default. As for the cameras, similar resolutions and frame rates were chosen to ensure consistent data and facilitate the synchronization procedure. Table 3.5 show the proposed settings.

Table 3.5: Point Grey and USB camera proposed parameters

Parameter	Point Grey	USB
Resolution	964×724	960×720
Frame Rate	10 fps	10 fps

Chapter 4

Proposed solution

This chapter focuses on a solution to address the central problem of this dissertation: a new concept named Risk Maps which displays the collision zones that a vehicle may face in specific scenarios, along with quantitative metrics that indicate the probability of these events. Given that this feature is novel in ATLASCAR2, with no similar applications developed in the past, this chapter proceeds to explore the whole process that led to the conception and implementation of the Risk Maps. It was developed in the Gazebo simulator [44] since the ATLASCAR2 repository (Section 2.1) had already packages and tools for this environment.

4.1 Collision Risk method

From the two open source methods of the state of the art (Chapter 2) about collision risk assessment methods, one was best aligned with the goals of the dissertation. The method is the Occupancy Map referred in section 2.6.2. It was chosen because it provides a predicted occupancy map for the future positions of obstacles and the ego vehicle in the scene, and the authors [21] made available their code [46], which was a good starting point for the intended Risk Maps. Additionally, the construction of the Occupancy Map only relies on instantaneous information about the obstacles, which is obtainable from the sensors.

In the first solution [21], the authors tested the Occupancy Map approach on two different scenarios: a simulated environment using CARLA [47], and ROS bags from the KITTI dataset [48]. In both situations the authors did not use sensors to obtain information about the obstacles, as that information was given by the simulator or present in the ROS bag files. In contrast, this dissertation utilizes obstacle information gathered from sensors installed in the ATLASCAR2. Section 4.2 provides an insight into how this information is obtained.

The above mentioned Occupancy Map is displayed using the Grid Map ROS package [49] in RViz as a 3D surface plot. The grid size for the proposed solution is 40×40 meters with 0.1 m cell resolution, meaning that 10 cells equal 1 meter. Each cell holds a numeric value ranging from 0 to 1, with the height of the cells directly related to their value. Figure 4.1 shows the mentioned Grid Map and the location of the ATLASCAR2 in it.

In [21], three distinct methods are presented for calculating probability values for various entities: one for vehicles, one for pedestrians and the last for the ego vehicle tra-

jectory. The authors consider any obstacle with a speed below $1\,\mathrm{m\,s^{-1}}$ to be stationary, regardless of whether it is a vehicle or a pedestrian, and for these cases they mention that simple kinematics projections can be used, which is not a case addressed in their work. Additionally, the predicted obstacle distributions are represented in the world frame rather than the car's frame.

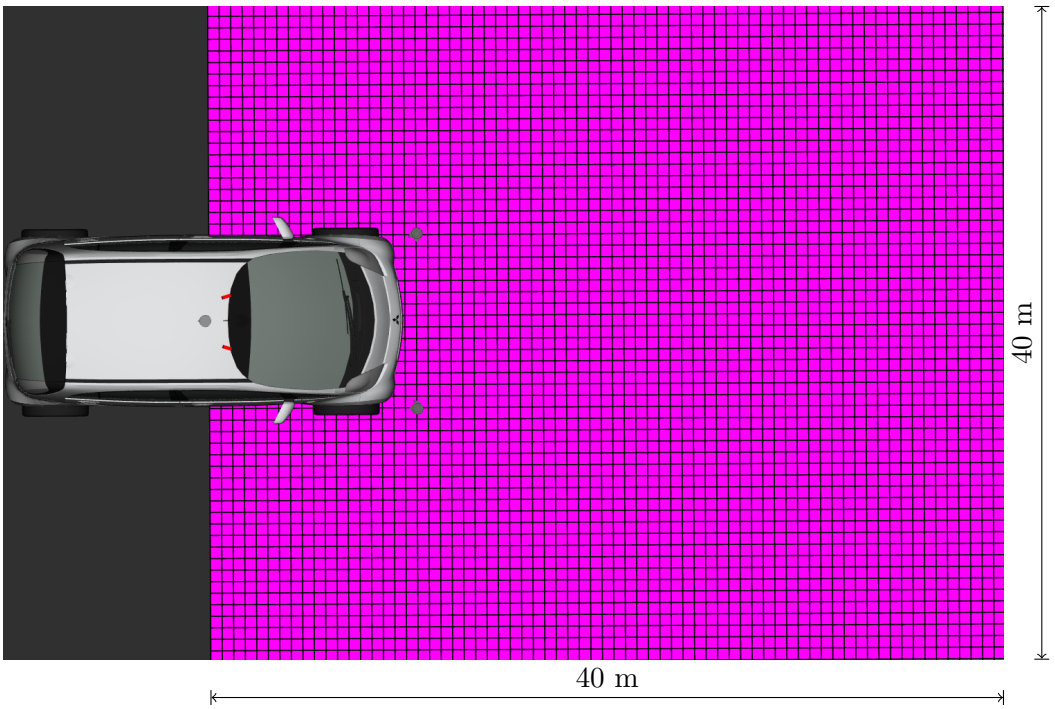


Figure 4.1: Grid map in Rviz with ATLASCAR2 robot model

Follows a detailed description of each model:

- **Vehicle model:** As illustrated in Figure 2.11, there are two types of occupancy distributions: radial and angular. The cell probability values for these distributions are achieved through equations (4.1) and (4.4), respectively:

$$P\left(D_{\text{actual}} \mid u, a\right) = \frac{1}{K_R}\left(1 - \frac{\Delta D^2}{\sigma_R}\right) \tag{4.1}$$

where K_R is a normalizing constant, ΔD and σ_R (radial support) are defined in equations (4.2), (4.3):

$$\Delta D = u_i \cdot t + 0.5 \cdot a_i \cdot t^2 \tag{4.2}$$

$$\sigma_R = \frac{1}{c_f}\left(u_i \cdot t \cdot \frac{u_i - 1}{u_i + 1} + \frac{a_i \cdot t^2}{2} \cdot \frac{a_i - 1}{a_i + 1}\right) \tag{4.3}$$

with u_i and a_i as instantaneous values of the speed (m/s) and acceleration (m/s^2), t as the time horizon (s) and c_f as a constant that depends on the class of the vehicle. The angular distribution is given by expression (4.4):

$$P(\Delta\theta \mid u, a, \dot{\psi}) = \frac{1}{K_A}\left(1 - \frac{(\Delta\theta - \dot{\psi}t)^2}{\sigma_A}\right) \tag{4.4}$$

where K_A is a normalizing constant, $\Delta\theta$ is the difference between current heading and next heading (rad), $\dot{\psi}$ is the angular velocity (rad/s) and σ_A is the angular support [21], in this case as a constant.
The final cell probabilities result from the product between radial and angular probability. Furthermore, for each cell with a value higher than 0, an oriented rectangle is formed with the dimensions of the vehicle's bounding box. The final distribution comes from the overlapping area of all the rectangles. Figure 4.2 shows an example of the final distribution for a car at constant speed of 2.5 m/s and time horizon of 2 s.

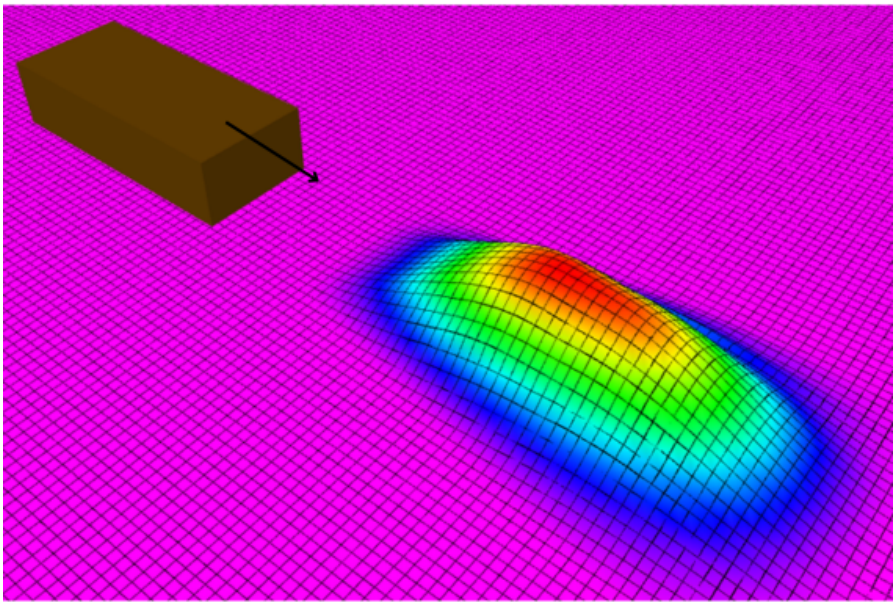


Figure 4.2: Car probabilistic distribution prediction. The brown bounding box indicates the current position, while the black arrow shows the orientation

- **Pedestrian model:** Similarly to the vehicle model, for pedestrians there is also a radial and an angular distribution combination, but with different equations as the motion model for a pedestrian is different from the one of a vehicle. The speed limits for the pedestrian model defined by the authors are 1 and 3 m/s, as minimum and maximum. According to [50], the average walking speed ranges from 0.94 to 1.43 $\mathrm{m\,s^{-1}}$. To cover for more situations, particularly for older individuals, the minimum threshold was set to 0.9 m/s. Equations (4.5) and (4.6) give the probability value for the radial and angular distribution, respectively.

$$P(d \mid D) = \frac{1}{K_P}\left(1 - \frac{(d-D)^2}{D_{\max}}\right) \tag{4.5}$$

$$P\left(\Delta\theta \mid \theta_t\right) = \frac{1}{K_N}\left(1 - \left|\sin\left(\frac{\Delta\theta}{2}\right)\right|\right) \tag{4.6}$$

with both K_P and K_N as normalizing constants, d, D, D_{max} as the current, expected and maximum distances to be covered by the pedestrian (Figure 2.11), $\Delta\theta$ as the variation in heading angles between current and future position.

The final cell probabilities result from the product between radial and angular probability, as stated in the author's paper [21]. Figure 4.3 shows an example of the final distribution for a pedestrian at constant speed of 0.9 m/s and time horizon of 2 s.

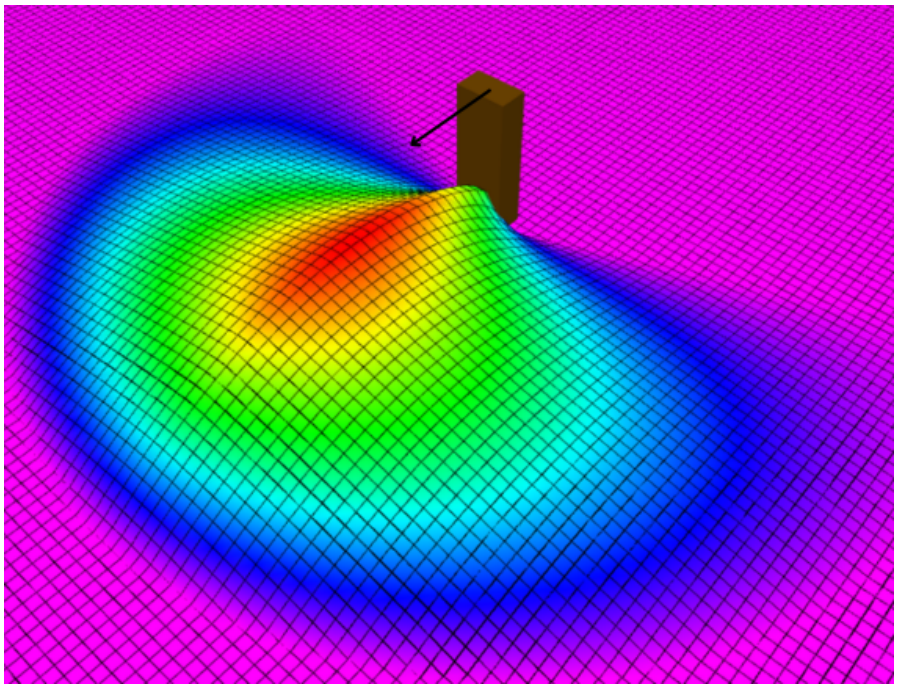


Figure 4.3: Pedestrian probabilistic distribution prediction. The brown bounding box indicates the current position, while the black arrow shows the orientation

- **Ego vehicle trajectory:** The authors use both v_x and v_y of the ego vehicle to display its trajectory. However, in the proposed solution for this dissertation, considering v_y is unnecessary. Therefore, equation (4.7) is used to calculate the distance (in cells) that the ego vehicle will cover in the given time horizon. Then, for every x value of that distance a range of cells in y, proportional to the car's width, have value of 1. Figure 4.4 illustrates the mentioned trajectory for a speed of 4 m/s with a 2 s time horizon.

$$dx = v_{sx} \cdot n \cdot t \tag{4.7}$$

where v_{sx} is the ego vehicle's speed in x direction, n is the number of cells per meter, and t the time horizon.

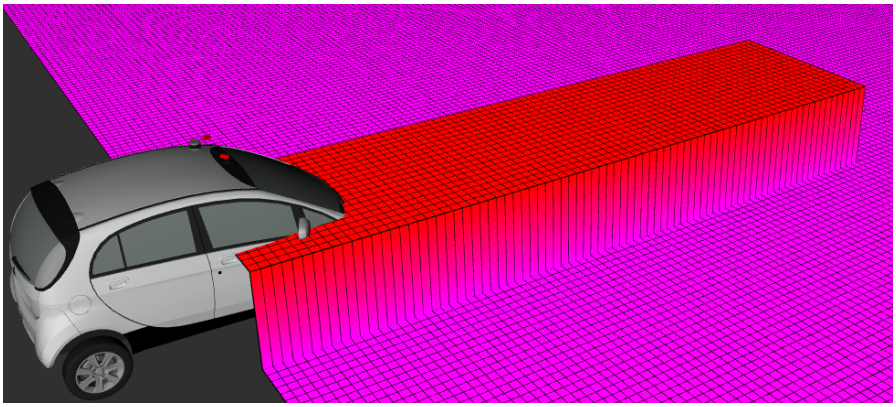


Figure 4.4: ATLASCAR2's trajectory

The problem with this representation is that the vehicle is not actually occupying the whole path covered at the moment of 2 s. Instead, a representation of the ATLASCAR2's occupation for the given time horizon is more accurate. Figure 4.5 portrays the mentioned occupation chosen for the proposed solution.

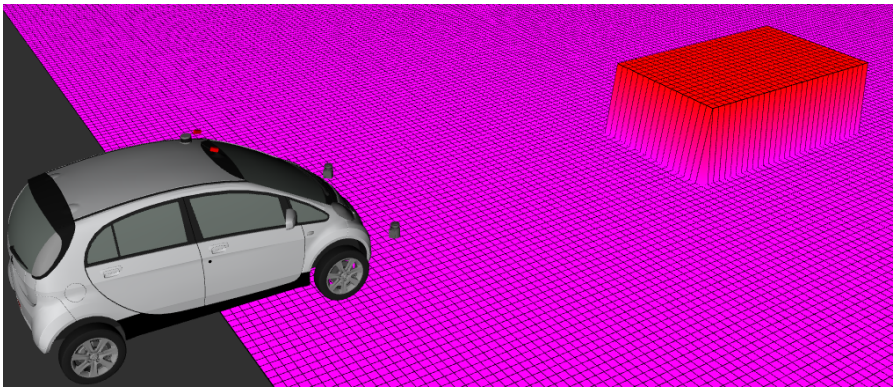


Figure 4.5: ATLASCAR2's future occupancy

An additional model was developed for the proposed solution which represents the probabilistic distribution of stopped obstacles. The model only considers the obstacle's bounding box dimensions and centroid. Figure 4.6 illustrates the stopped model for the probabilistic distribution. The inner red rectangle represents the obstacle's bounding box and the outer black rectangle represents the probabilistic distribution. The difference in width and length is 5 cells in each direction. This cell value provides a reasonable offset for the bounding box.

As expected, since the obstacles are stopped, the final cell probabilities have the value of 1. Figure 4.7 shows the stopped model distribution of a car in Gazebo.

Lastly, in the original author's method, the collision risk is determined by identifying the maximum value within the intersecting cells of the ATLASCAR2's distribution and the obstacle's distribution for the specified time horizon. In this dissertation other metrics will be explored aside the maximum metric in Chapter 5. Figure 4.8 shows the mentioned intersection for a 2 s time horizon.

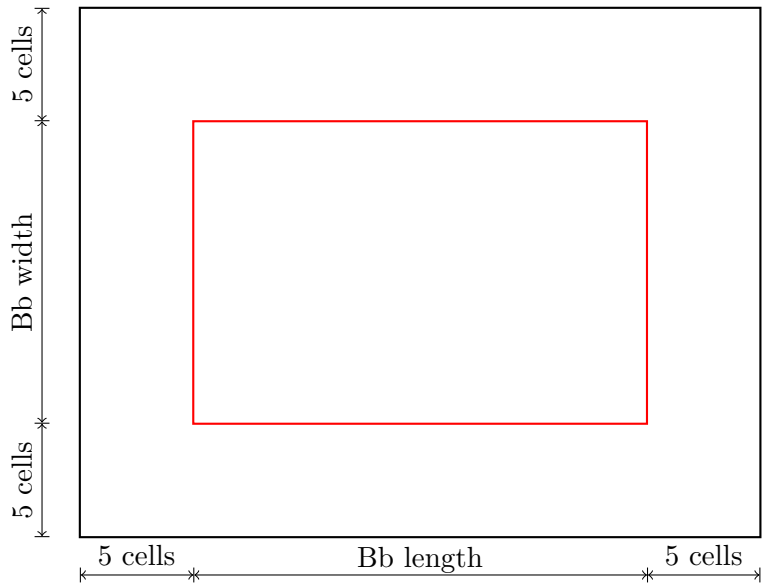


Figure 4.6: Stopped obstacles model (Bb stands for Bounding box)

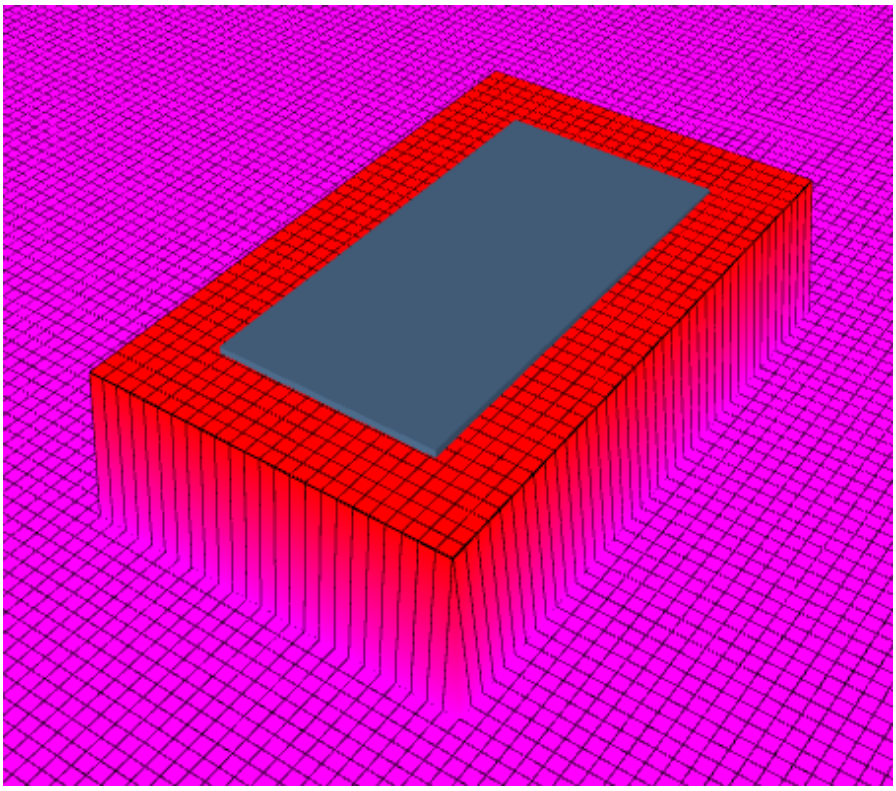


Figure 4.7: Stopped probabilistic distribution of a car, with the bounding box illustrated in blue

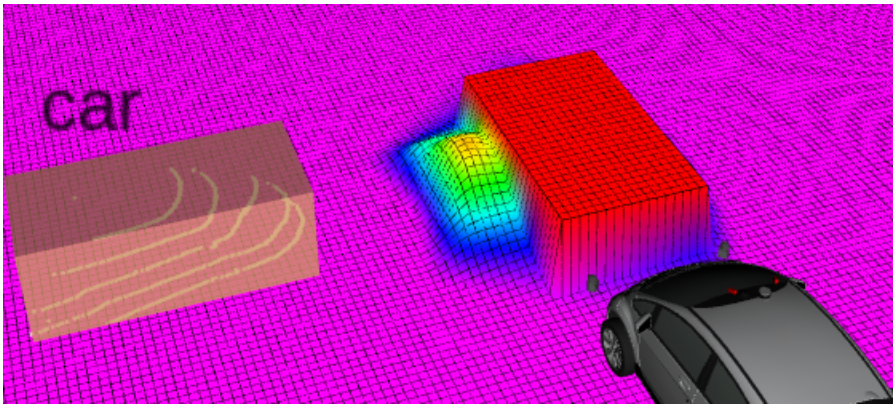


Figure 4.8: ATLASCAR2's distribution intercepting another car's distribution for a 2 s time horizon

4.2 Perception

This section provides an overview of how information about road obstacles is obtained through the sensors on the ATLASCAR2. This data serves as input for the Occupancy map discussed above. As stated in Section 2.6.2, the needed data is:

- Position (x, y) relative to the ATLASCAR2
- Speed (v_x, v_y)
- Acceleration (a_x, a_y)
- Orientation angle (yaw)
- Dimensions (x, y)
- Obstacle's nature (e.g. pedestrian, car, ...)
- ATLASCAR2's speed (v_x)
- Time horizon (time in the future for which the prediction is made)

ATLASCAR2's speed is obtained through the ROS package called Ackermann steering controller [51] which commands the vehicle movement from velocity "Twist" messages [52]. Furthermore, this controller publishes the vehicle's odometry data through "Odometry" messages [53], allowing for the retrieval of the vehicle's speed from these messages. The user sets the time horizon, while the remaining topics are determined by data collected from sensors. Figure 4.9 shows the ROS framework visualized using `rqt_graph`, a tool provided by ROS [54]. This tool displays how nodes and topics are hierarchically connected, starting with Gazebo and ending in a Risk Maps grid.

4.2.1 LiDAR

As described in Section 2.2, the Lidar obstacle detector application outputs the 3D bounding boxes of the segmented obstacles as a ROS topic. The messages published in this topic have the information about the obstacle's centroid position (x, y, z), the bounding box's ID, and its dimensions (x, y, z). Having the position and the time interval between messages it is possible to get the speed and acceleration. The perception of

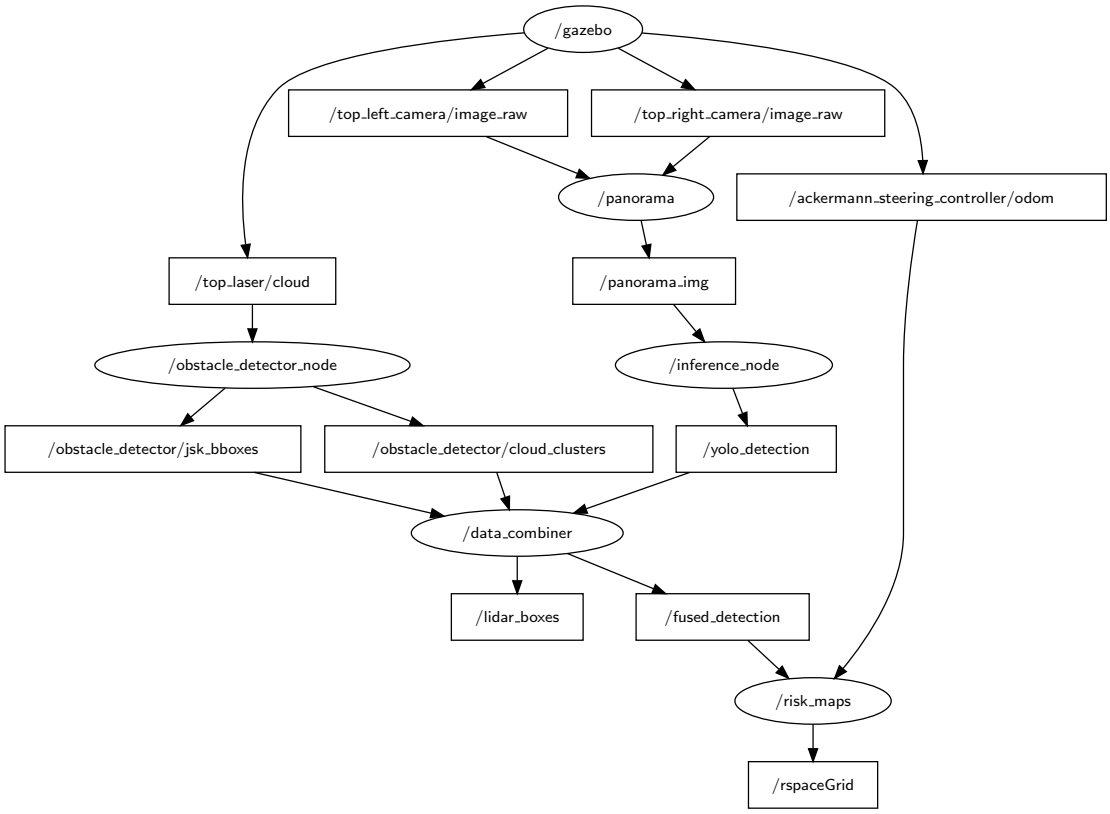


Figure 4.9: Risk Maps ROS framework showcasing the connections between nodes (round shapes) and topics (rectangular shapes)

speed varies depending on the frame of reference used, whether it is the world frame or the car frame. Absolute speed is derived from the combined velocities of obstacles and ATLASCAR2. Similarly, relative orientation is determined by consecutive relative positions in the car frame, whereas in the world frame, it is inferred from the speed vector.

The point cloud segmentation exhibits fluctuations, particularly in the boundary zone between ground and obstacle points. As a result, erroneous bounding boxes may appear in areas without obstacles. The way to tackle this issue was ensuring that bounding boxes having a centroid with a negative z value are not taken into account in further processing actions. Figure 4.10 shows the best fit parameters for the obstacle segmentation.

In comparison with Figure 2.4, the parameters defining the region of interest in Figure 4.10 were adjusted to align with the grid dimensions (Figure 4.1), with a slight modification in the roi_max_x from 40 to 30 due to uncertainty. The roi_min_z and $ground\_threshold$ values were set accordingly in order to minimize the fluctuations. The $cluster\_threshold$ was set to its maximum value to ensure that objects located farther away from the car, where there are fewer points defining the cluster, are encompassed within a single bounding box. The $voxel\_grid\_size$ was reduced slightly, to improve obstacle definition. Additionally, the tracking formulation regarding the $iou\_threshold$ suffered a minor change where the z component was removed. The calculation of the IoU is carried through the equation present in Algorithm 1. With $iou\_threshold$ set to 1, it

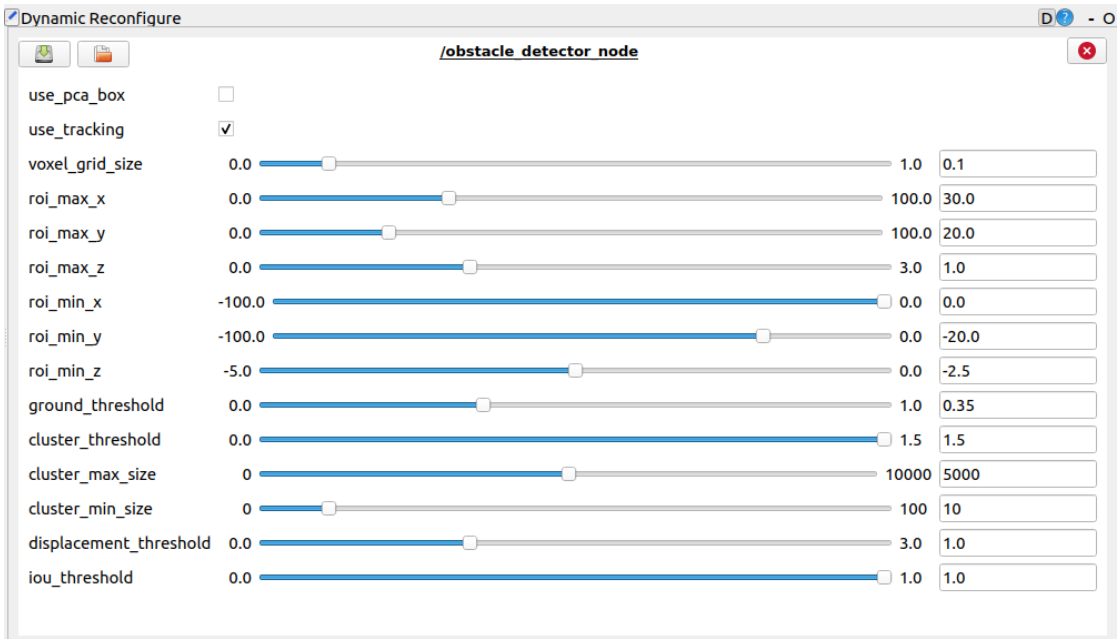


Figure 4.10: Proposed LiDAR clustering parameters

indicates that the dimensions of the current bounding box can be as small as one-third of the dimensions of the previous bounding box, reaching the limit. This allows for good tracking in general, except in the transition zones between LiDAR channels where the bounding box height can drop to 0, leading to tracking failures. Therefore, removing the z dimension from Algorithm 1 solved this exceptional problem, enabling good tracking overall. Figure 4.11 shows the resulting bounding boxes.

Algorithm 1 Tracking IoU calculation

Require: Previous bounding box a, Current bounding box b

1: **for** each bounding box dimension in $\{x, y, z\}$ **do**
2: $\quad a_{\text{dim}} = a.\text{dimension}$
3: $\quad b_{\text{dim}} = b.\text{dimension}$
4: $\quad \text{IoU}_{\text{dim}} = 2\left(\frac{a_{\text{dim}} - b_{\text{dim}}}{a_{\text{dim}} + b_{\text{dim}}}\right)$
5: **end for**

4.2.2 Camera

The purpose of the cameras is to identify the nature of road obstacles, which is the final feature required for the Occupancy map (Section 4.1). Within this framework, Ribeiro's application [11], as described in Section 2.3, allows to experiment various deep learning models for object detection and classification. Three models were tested: YOLOv5 [55], YOLOv7 [56] and YOLOv8 [12]. Since YOLOv7 exhibited the best detection performance across different classes, it was the chosen model.

Another matter related to the camera was the fact that ATLASCAR2 had two RGB cameras installed in its infrastructure. However, the initial field of view was sub-optimal,

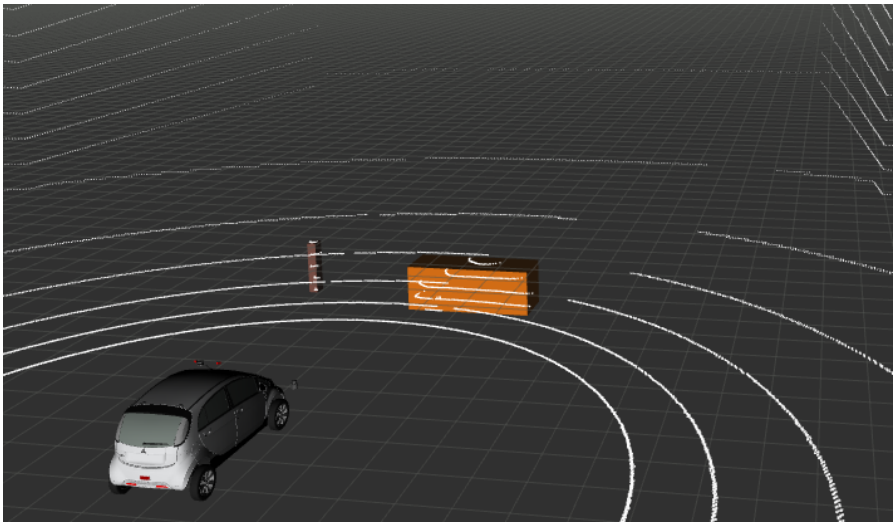
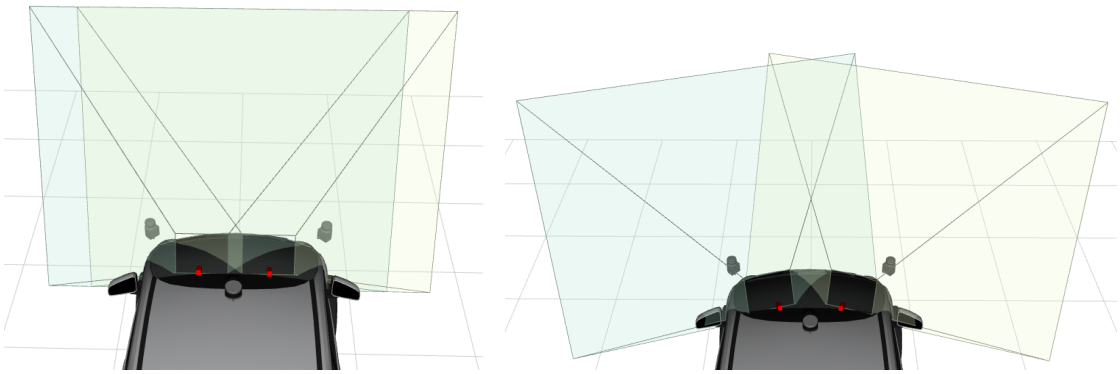


Figure 4.11: Example of Lidar Obstacle Detector object segmentation: Pedestrian on the left, car on the right.

failing to fully utilize the potential of having two cameras. For this reason, the cameras position was changed. Figure 4.12 shows the initial and the proposed setup.



(a) Initial setup (b) Proposed setup

Figure 4.12: Changes in cameras setup and their respective field of view

The issue with the new setup is that the front area of the ATLASCAR2 is poorly defined in each camera filed of view, which can lead to failed obstacle detections as only parts of obstacles are visible. To address this issue, the idea was to apply the image stitching method described in Section 2.3, to transform the two camera images into a single panoramic view. Algorithm 2 describes the stitching process based on the OpenCV library [57]. Before running the algorithm, synchronization of both camera's image is mandatory. To achieve this, ROS message filters [58] were used, specifically the "TimeSynchronizer" function. This function queues the images from both cameras and invokes the callback function only when it finds a pair of images with matching timestamps. In simulation, this step is not necessary, but it will be very valuable when processing real data.

Algorithm 2 begins by detecting keypoints and extracting features from the input images using the `detect()` function. These keypoints and features are then matched between the images using the `match()` function. If four good matches are found (computing the homography requires at least four matches), then the algorithm proceeds to estimate a homography transformation using the matched keypoints. This homography is then used to warp the right image so that the transition between the left and right images is smooth. The resulted panorama is outputted and showed to the user for each new frame.

Algorithm 2 Panorama Stitching Algorithm

Require: Two input images $image1$ and $image2$
Ensure: Panorama image $panorama$

Function detect($image$):
    $keypoints, features \leftarrow$ cv2.SIFT_create().detectAndCompute($image$, None)

return $keypoints, features$

Function match($features1$, $features2$):
    $matcher \leftarrow$ cv2.DescriptorMatcher_create("BruteForce")
    $matches \leftarrow$ matcher.match($features1, features2$)
    $good\_matches \leftarrow$ filter $matches$ to keep only the best ones
if len($good\_matches$) ≥ 4 **then**
    $src\_points \leftarrow$ keypoints from $image1$ corresponding to good matches
    $dst\_points \leftarrow$ keypoints from $image2$ corresponding to good matches
    $homography \leftarrow$ cv2.findHomography(src_points, dst_points)

    return $homography$
end if

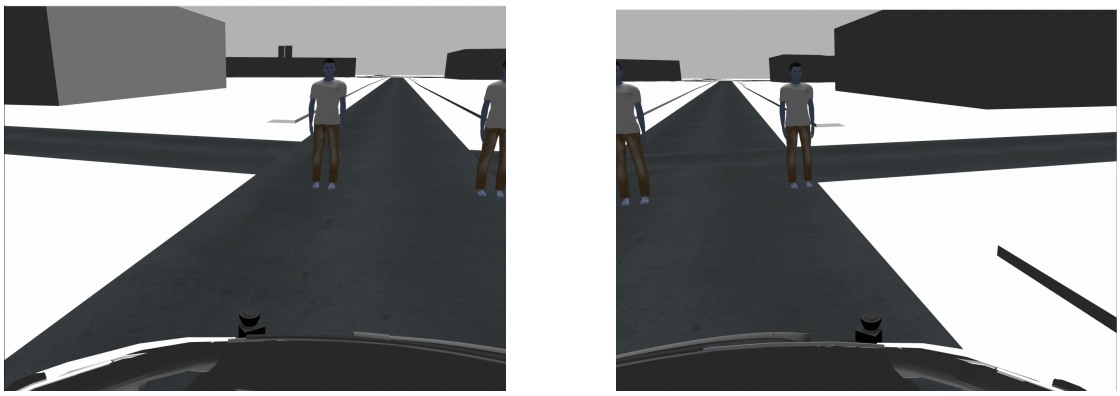
Function stitch($image1$, $image2$):
    $keypoints1, features1 \leftarrow$ detect($image1$)
    $keypoints2, features2 \leftarrow$ detect($image2$)
    $homography \leftarrow$ match($features1, features2$)
if $homography \neq$ None **then**
    $sum\_width = image1.width + image2.width$
    $max\_height = \max(image1.height, image2.height)$
    $panorama.size = (sum\_width, max\_height)$
    $panorama \leftarrow$ cv2.warpPerspective($image2, homography, panorama.size$)
    $panorama[0 : max\_height, 0 : image1.width] = image1$

    return $panorama$
end if

Once the transition appeared smooth, the process was stopped, and the homography matrix was saved. The homography matrix obtained for the proposed solution is present in equation (4.8).

$$\text{Homography matrix} = \begin{bmatrix} 0.458 & -0.137 & 870.350 \\ -0.061 & 0.920 & 14.989 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{4.8}$$

To evaluate qualitatively the transition smoothness, pedestrians were chosen as they provide a clear indication of the transition while also posing the most challenging obstacle for stitching. Figure 4.13 shows the setup for the stitching process, and Figure 4.14 shows the resulting panorama image.



(a) Left camera image (b) Right camera image

Figure 4.13: Setup for the panorama stitching method

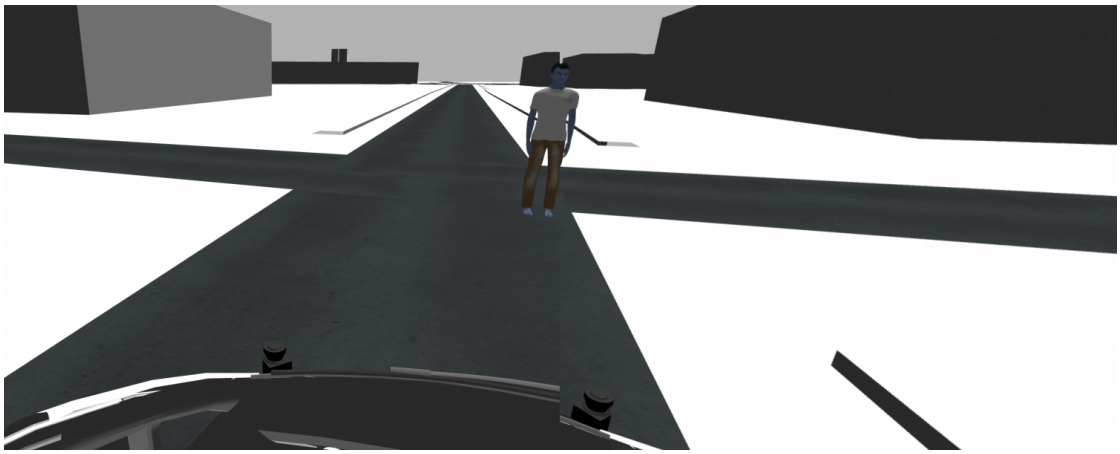
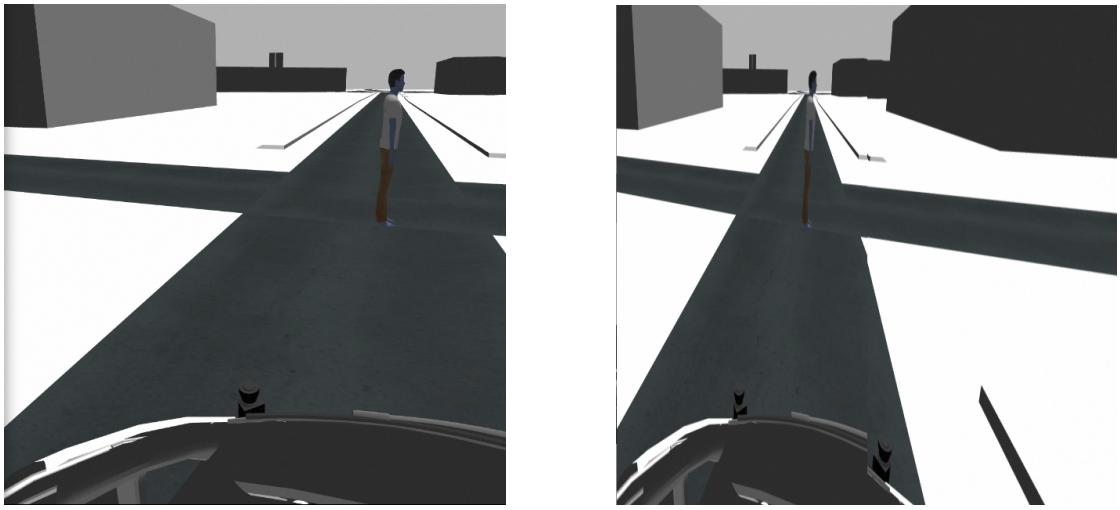


Figure 4.14: Panorama image

A new problem appeared when passing the panorama image to the YOLOv7 for the object detection and classification leading to failed detections. The panorama image has the same height but has a width approximately equal to the sum of the widths of each image. Consequently, when YOLOv7 resizes to 640×640, as it does for every image before detection, the objects become shrunk (highly distorted aspect ratio), and YOLOv7 does no longer recognize them. Figure 4.15 shows the difference in the display of a person in the left camera image and in the panorama image when resized to 640×640 by YOLOv7.



(a) Left camera image 640 × 640 (b) Panorama image 640 × 640

Figure 4.15: Comparison of person's display between left camera image and panorama image when resized to 640 × 640

The approach to solve this issue was to divide the panorama into two images both covering well the front area of the ATLASCAR2 (75 % of the panorama), as shown in Figure 4.16. Since the panorama is smaller in each rectangle the object aspect ratio change is not so marked, and YOLOv7 is able to recognize it. Figure 4.17 shows the difference in the person display between the original panorama and the red rectangle panorama.

Afterwards, YOLOv7 performs object detection and classification in each rectangle image. Subsequently, redundant detections are eliminated by comparing the corner points of bounding boxes in both images. Boxes with matching heights and widths differing by less than 5 pixels are discarded. The resulting detections and classifications of the four relevant classes for the proposed solution in the panorama image are represented in Figure 4.18.

4.2.3 Camera-LiDAR data combination

The final step of the perception section involves associating each detected object class with its corresponding bounding box. The method analyzed in Section 2.5 achieves this by projecting the LiDAR cluster points into the image frame and matching them with classifier's bounding boxes, which fits well the proposed solution. As illustrated in Figure 4.9, the 3 topics arriving to the data combiner node are `/obstacle_detector/jsk_boxes`, `/obstacle_detector/cloud_clusters` and the `/yolo_detection`. The first two transport the LiDAR bounding boxes and the point cloud of the clusters, respectively. The latter transports the YOLO detections on the panorama image, as well as the corresponding images for those detections. It is on this image frames that the LiDAR points will be projected, avoiding the need for an extra synchronization. ROS message filters are used again to synchronize these three topics, which is necessary, because the YOLO inference is slightly slower than the LiDAR obstacle detector.

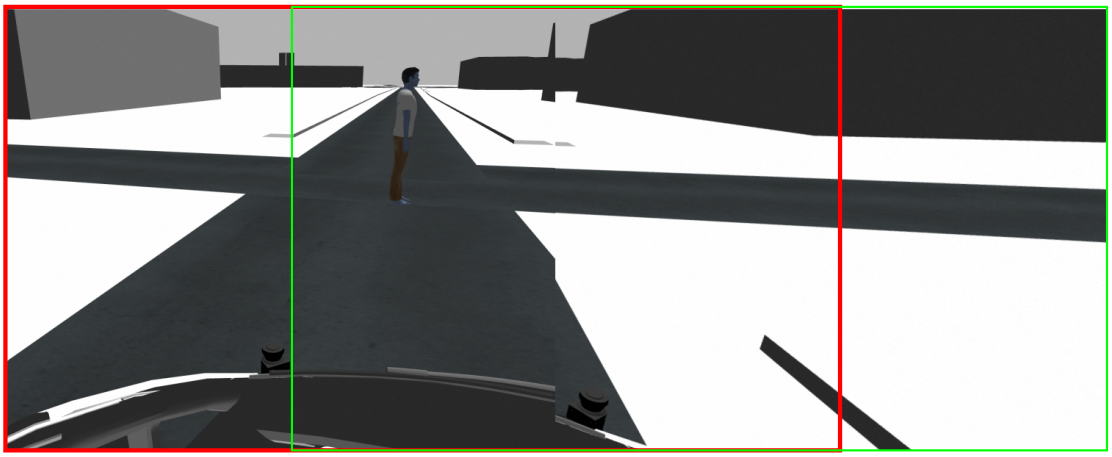
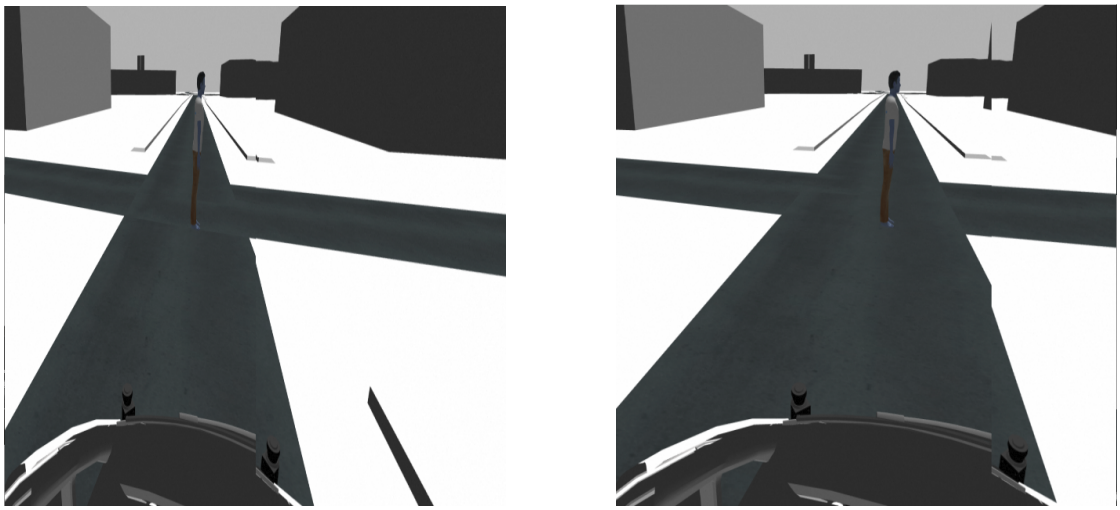


Figure 4.16: Division of the panorama image into two images represented by the red and green rectangle



(a) Panorama image 640 × 640 (b) Red rectangle panorama image 640×640

Figure 4.17: Comparison of person's display between panorama image and red rectangle panorama image when resized to 640 × 640

The initial step involves associating the points in the cluster point cloud with their respective bounding box. Algorithm 3 describes this process. It starts by finding the maximum and minimum values of the eight 3D bounding box corners in each coordinate (x, y, z). Then, it iterates over every point in the cluster point cloud and appends the points that fit within those limits.

Furthermore, the associated cluster points are projected onto the panorama image. Algorithm 4 describes this process which starts by projecting points within the left half of the panorama image onto the left camera image plane, and points within the right half onto the right camera image plane, followed by transformation into the homography plane. The projection onto both camera images is obtained through equation (4.9). In the case of the right half of the image, an additional multiplication by the homography

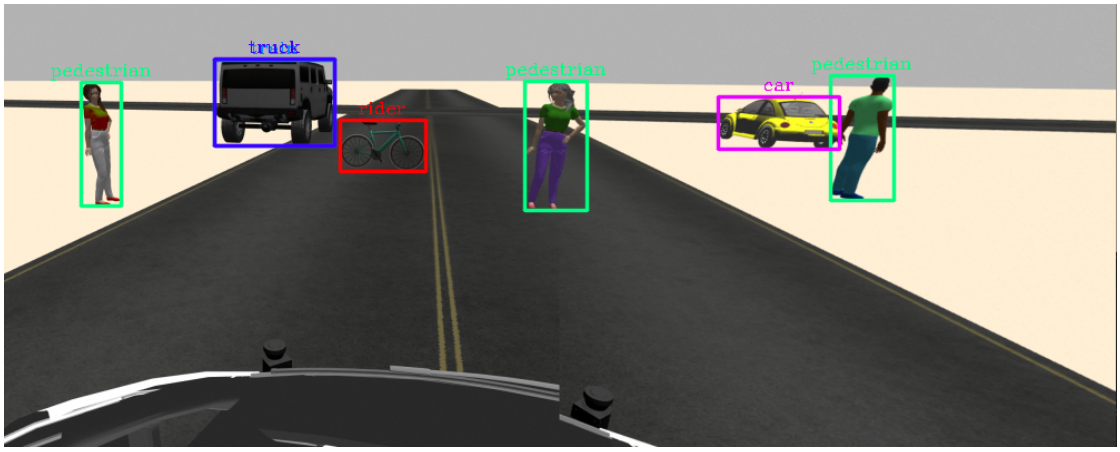


Figure 4.18: YOLOv7 obstacle detection and classification on the panorama image

matrix follows equation (4.9).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{\text{Extrinsic Matrix}} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.9}$$

Algorithm 3 Cluster-to-BoundingBox Association

Require: Cluster points $P_c = \{p_{c1}, p_{c2}, \ldots, p_{cn}\}$, 3D bounding box corner points $P_b = \{p_{b1}, p_{b2}, \ldots, p_{b8}\}$
Ensure: Cluster points within bounding box $P_{c'}$

$Min_x, Max_x = P_b[0][0]$
$Min_y, Max_y = P_b[0][1]$
$Min_z, Max_z = P_b[0][2]$
for all $corner \in P\_b$ **do**
  $x, y, z = corner$
  $Min_x = \min(Min_x, x)$
  $Max_x = \max(Max_x, x)$
  $Min_y = \min(Min_y, y)$
  $Max_y = \max(Max_y, y)$
  $Min_z = \min(Min_z, z)$
  $Max_z = \max(Max_z, z)$
end for
$P_{c'} \leftarrow$ empty list
for each point $p_c \in P_c$ **do**
  if $Min_x \leq p_c.x \leq Max_x$ **and** $Min_y \leq p_c.y \leq Max_y$ **and** $Min_z \leq p_c.z \leq Max_z$ **then**
    Append p_c to $P_{c'}$
  end if
end for
return Associated cluster points $P_{c'}$

where X, Y, and Z are the 3D point coordinates; r_{ij} are the elements of the rotation matrix; t_x, t_y, and t_z are the translation parameters; f_x and f_y are the focal lengths; c_x and c_y are the principal point offsets; u and v are the image homogeneous coordinates.

Algorithm 4 LiDAR Cluster Points Projection onto Panorama Image

Require: LiDAR points P_{lidar}, Extrinsic matrices T_{left}, T_{right}, Intrinsic matrices K_{left}, K_{right}, Homography matrix H, Panorama width $W_{panorama}$
Ensure: Projected cluster points P_c

$P_c \leftarrow$ empty list
for all $p_{lidar} \in P_{lidar}$ **do**
  $p_{left\_cam} = T_{left} \cdot p_{lidar}$ {Extrinsic transformation to left camera frame}
  $p_{right\_cam} = T_{right} \cdot p_{lidar}$ {Extrinsic transformation to right camera frame}
  $p_{left\_img} = K_{left} \cdot p_{left\_cam}$ {Intrinsic transformation to left image}
  $p_{right\_img} = K_{right} \cdot p_{right\_cam}$ {Intrinsic transformation to right image}
  $p_{right\_hom} = H \cdot p_{right\_img}$ {Homography transformation to homography plane}
  if $0 \leq p_{left\_img}.x < \frac{W_{panorama}}{2}$ **then**
    $P_c \leftarrow P_c \cup \{p_{left\_img}\}$
  end if
  if $\frac{W_{panorama}}{2} \leq p_{right\_hom}.x < W_{panorama}$ **then**
    $P_c \leftarrow P_c \cup \{p_{right\_hom}\}$
  end if
end for
return P_c

After the associated cluster points are projected onto the panorama image, a 2D bounding box is formed using the minimum and maximum pixel coordinates obtained from the projection.

The 2D bounding box formation process could be easier and faster by directly projecting the corner points of the 3D LiDAR bounding box. The problem with this method lies in the axis-aligned nature of the 3D bounding box, which may not accurately represent the object's orientation in the real world. Consequently, when projected into 2D, this alignment discrepancy can lead to a bounding box that exceeds the segmented object's boundaries. Figure 4.19 shows the difference between the 2D bounding box derived from the 3D bounding box corners (orange box) and those formed from the cluster points (green box).

The combination of the mentioned green bounding boxes with the YOLOv7 bounding boxes is achieved with the metric IoU and is described in Algorithm 5. For each LiDAR bounding box, IoU is calculated with all YOLOv7 bounding boxes. If the value exceeds 0.4, the YOLOv7 class is added to the LiDAR bounding box dictionary as a key with value 1. The key value is incremented by 1 every time the IoU threshold is exceeded. Therefore, the YOLOv7 class assigned is the one with the highest value in the LiDAR bounding box label dictionary. This approach prevents the erroneous class attribution when YOLOv7 misidentifies an object, assuming that YOLOv7 has good classification accuracy. The threshold was assigned considering the LiDAR bounding box size changes depending on the distance to the sensor. Figure 4.20 shows the mentioned combination on three different classes.

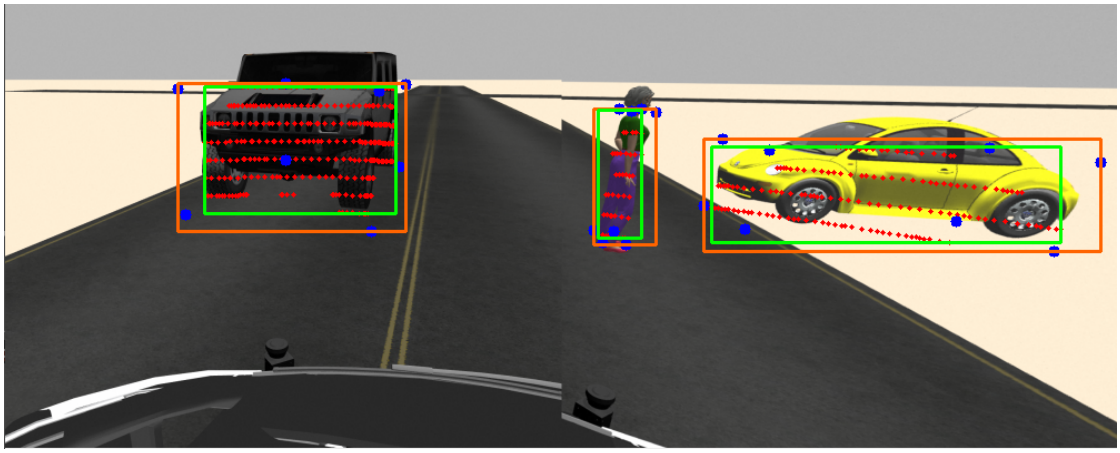


Figure 4.19: Corners derived bounding box (orange), cluster points derived bounding box (green), projected 3D bounding box corners (blue) and cluster points (red)

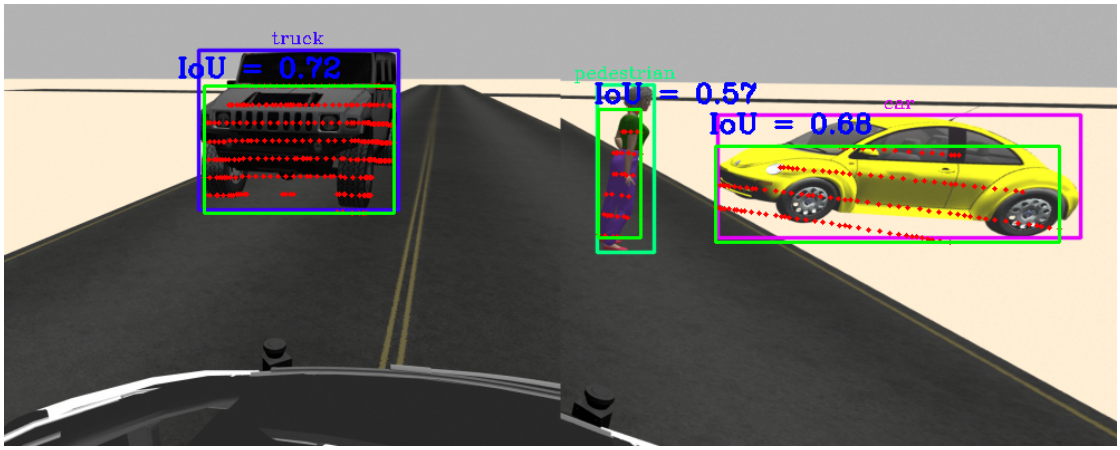


Figure 4.20: LiDAR's and YOLOv7's bounding boxes combination through IoU

4.3 Risk Maps

Risk Maps is the pivotal concept of this dissertation which is also a novel application introduced in the ATLASCAR2. It is based on the Occupancy Maps (Section 4.1) and for this reason, a deeper look into the author's [46] implementation is mandatory. They have two main similar functions that are responsible for applying the methods and equations described in Section 4.1. One function outputs the cell probabilities for the entire grid, while the other outputs the collision risk. The first function is defined as follows:

```
@njit(f64[:](int8,int8, int8, f64[:,:], f32, f32, int8), nogil=True, fastmath=
    ↪ True, cache=True)
def prob_machine_gridgen(f, originX, originY, objectList, vsx, vsy, t):
    # function implementation goes here
    return rspaceDat
```

The first two lines before the function definition (`def ...`) specify a *njit* decorator from the Numba library [59], which is used to compile Python functions into machine

Algorithm 5 IoU Calculation for Bounding Box Matching

Require: YOLO bounding box $B_y = (x_{y1}, y_{y1}, x_{y2}, y_{y2})$, Cluster bounding box $B_c = (x_{c1}, y_{c1}, x_{c2}, y_{c2})$, YOLO class C_y, Cluster bounding box label dictionary *boxes_dict*
Ensure: IoU score and frequency dictionary *boxes_dict*

$x_{\text{left}} = \max(x_{y1}, x_{c1})$ {Left top corner of the intersection area}
$y_{\text{top}} = \max(y_{y1}, y_{c1})$
$x_{\text{right}} = \min(x_{y2}, x_{c2})$ {Right bottom corner of the intersection area}
$y_{\text{bottom}} = \min(y_{y2}, y_{c2})$
if $x_{\text{left}} < x_{\text{right}}$ **and** $y_{\text{top}} < y_{\text{bottom}}$ **then**
    $I_{\text{area}} = (x_{\text{right}} - x_{\text{left}}) \times (y_{\text{bottom}} - y_{\text{top}})$
else
    $I_{\text{area}} = 0$
end if
$A_y = (x_{y2} - x_{y1}) \times (y_{y2} - y_{y1})$ {YOLO bounding box area}
$A_c = (x_{c2} - x_{c1}) \times (y_{c2} - y_{c1})$ {Cluster bounding box area}
$U_{\text{area}} = A_y + A_c - I_{\text{area}}$ {Union area}
if $U_{\text{area}} > 0$ **then**
    IoU $= \frac{I_{\text{area}}}{U_{\text{area}}}$
else
    IoU $= 0$
end if
if IoU > 0.4 **then**
    if B_c.label $\notin$ *boxes_dict* **then**
        boxes_dict[B_c.label] = {'unknown' : 0} {Initialize label entry if not present}
    end if
    if $C_y \notin$ *boxes_dict*[B_c.label] **then**
        boxes_dict[B_c.label][C_y] $= 1$ {Initialize YOLO class count if not present}
    else
        boxes_dict[B_c.label][C_y]$+ = 1$ {Increment YOLO class count if present}
    end if
end if
return IoU, *boxes_dict* {Return the IoU score and the updated dictionary}

code at runtime for execution speed. It outlines the function's arguments types and certain options to increase compilation speed.

Follows a detailed overview of the function's input and output parameters:

f (int8): 8-bit integer that represents the number of obstacles.
originX (int8): 8-bit integer that represents the grid x origin in relation to the ego vehicle.
originY (int8): 8-bit integer that represents the grid y origin in relation to the ego vehicle.
objectList (f64[:,:]): 2-dimensional Numpy [60] array of type $float64$ that has the obstacles properties such as speed, dimensions, etc.
vsx (f32): 32-bit floating-point number that represents the ego's vehicle speed in x direction.

vsy (f32): 32-bit floating-point number that represents the ego's vehicle speed in y direction.

t (int8): 8-bit integer that represents the time horizon.

rspaceDat (f64[:]): 1-dimensional Numpy array of type $float64$ that has all the grid cell's probability values.

The other function, which outputs the collision risk has the same input arguments. The collision risk is the maximum probability within the cells that are in a collision zone, as stated in Section 4.1. In order to use other metrics to evaluate the risk, a slight modification was made to the code. Instead of returning only the maximum value, all probability values within the collision zone are now returned as a 1-dimensional `Numpy` array. This information is crucial for understanding the data structure required to utilize the functions effectively.

As illustrated in Figure 4.9, the obstacle information arrives to the `risk_maps` node through the topic `/fused_detection`. This topic carries messages of type "MarkerArray" [61] which is a group of markers, each representing an obstacle. This type of message is very convenient as it allows to transport the obstacle's information and also enables the visualization of obstacle classes as text in RViz. The properties of the "Marker" message utilized are shown in the following code-snippet:

```
marker = Marker()
marker.header = lidar_box.header
marker.color.a = 0.8 # Transparency value
marker.id = lidar_box.label
marker.type = Marker.TEXT_VIEW_FACING
marker.action = Marker.ADD
marker.color.r = 0 # Black color
marker.color.g = 0
marker.color.b = 0
marker.pose.position.x = lidar_box.centroid.position.x
marker.pose.position.y = lidar_box.centroid.position.y
marker.pose.position.z = lidar_box.dimensions.z + 1 # 1 unit above the box
marker.pose.orientation.w = 1 # Align with world coordinates
marker.scale = lidar_box.dimensions
marker.text = "YOLO class"
```

Moreover, some processing is needed in order to get the markers information into a 2-dimensional Numpy array so that it can be used in the functions mentioned above. Referring to the array as the object list, its first dimension is filled with the various obstacles, while its second dimension comprises their properties. To iterate over the markers in the "MarkerArray" message, using *marker.id* is the best option because it uniquely identifies each marker. While other properties of the markers (eg.: position, scale, ...) may change, the id remains constant unless the bounding box tracking fails. As mentioned before, the *marker.id* is derived from the LiDAR bounding box label and is assigned randomly. This means that one bounding box might have an id of 20, while another might have an id of 1140. For this reason, there is a need to remap the ids to organize them systematically. This is crucial for using them as indexes for the object list, allowing the assignment of values to the object list within the same iteration loop. Algorithm 6 describes this process where the *id_to_new_id* dictionary is responsible for keeping the correspondence between the old and new ids, while the *new_markers* dictionary stores the markers with the new ids.

Algorithm 6 ID Mapping

Require: Marker messages
Initialize:
  id_to_new_id ← {}
  new_markers ← {}
  new_id_counter ← 0
for *marker* **in** *message.markers* **do**
  if *marker.id* **not in** *id_to_new_id* **then**
    id_to_new_id[*marker.id*] = *new_id_counter*
    new_id_counter = *new_id_counter* + 1
  end if
  new_markers[*id_to_new_id*[*marker.id*]] = *marker*
end for

Another feature, that could also be addressed in the data combination section (Section 4.2.3), is the assignment of an obstacle class based on the bounding box volume. While the obstacle is not in the cameras field of view, its *marker.text* is "Unknown". In this situations, the obstacle class is attributed based on the pedestrian's bounding box volume which is approximately $0.148\,\mathrm{m}^3$. Therefore, if the *marker.scale* volume is lower or equal to $0.148\,\mathrm{m}^3$, the class is "pedestrian", else the class is "vehicle". This estimation is then confirmed by the *marker.text* when the obstacle enters the cameras field of view.

Three obstacle's properties still need to be calculated and assigned to the object list: speed, acceleration, and orientation. The first paragraph of Section 4.2.1 provides a brief overview on how these properties are obtained. Algorithm 7 describes the process in greater detail, showing the mentioned calculations and also the assignment of all the obstacle's properties to the object array. A mean filter was used to smooth the orientation values by averaging the last three measurements. Testing was conducted with both three and five values for the average, indicating that the former led to slightly faster orientation adjustment, thus becoming the chosen option.

Once the functions arguments are set, the probabilities of the grid cells are retrieved for a given time horizon. This approach limits the real evaluation of risk in the scenario, because it only considers the intersections (space based approach) for a collision event to happen. This is not sufficient, because a collision scenario at a 3 s time horizon might have a low probability of collision, but at a 2 s time horizon (which naturally occurs earlier), it might indicate a clear collision with a probability near 1. In this context, the interceptions (time based approach) were also included by representing the distributions of several time horizons simultaneously, which mathematically, equals to the sum of the arrays containing the cell probabilities of each time horizon. By proposing the name "Intersection Maps" to the output of the collision risk function, the formalization of the Risk Maps and Temporal Occupancy Maps is presented in expressions (4.10), (4.11).

$$\text{Temporal Occupancy Maps} = \sum_{k=1}^{n} \text{Occupancy Maps}(t_k) \tag{4.10}$$

Algorithm 7 Assignment of Obstacle Properties to Object List

Require: Marker messages

Initialize: $objectList \leftarrow$ zeros(16, 16, dtype=np.float64)
for i **in** $new\_markers$ **do**
  $marker = new\_markers[i]$
  $current\_time = marker.header.stamp$
  if $objectList[i][12] \neq 0$ **then**
    $elapsed\_time = current\_time - objectList[i][12]$
    if $elapsed\_time > 0$ **then**
      $x = marker.pose.position.x$
      $y = marker.pose.position.y$
      $vx = (x - objectList[i][3])/elapsed\_time$
      $vy = (y - objectList[i][4])/elapsed\_time$
      $vx = vx + vxmy$ {Add ATLASCAR2's speed for absolute value}
      $vy = vy + vymy$
      $yaw = \text{atan2}(vy, vx)$ {Orientation}
      $smoothed\_yaw \leftarrow \text{mean\_filter}(yaw)$
      $ax = (vx - objectList[i][8])/elapsed\_time$
      $ay = (vy - objectList[i][9])/elapsed\_time$
      $objectList[i][12] = current\_time$
    else
      $vx = objectList[i][8]$
      $vy = objectList[i][9]$
      $ax = objectList[i][1]$
      $ay = objectList[i][2]$
      $x = objectList[i][3]$
      $y = objectList[i][4]$
      $smoothed\_yaw = objectList[i][5]$
    end if
    $objectList[i][0] = marker.id$
    $objectList[i][1] = ax$
    $objectList[i][2] = ay$
    $objectList[i][3] = x$
    $objectList[i][4] = y$
    $objectList[i][5] = smoothed\_yaw$
    $objectList[i][6] = type$ {1 for vehicle, 4 for pedestrian}
    $objectList[i][8] = vx$
    $objectList[i][9] = vy$
    $objectList[i][10] = marker.scale.x$
    $objectList[i][11] = marker.scale.y$
  else
    $objectList[i][12] = current\_time$
  end if
end for

$$\text{Risk Maps} = \sum_{k=1}^{n} \text{Intersection Maps}(t_k) \tag{4.11}$$

where t is the time horizon and $k \in \{1, \ldots, n\}$.

The maximum time horizon chosen for the Risk Maps is 3 s. This value was selected because, at low speeds, the distribution of time horizons beyond 3 seconds is not well defined due to the large uncertainty of the motion models. At high speeds, a time horizon longer than 3 seconds is not useful because of the limitations of the LiDAR sensor. As stated in Section 4.2.1 and shown in Figure 4.10, the LiDAR can recognize obstacles without loosing its track within 20 m to each side and 30 m in front of the ATLASCAR2. For example, if the ATLASCAR2 is traveling at $10\,\mathrm{m\,s^{-1}}$, the 3 s future occupancy would be at the 30 m limit. This means that for this speed, a time horizon of 3 s is not useful, as the ATLASCAR2's occupancy does not intersect any obstacle's occupancy. At higher speeds, the time horizon window reduces even further. In this dissertation, the Risk Maps are tested for speed values below $10\,\mathrm{m\,s^{-1}}$, making a 3 s time horizon a reasonable maximum value.

The minimum useful number of time horizons depends on the ATLASCAR2's speed and physical dimensions. For example, for the speed value of $6\,\mathrm{m\,s^{-1}}$, illustrated in Figure 4.21, the Temporal Occupancy Map results from the sum of the Occupancy Maps for the following time horizons: 1 s, 2 s and 3 s. There are empty zones (not covered by the risk analysis procedures) between those time horizons. Depending on the size of the obstacle's bounding box, its occupancy prediction could possibly intersect the empty zones for intermediate time horizons, leading to information loss. Algorithm 8 enables the calculation of the number of time horizons that should be utilized as well as their value. For the example given, Algorithm 8 outputs 6 time horizons spaced by 0.5 s. Therefore, the Temporal Occupancy Maps should incorporate the Occupancy Maps for 0.5 s, 1 s, 1.5 s, 2 s, 2.5 s and 3 s, resulting in complete Risk Maps.

Algorithm 8 Time Horizon Calculation

Require: $Speed$ (ATLASCAR2 speed in x), $CellResolution = 0.1$, $MaxTimeHorizon = 3$, $CarLengthInCells = 30$
Ensure: number of time horizons t_{num}, value t
1: $distance = Speed \times \left(\frac{1}{CellResolution}\right) \times MaxTimeHorizon$
2: $t_{num} = \frac{distance}{CarLengthInCells}$
3: $t_{num} = \text{round}(t_{sub})$
4: **if** $t_{num} \leq 3$ **then**
5: $\quad t_{num} = 3$
6: **end if**
7: $t = \frac{MaxTimeHorizon}{t_{num}}$
8: **return** t_{num}, t

Figure 4.22 shows the Temporal Occupancy Maps resulted from the sum of three Occupancy Maps. Figures 4.23, 4.24 show the comparison between the Temporal Occupancy Maps and the Risk Maps in four subsequent frames (scenario instances).

Finally, the collision risk within the Risk Maps is evaluated using the maximum probability value within the Risk Map cells. This metric is effective because it addresses the worst-case scenarios, ensuring that the most significant potential danger is considered

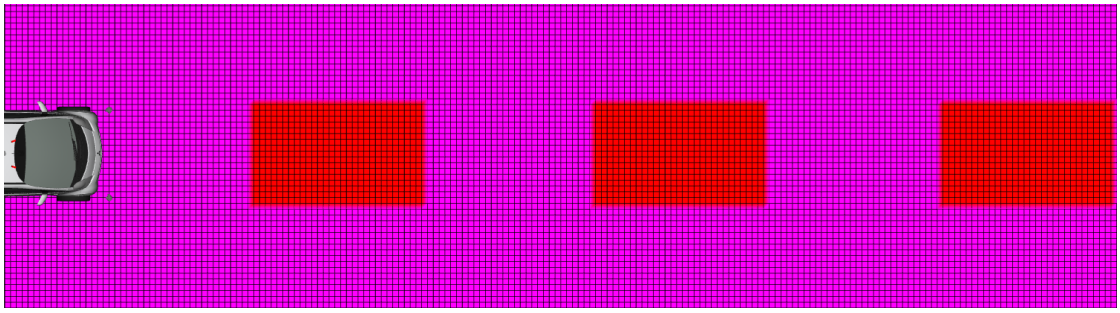
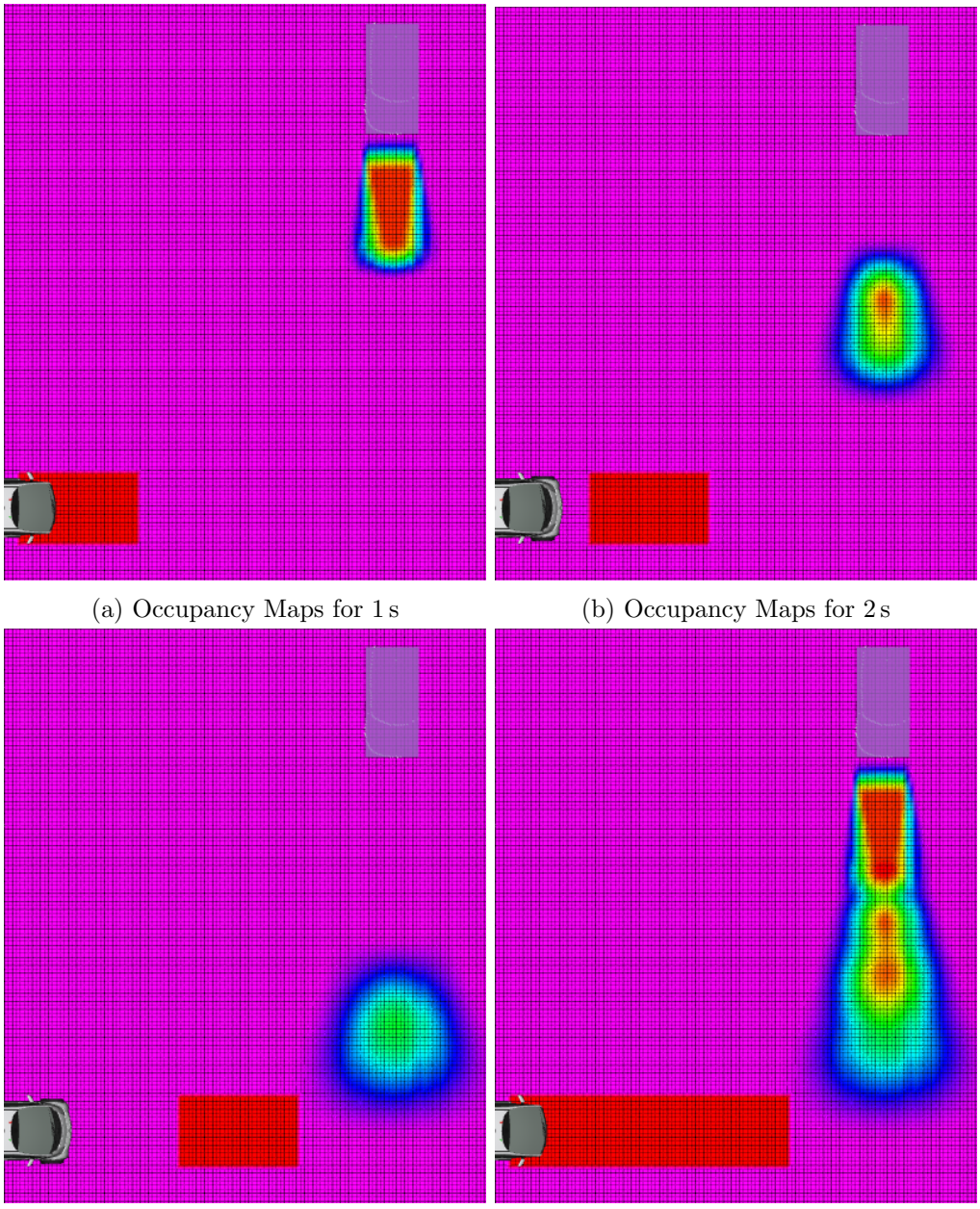


Figure 4.21: Example of the Temporal Occupancy Map for the ATLASCAR2 with speed of $6\,\mathrm{m\,s^{-1}}$

and mitigated. However, this approach has a limitation. For example, in a probability distribution like the Risk Maps, the risk might appear high if one cell has an extremely high value, while the rest have comparatively lower values. This can lead to an overemphasis on a single risk point and the overall risk might be misinterpreted. To tackle this issue, additional metrics such as the mean value, median value, and standard deviation are used to analyze the highest values in the probabilistic distribution, avoiding the overstatement of a single extreme value.



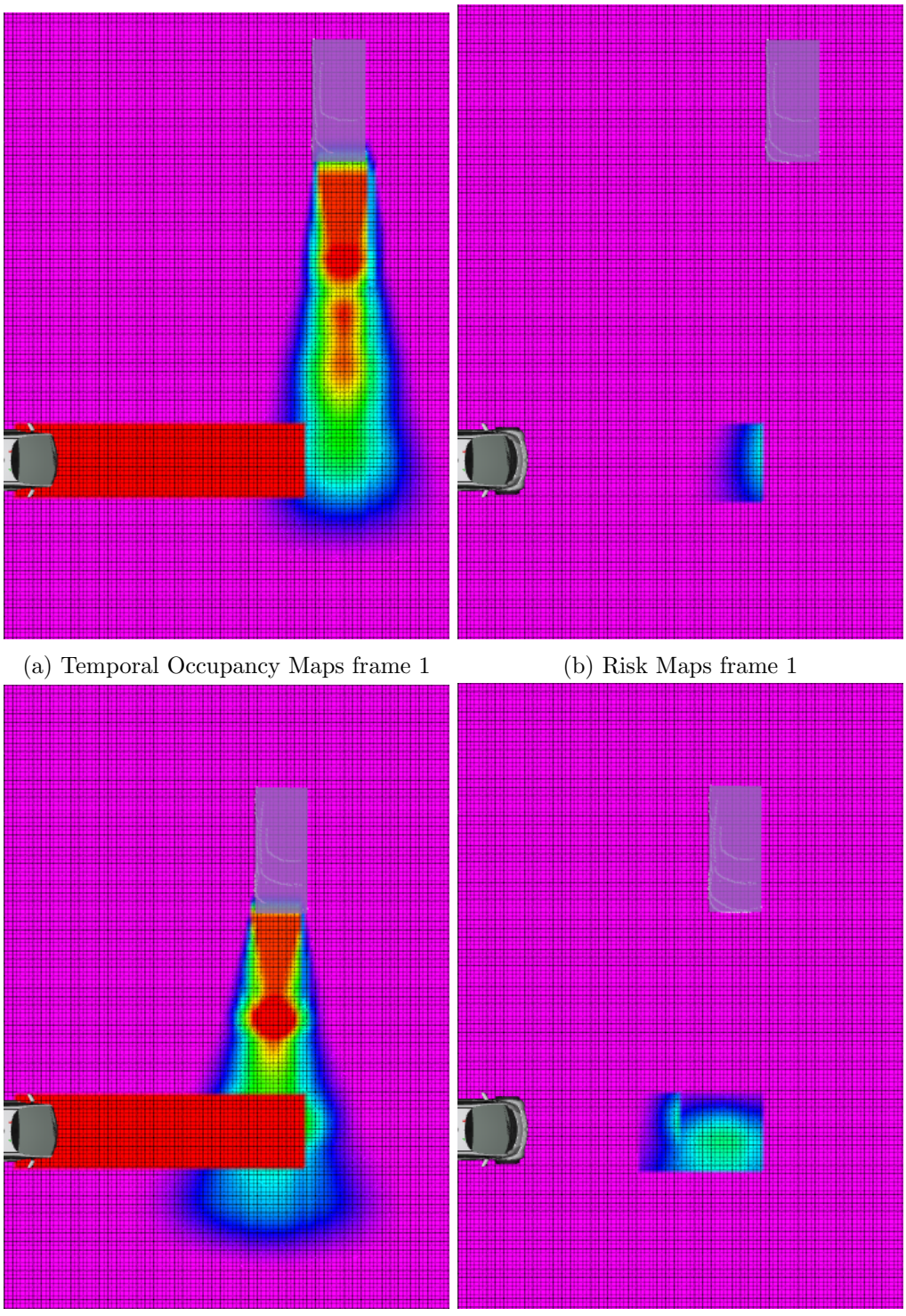
(a) Occupancy Maps for 1 s

(b) Occupancy Maps for 2 s

(c) Occupancy Maps for 3 s

(d) Temporal Occupancy Map

Figure 4.22: Temporal Occupancy Map resulting from the union of the Occupancy Maps for 1 s, 2 s and 3 s for the ATLASCAR2 and another car



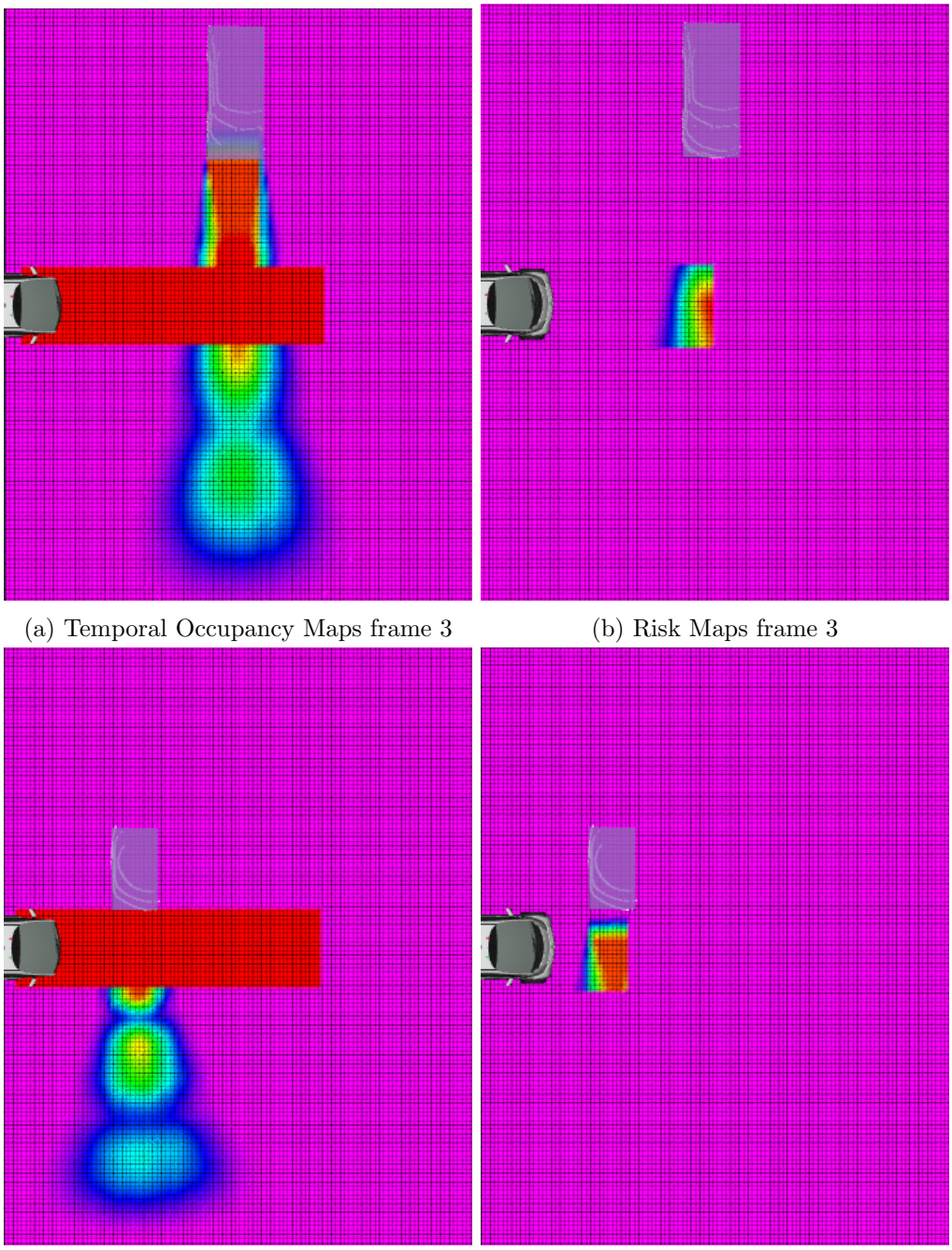
(a) Temporal Occupancy Maps frame 1

(b) Risk Maps frame 1

(c) Temporal Occupancy Maps frame 2

(d) Risk Maps frame 2

Figure 4.23: Comparison between the Temporal Occupancy Maps and the Risk Maps in the first 2 frames



(a) Temporal Occupancy Maps frame 3

(b) Risk Maps frame 3

(c) Temporal Occupancy Maps frame 4

(d) Risk Maps frame 4

Figure 4.24: Comparison between the Temporal Occupancy Maps and the Risk Maps in the last 2 frames

Chapter 5

Tests and Results

This chapter focuses on testing the proposed solution (Chapter 4) in several scenarios, illustrating the functionality and potential of the Risk Maps. Additionally, an evaluation of how the risk evolves through subsequent frames is provided. The risk is obtained from the maximum probability value within the Risk Maps cells. Additionally, the risk is validated by calculating the mean value, median value and standard deviation value for the highest twenty values of the distribution. Twenty cells cover an area of $0.2\,\mathrm{m}^2$, which provides a reasonable and representative sample of the highest risk area.

The chapter is divided in three parts: tests and results in simulation, ATLASCAR2's software configuration for real data collection, and tests and results in real data. The source code is available at GitHub.[1]

5.1 Simulation scenarios

This section provides an overview on the Risk Maps tested in several simulated scenarios in Gazebo with a graphical representations of the risk (maximum value) evolution throughout some frames (scene instances), as well as the median, mean and standard deviation of the highest twenty values.

The speed values of ATLASCAR2 used in the simulated scenarios are low (below $10\,\mathrm{m\,s^{-1}}$) for two reasons. First, lower speeds allow for more time horizons, providing a more comprehensive view of the scenes and better demonstration of the method. Second, the method is especially intended for urban scenarios, like the city of Aveiro, an urban environment where lower speeds are typical.

The stopped obstacles that appear in the scenarios are hard coded, meaning they are predefined by their role and position. The intention was to set the attribute for the stopped model to obstacles whose speed is less than $1\,\mathrm{m\,s^{-1}}$, if it is a vehicle, or less than $0.9\,\mathrm{m\,s^{-1}}$ if it is a pedestrian. However, fluctuations in speed cause the obstacle's model to switch between the stopped model and the vehicle or pedestrian model, resulting in an unwanted unstable scenario.

[1] `https://github.com/lardemua/atlascar2/tree/risk_maps`

5.1.1 Junction scenario

This scenario demonstrates a potential collision situation where the ATLASCAR2 approaches a junction with constant speed of $2\,\mathrm{m\,s^{-1}}$, while another car approaches from the left at approximately $3\,\mathrm{m\,s^{-1}}$, as shown in Figure 5.1.

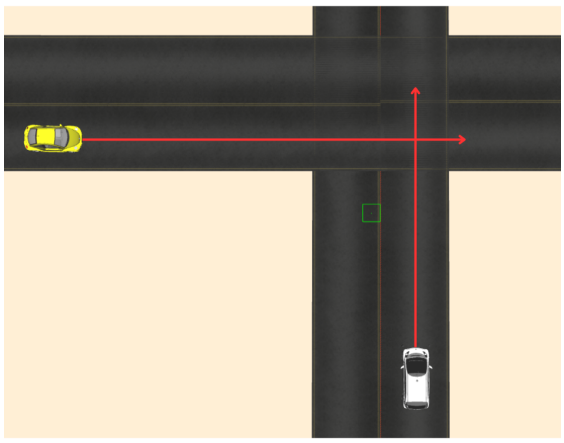


Figure 5.1: Junction scenario

Figure 5.2 shows the risk evolution from the moment that the first probabilities appear in the Risk Maps until the virtual collision. Additionally, Figure 5.3 illustrates specific frames of the Risk Maps and in Appendix A, Figure A.1 shows the Temporal Occupancy Maps of those same frames.

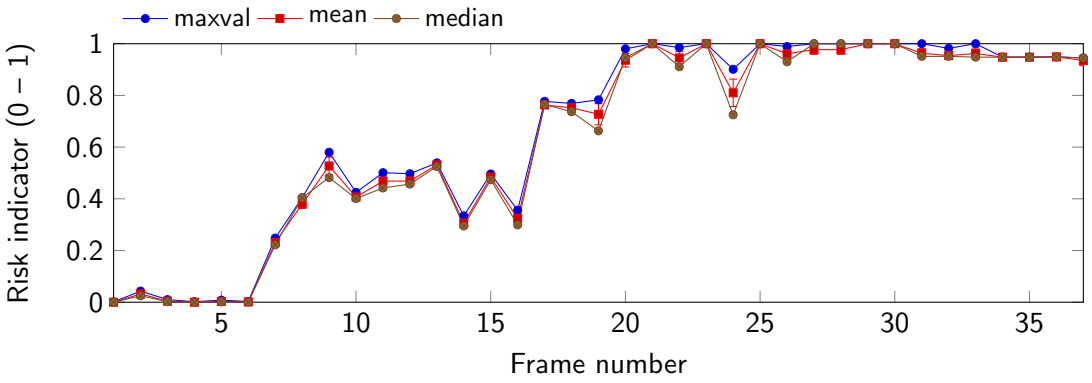
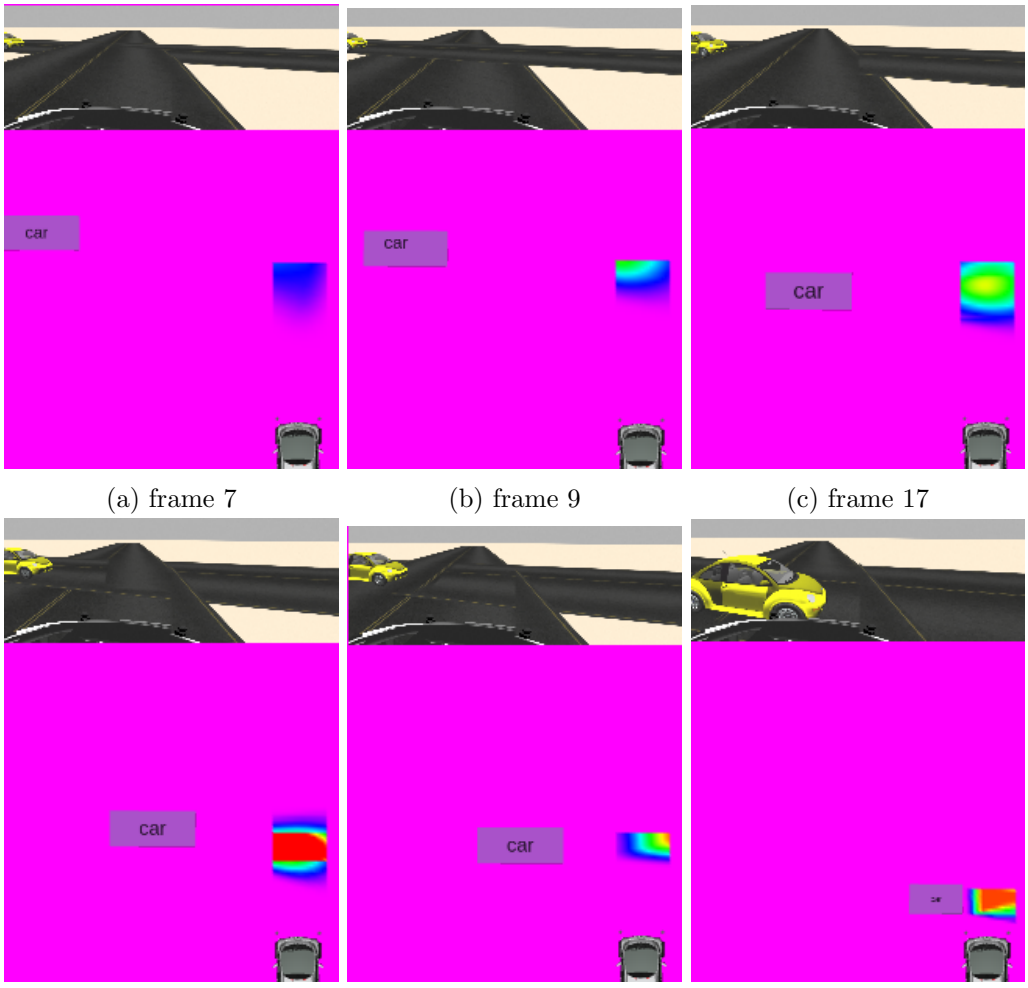


Figure 5.2: Junction scenario risk evolution. The risk indicator is displayed with 3 different items calculated in the Risk Maps: the maximum value (a single number), the mean and the median of the 20 largest values. In this case all 3 indicators agree and show a growing trend along the frames, certifying that the risk is real and will ultimately end in a collision.

Figure 5.3a shows the first relevant rise of the risk value (0.248) at frame 7 which ends on the first risk peak (0.58) at frame 9, illustrated in Figure 5.3b. The risk starts increasing again at frame 16, peaking (0.77) at frame 17 (Figure 5.3c). Figure 5.3d shows

the highest risk peak, reaching the maximum probability value (1). At frame 24 there is a slight decrease of the risk (0.901), as shown in Figure 5.3e. In the next frame, it rises again to the top, maintaining its value for a few frames. Figure 5.3f shows a frame very close to the potential collision event.

Overall, the risk increases gradually from frame 0 to frame 10; there is some fluctuations between frames 10 and 20, and from frame 20 onwards, the risk stabilizes at very high level with minor variations. These fluctuations are due to variations in the speed and acceleration changing directly the probabilistic distributions. Slight variations of the orientation can also cause the mentioned fluctuations. The median and mean values slightly drop in relation to the maximum value at frames 19 and 24, indicating some overstatement of the risk. However, the drop is not large enough and consistent to be relevant. Therefore, the Risk Map effectively transmits that this is a collision scenario.



(a) frame 7 (b) frame 9 (c) frame 17

(d) frame 21 (e) frame 24 (f) frame 34

Figure 5.3: Representative frames of the junction scenario

5.1.2 Opposite lane pass scenario

This scenario demonstrates ATLASCAR2 passing a parked car on its right while a truck approaches from the opposite lane. The truck has a constant speed of $2.5\,\mathrm{m\,s^{-1}}$ while the ATLASCAR2 is moving with speed of $2\,\mathrm{m\,s^{-1}}$. The other car is parked at approximately 2.5 m at the right of the ATLASCAR2, as illustrated in Figure 5.4.

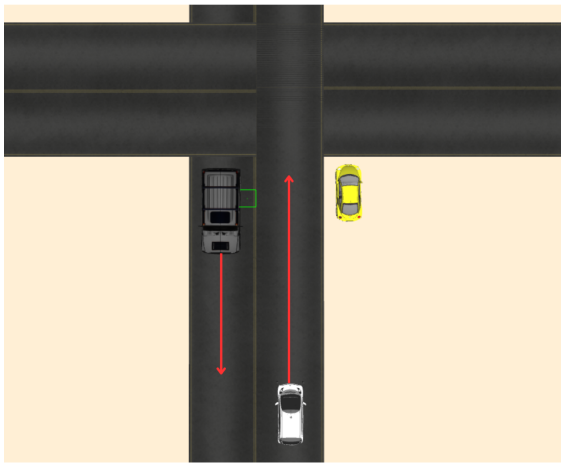


Figure 5.4: Opposite lane pass scenario

Figure 5.5 shows the risk evolution of this scenario where A represents the truck and B the parked car. Additionally, Figure 5.6 illustrate specific frames of the Risk Maps and, in Appendix A, Figure A.2 show the Temporal Occupancy Maps of the same frames.

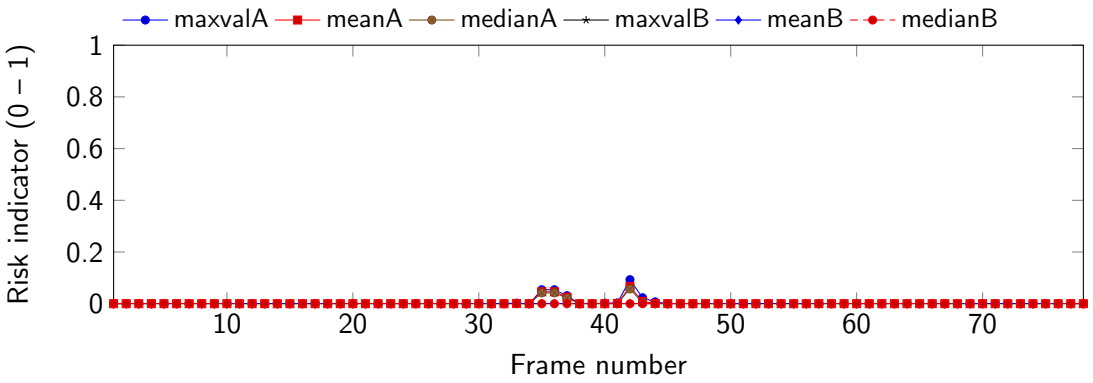
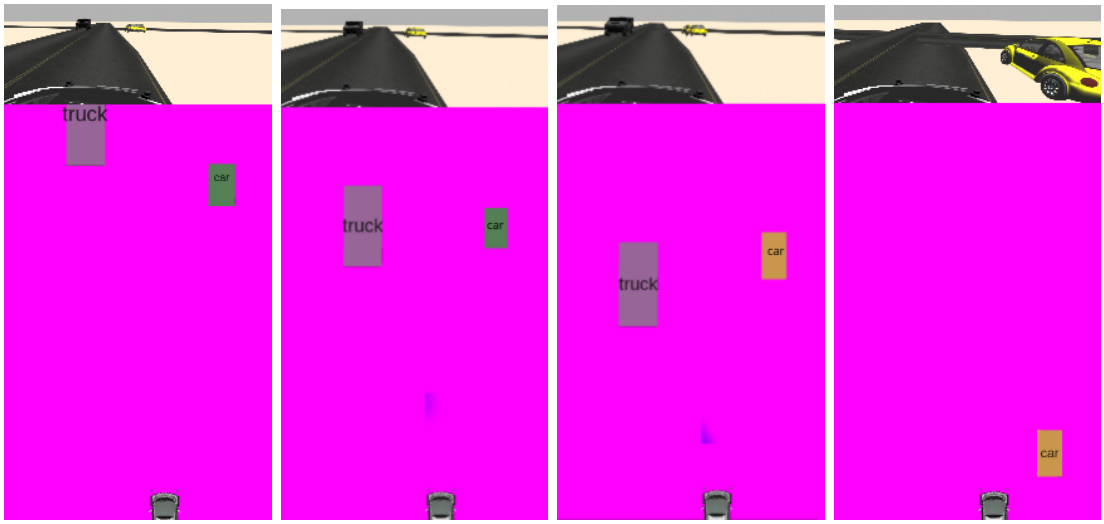


Figure 5.5: Opposite lane pass scenario risk evolution

Overall, the risk is minimum, reaching its peak of 0.093 at frame 42. For the truck, the risk starts to be noticeable at frame 35 where blue color cells appear in the intersection area, as illustrated in Figure 5.6b. Figure 5.6c shows the risk peak where the blue color is slightly more intense. As for the parked car, the risk is always 0 through the whole scenario because the ATLASCAR2 future occupancies never intersect the car's. The Risk Maps effectively convey the minimum risk in this scenario which is expected because the

vehicle are in different lanes meaning that there is space for them to pass each other safely and generally the movement is very predictable as they follow predictable paths.



(a) frame 23 (b) frame 35 (c) frame 42 (d) frame 78

Figure 5.6: Four representative frames of the opposite lane pass scenario

5.1.3 Lane collision scenario

This scenario demonstrates the ATLASCAR2 moving towards a junction with a stopped car at the entrance. Additionally, a pedestrian is walking in the opposite direction of the ATLASCAR2 passing approximately 4 m to its right. ATLASCAR2 is moving at $4\,\mathrm{m\,s^{-1}}$ and the pedestrian at $0.9\,\mathrm{m\,s^{-1}}$, as shown in Figure 5.7.

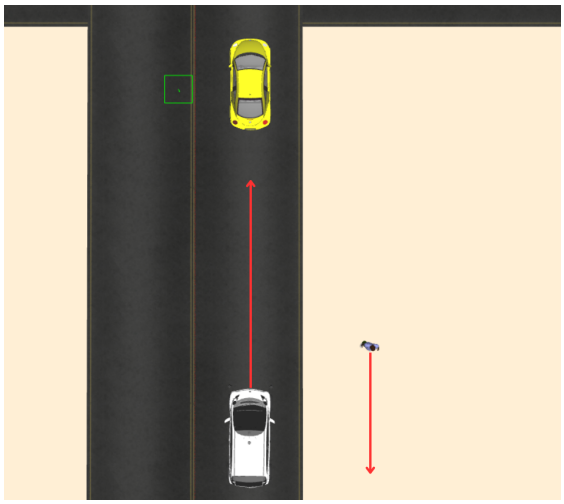


Figure 5.7: Lane collision scenario

Figure 5.8 shows the risk evolution of this scenario where A represents the pedestrian and B the stopped car. Additionally, Figure 5.9 illustrates specific frames of the Risk

Maps and in Appendix A, Figure A.3 shows the Temporal Occupancy Maps of these same frames.

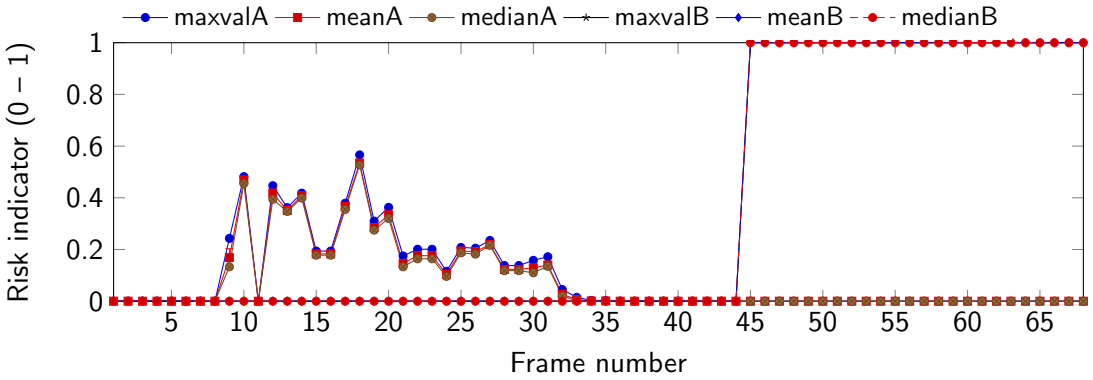


Figure 5.8: Lane collision scenario risk evolution

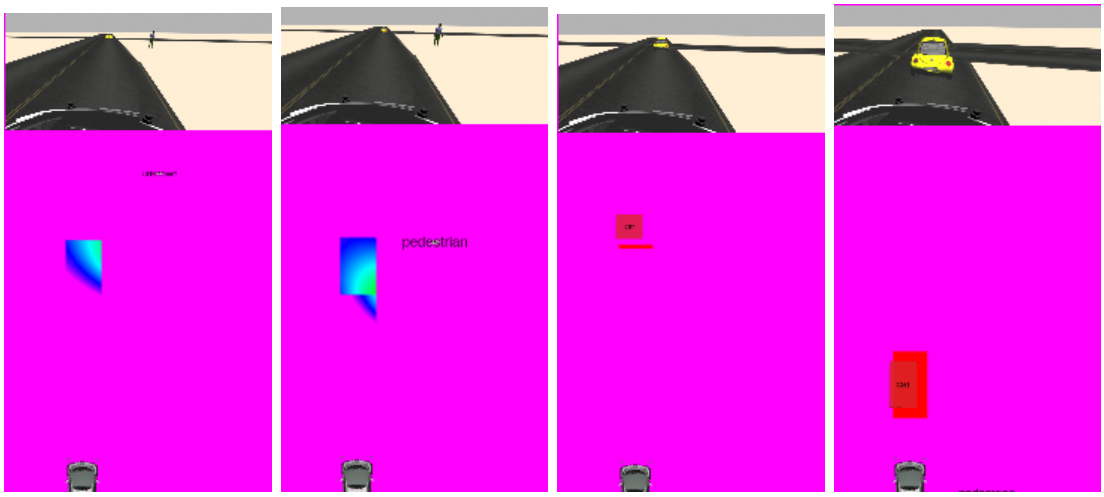
The risk reaches its first peak (0.482) at frame 10, decreasing abruptly (0) at frame 11. This fluctuation is due to the variations in the speed and acceleration values which make the distribution smaller, thereby not intersecting the ATLASCAR2's future occupancy, as illustrated in Figures A.3b, A.3c. After that, the risk rises again reaching its second peak (0.566) at frame 18, decreasing gradually in the following frames. At frame 45 the risk rises to the maximum value (1), as the ATLASCAR2's future occupancy begins to intersect the stopped vehicle occupancy as shown in Figure 5.9c. The risk is maintained at the maximum value until the end of the scenario (Figure 5.9d). Overall, the mean and median values are very close to the maximum, validating the latter. Perhaps the Risk Maps overstate the collision risk when passing by the pedestrian, as he is 4 m away and the risk reaches a value of 0.566. On the other hand, pedestrians are highly unpredictable, meaning that drivers should be careful when passing by them. Therefore, the Risk Maps would be more realistic indicating risk with lower probability values in this specific situation. As for the stopped vehicle, it effectively transmits the risk associated.

5.2 Crosswalk approach scenario

This scenario demonstrates the ATLASCAR2 approaching a crosswalk with a pedestrian passing through. ATLASCAR2 is moving at $6\,\mathrm{m\,s^{-1}}$ and the pedestrian is walking at $1\,\mathrm{m\,s^{-1}}$. It is a close to collision scenario, as the ATLASCAR2 passes very close to the pedestrian, as illustrated in Figure 5.10.

Figure 5.11 shows the risk evolution of this scenario. Moreover, Figure 5.12 illustrates specific frames of the Risk Maps and in Appendix A, Figure A.4 shows the Temporal Occupancy Maps of these same frames.

Until frame 10, the risk is null meaning that car and the pedestrian future occupancies have not yet intercepted. At frame 11, the risk peaks (1) to the maximum value and drops slightly in the next frame as the distribution gets bigger (Figure A.4c) due to variations in speed and acceleration. After that, the risk rises again to the maximum value at frame 13 and stays there till the end of the scenario. In the last frames there is a slight drop of the mean and medium values indicating that the highest twenty values



(a) frame 10 (b) frame 18 (c) frame 45 (d) frame 68

Figure 5.9: Representative frames of the lane collision scenario

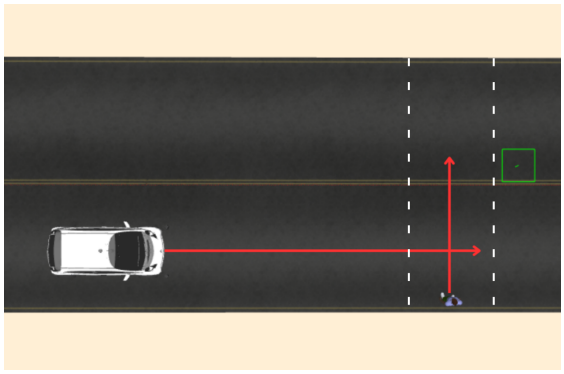


Figure 5.10: Crosswalk approach scenario

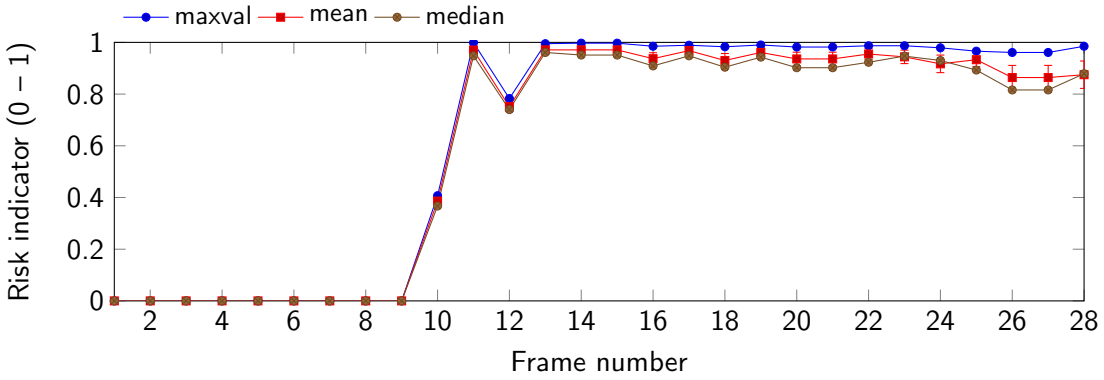
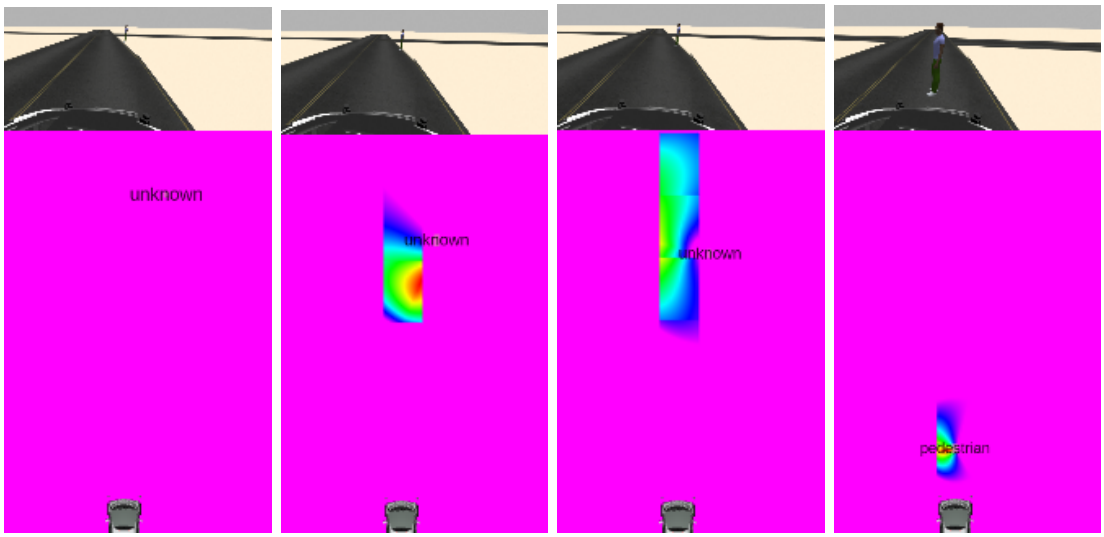


Figure 5.11: Crosswalk approach scenario risk evolution

are not exactly the maximum. Nonetheless, the drop is not big enough to be relevant, meaning that it still transmits an high risk situation. Even though there is no collision,

ATLASCAR2 passes very close to the pedestrian, indicating a high risk. The Risk Maps effectively convey this risk in such situations.



(a) frame 7 (b) frame 11 (c) frame 12 (d) frame 28

Figure 5.12: Four representative frames of the crosswalk approach scenario

5.3 Software configuration for real data collection

This section details the configurations utilized for real world data collection in the perception field (Section 4.2) of the proposed solution. Additionally, it addresses the crucial procedure of calibration, which is paramount in real world scenarios. The following subsections describe the solutions adopted.

5.3.1 Calibration

The calibration process was divided into two parts: intrinsic calibration of the cameras and extrinsic calibration of the cameras + LiDAR.

Intrinsic calibration was achieved utilizing the ROS `camera_calibrator` package [62]. It provides a graphical interface where the user get immediate feedback of what the camera is seeing and if the detection of the chessboard's corners is being successful as shown in Figure 5.13. The calibration process involves showing the chessboard in different positions and orientations until the application gathers enough information and the calibrate button is highlighted. From here, the user pushes the calibration button, and the results are displayed in the terminal. There is also an option to save the calibrated intrinsic parameters in YAML file, along with the image collections taken and used during the calibration.

This procedure was conducted for both cameras using a 9×7 chessboard with 105.4 mm square size, available at LAR. Figure 5.14 shows four collections of the Intrinsic calibration process.

Extrinsic calibration was achieved utilizing ATOM [15] described in Section 2.4. Following the pipeline presented in Figure 2.8, the first three steps consisted of recording

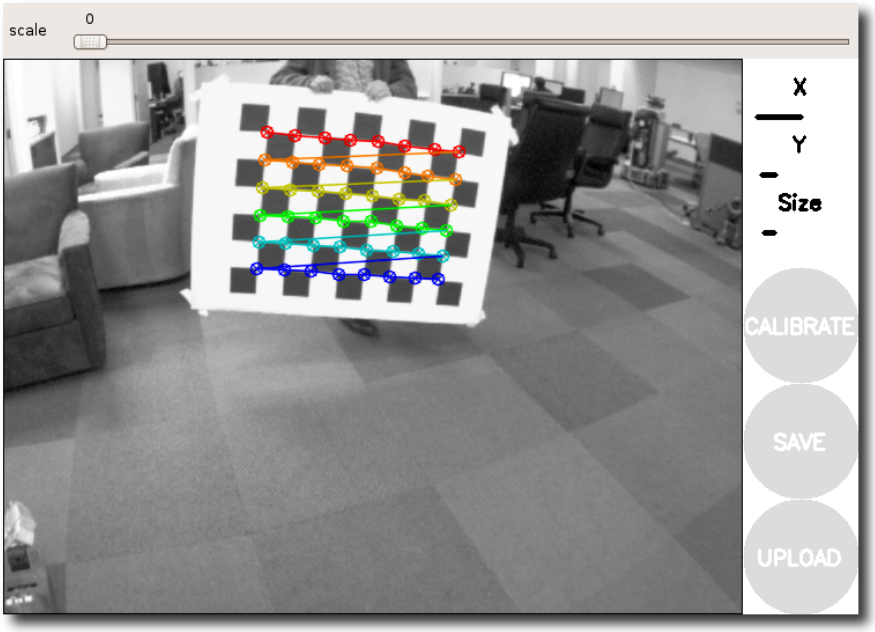


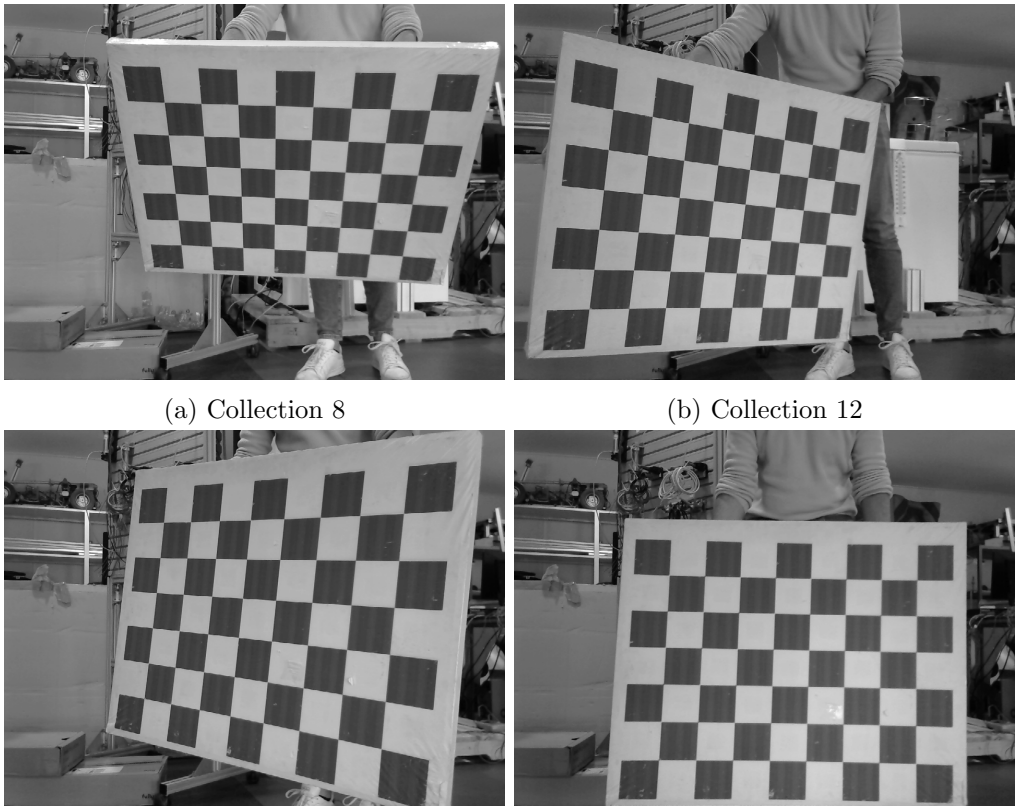
Figure 5.13: ROS camera calibration interface showing the chessboard's corner detection [62]

a ROS bag, configuring the ATLASCAR2's XACRO file and the calibration YAML file. The bag file was recorded using a charuco in various positions and orientations, ensuring visibility for the three sensors. The charuco pattern properties are present in Table 5.1. As for the XACRO file, it contains the properties of each component in the car, namely the sensors, the wheels, the joints and links between them. An estimation of the transformations between frames is also defined in this file. Figure 5.15 shows the proposed transformation tree for the sensors, where `base_footprint` is the world frame placed in position (0,0,0), and `base_link` is an intermediate frame positioned slightly above `base_footprint` at the height of the wheel radius. The calibration optimizes the transformations between the `base_link` and the subsequent frames in the tree. Lastly, the configuration YAML file is where the calibration process is set up, specifying which sensors are to be calibrated, the world frame, and the pattern properties.

Table 5.1: Charuco properties

Properties	
Dictionary	5×5
Border size	$(40, 30)$ mm
Dimension	$(11, 8)$
Size	60 mm
Inner size	45 mm

Moving to the data collection and labeling step, the user saves collections (snapshots of data) from the bag file when the pattern is detected by the sensors. For this, ATOM provides graphical feedback in RViz to indicate whether the pattern is being detected or



(a) Collection 8 (b) Collection 12

(c) Collection 22 (d) Collection 39

Figure 5.14: Four collections illustrative of the Intrinsic calibration process

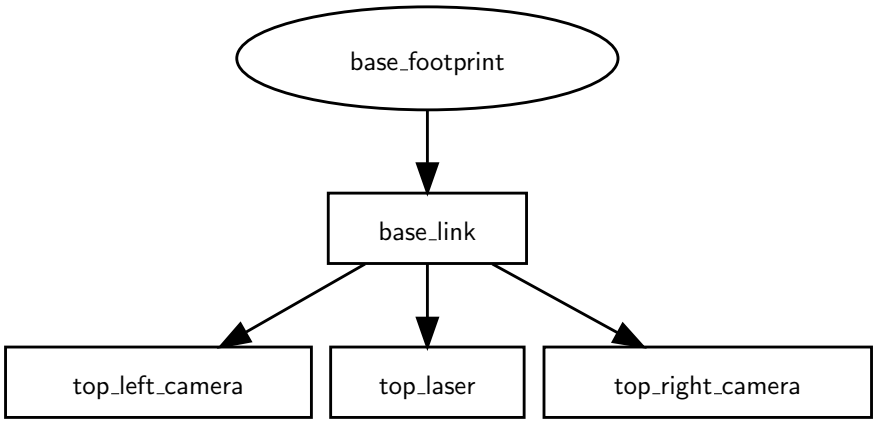
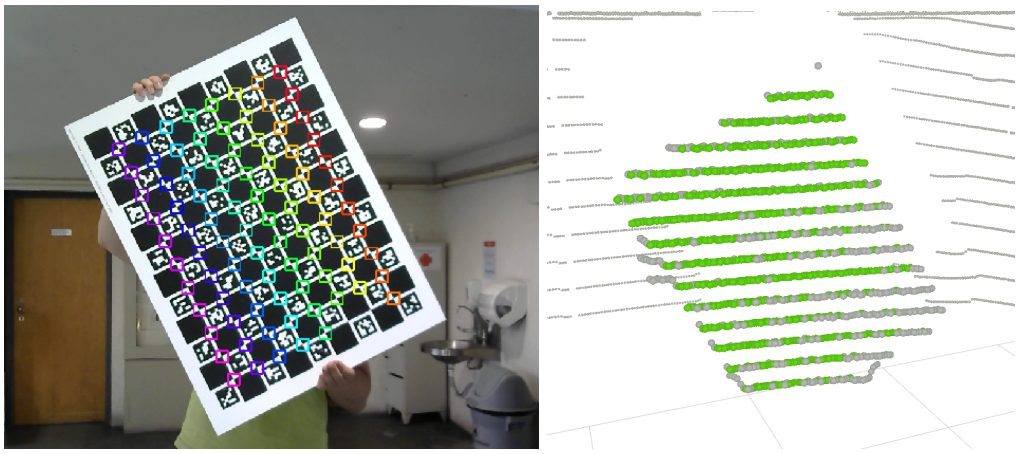


Figure 5.15: Transformation tree of the ATLASCAR2's sensors

not. The camera labeling (annotations regarding detections) occurs automatically while the LiDAR needs the user to position an interactive marker close to pattern location in the point cloud. Figure 5.16 shows the results of the camera and LiDAR labeling of the same frame. Once the collections are saved, it is time to correct the LiDAR labeling. As shown in Figure 5.16b, there are some points of the pattern that are not registered (grey

color). To correct this, ATOM provides an RViz tool that allows the user to manually register the points that are missing and set the boarder points of the pattern area (black color). The result is illustrated in Figure 5.17.



(a) RGB camera automatic labeling (b) LiDAR semi-automatic labeling

Figure 5.16: ATOM camera and LiDAR labeling result

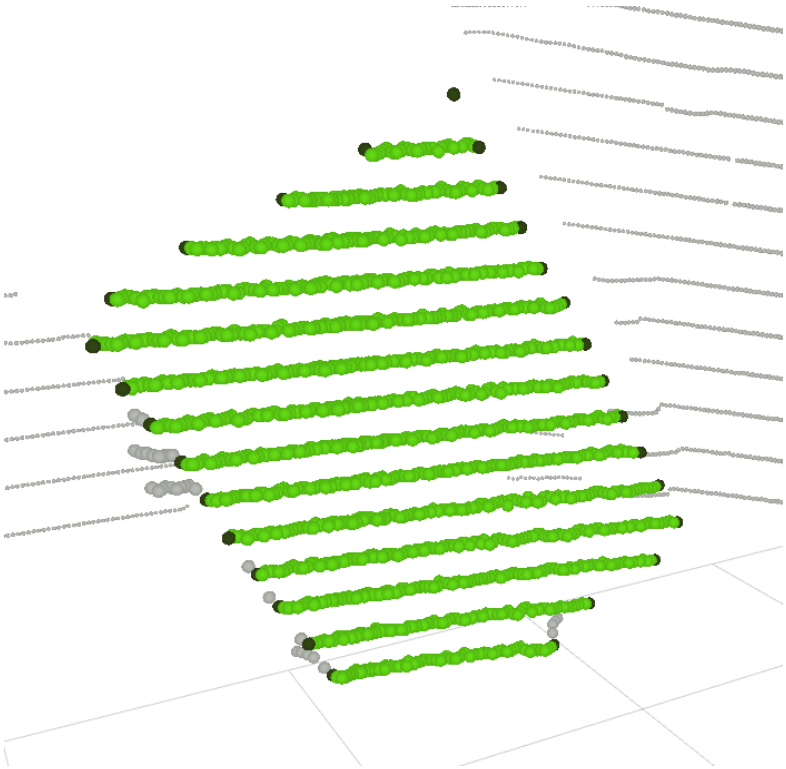


Figure 5.17: LiDAR semi-automatic labeling corrected

Once the collections are corrected, the next step is to begin the calibration. During this process, ATOM conducts an optimization module that changes the position of the sensors in order to minimize their reprojection error. Figure 5.18 shows the calibration

results. The average reprojection error is displayed at the bottom in blue, indicating sub-pixel accuracy for both cameras and around 1 cm for the LiDAR, which is considered good. Table 5.2 shows the initial estimated and the calibrated position of the sensor frames with relation to the `base_link` frame (Figure 5.15).

Collection	top_left_camera [px]	top_lidar [m]	top_right_camera [px]
000	0.1070	0.0102	---
002	---	0.0074	0.3025
003	---	0.0098	0.4828
004	---	0.0191	1.2309
005	---	0.0098	0.2996
006	---	0.0125	0.3744
007	---	0.0100	0.5829
008	---	0.0095	0.4093
009	---	0.0245	0.4575
010	---	0.0150	0.4494
011	---	0.0262	0.3673
012	---	0.0251	0.4627
013	---	0.0262	0.4884
014	0.0918	0.0103	---
015	0.1380	0.0100	---
016	0.1866	0.0096	---
017	0.3482	0.0162	---
018	0.1029	0.0088	---
019	0.1809	0.0103	---
020	0.1134	0.0089	---
021	0.1457	0.0114	---
022	0.1732	0.0132	---
023	0.1604	0.0110	---
024	0.1560	0.0096	---
Averages	0.1587	0.0135	0.4923

Figure 5.18: ATOM calibration results

Table 5.2: Initial and calibrated positions of the three sensors frames with relation to the `base_link` frame (Figure 5.15)

	Initial estimation			Calibrated		
	Right_cam	Left_cam	LiDAR	Right_cam	Left_cam	LiDAR
x (m)	0.5	0.6	0.5	0.511	0.647	0.457
y (m)	−0.27	0.35	0	−0.041	0.341	−0.041
z (m)	1.345	1.295	1.515	1.346	1.294	1.474
roll (rad)	0	0	0	−0.007	−0.013	−0.011
pitch (rad)	0	0	0	−0.029	−0.011	−0.013
yaw (rad)	−0.35	0.45	0	−0.157	0.451	0.018

5.3.2 Stitch

The stitching method described in Section 4.2.2 had to be adjusted (real-world camera setup differs from the simulation camera setup) to the real world setup. The procedure to obtain the homography matrix was the same. A colleague from LAR helped on this assignment, by position himself on the limit zone of the two camera images.

Figure 5.19 shows the left (Point Grey) and the right (USB) camera images used to find the homography matrix and Figure 5.20 shows the resulting panorama image. The homography obtained and used with the real world setup is presented in equation (5.1).

(a) Left camera image (b) Right camera image

Figure 5.19: Panorama stitching setup to find the homography matrix

Figure 5.20: Real panorama image result

$$\text{Homography matrix} = \begin{bmatrix} 1.294 & 0.176 & 778.058 \\ 0.071 & 1.206 & -57.029 \\ 0.000 & 0.000 & 1.000 \end{bmatrix} \tag{5.1}$$

5.3.3 LiDAR object detection and clustering

Similar to the simulation scenarios, the Lidar obstacle detector application [8] was used to detect and segment obstacles into 3D bounding boxes. Section 4.2.1 details this application and the parameters (Figure 4.10) used for the simulation scenarios. While the parameters remained consistent across all simulation scenarios, they had to be customized for each real-world scenario. This is due to the high level of noise that varies in each scenario. The main goal was to ensure successful tracking of the obstacle bounding boxes.

5.3.4 Sensor synchronization

Similarly to the simulated scenarios, it is necessary to synchronize both camera messages for the stitching algorithm. Additionally, in the `data_combiner` node (Figure 4.9), synchronization is required when combining the panorama image and the YOLO detections with the LiDAR data. For this purpose, ROS message filters [58] were used, specifically the `ApproximateTimeSynchronizer()` function. This function queues the messages that are chosen for synchronization for a specified queue time. Unlike the `TimeSynchronizer()` function used in simulation, `ApproximateTimeSynchronizer()` includes an additional parameter, the `slop`. This parameter compensates for the time interval between messages. For example, setting the `slop` to 0.1 enables two messages with a 0.1 s or smaller time differential to be considered synchronized. This synchronization method is particularly beneficial for real world data as it provides the necessary tolerance to handle the time variations.

5.3.5 ATLASCAR2 velocity

Initially, the ATLASCAR2's velocity was meant to be obtained from the encoder pulses of the odometry solution described in Section 3.1.1.3. However, the resulting velocity values from this solution were very unstable, probably due to some hardware issue. For this reason, a new approach was needed. In Pombinho's work [36], another approach for the ATLASCAR2 velocity was tested, which involved using the actual car's velocity provided by the vehicle's CAN bus. This is the same velocity value displayed by the car's speedometer behind the steering wheel. This velocity value is very stable and realistic. However, its resolution of $1\,\mathrm{km\,h^{-1}}$ ($0.278\,\mathrm{m\,s^{-1}}$) was insufficient for Pombinho's work, which required higher resolution for accurate odometry calculations. In the case of the Risk Maps, stability in velocity values is paramount. The resolution of $0.278\,\mathrm{m\,s^{-1}}$ is considered sufficient, hence this solution was adopted. Figure 5.21 shows the stability difference between the velocity values of the encoder and the speedometer for the time duration of 10 s.

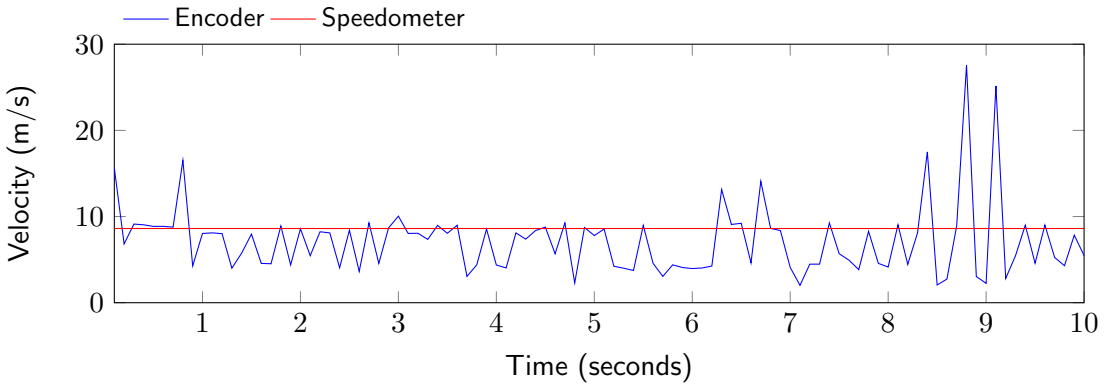


Figure 5.21: Comparison between the odometry encoder and the car speedometer velocity values for a duration of 10 s.

5.4 Real world scenarios

This section provides an overview on the Risk Maps tested in real world scenarios. The data was collected in Aveiro, near the university. Figure 5.22 shows the path traveled with ATLASCAR2 while collecting data, as well as the locations of the real world scenarios, numbered from 1 to 5. Similarly to the simulated scenarios, there are graphical representations of the risk (maximum value) evolution through some frames, as well as the median, mean and standard deviation of the highest twenty values.

Figure 5.22: Real data collection path in Aveiro and scenario locations numbered from 1 to 5

5.4.1 Roundabout approach scenario

This scenario is indicated by number 1 in Figure 5.22. It demonstrates the ATLASCAR2 approaching a roundabout (Figure 5.24) at approximately $4.7\,\mathrm{m\,s^{-1}}$. The velocity decreases the closest it gets to the roundabout, as there is another car inside, making the ATLASCAR2 to stop eventually. Figure 5.23 illustrates the evolution of the risk in

this scenario. Additionally, Figure 5.24 illustrates specific frames of the Risk Maps and in Appendix A, Figure A.5 shows the correspondent Temporal Occupancy Maps of these same frames.

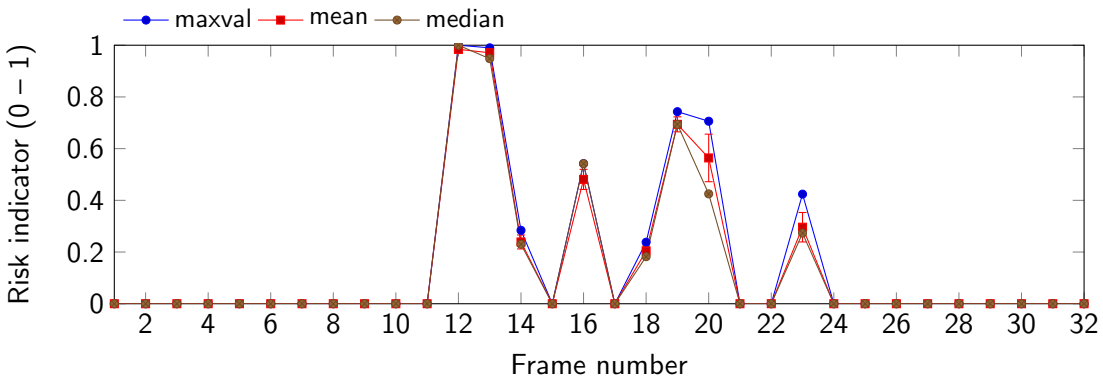


Figure 5.23: Roundabout approach scenario risk evolution

The scene starts with the ATLASCAR2 approaching the roundabout with the LiDAR first detecting the other car at frame 6. After that, the risk reaches its peak (0.994) at frame 12 (Figure 5.24b) indicating the first intersection of both occupancies. Then, the risk decreases abruptly at frame 14 and being null at frame 15 (Figure 5.24c). Right in the next frame, the risk increases to the value of 0.542 at frame 16. The same behaviour appears from frame 16 to frame 21. This is due to fluctuations in the velocity and acceleration, making the distributions to keep switching their size (Figures A.5b, A.5c, A.5d).

Overall, the risk starts at the maximum value and decreases, as the car inside the roundabout moves out and the ATLASCAR2 slows down. Despite the fluctuations, the Risk Maps effectively transmit that there is a potential collision scenario, if precautions are not taken, and demonstrates it in several frames.

5.4.2 Opposite lane pass scenario

Two scenarios were obtained for the opposite lane passing: one with wide lanes and the other with tight lanes.

5.4.2.1 Wide lanes

This scenario is indicated by number 2 in Figure 5.22. It demonstrates the ATLASCAR2 passing by three cars coming in the opposite lane. The ATLASCAR2's speed ranges from 6.3 to 8.6 $\mathrm{m\,s^{-1}}$ by the end of the scenario. Figure 5.25 shows that there is no risk in this scenario, as the maximum value remains 0 till the last frame. Figure 5.26 shows three frames of the Risk Maps and in Appendix A, Figure A.6 shows the correspondent Temporal Occupancy Maps. These frames capture the moment each vehicle is about to pass by the ATLASCAR2.

(a) frame 6 (b) frame 12 (c) frame 15

(d) frame 16 (e) frame 23 (f) frame 24

Figure 5.24: Four representative frames of the roundabout approach scenario

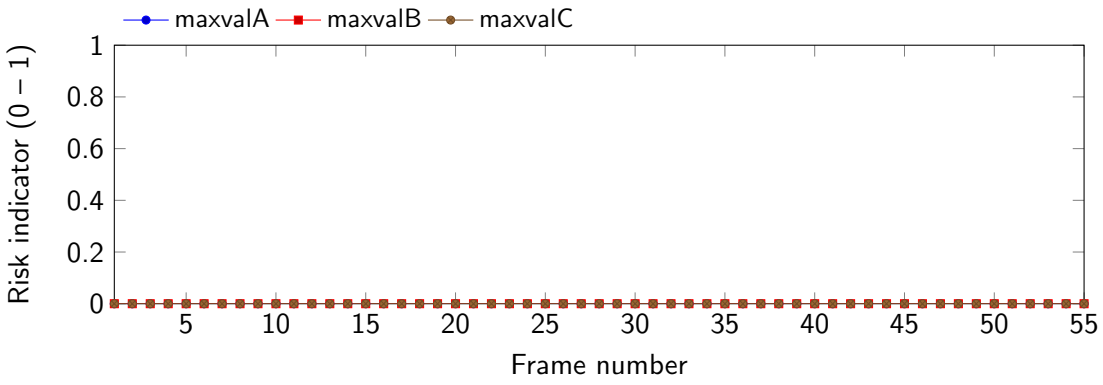
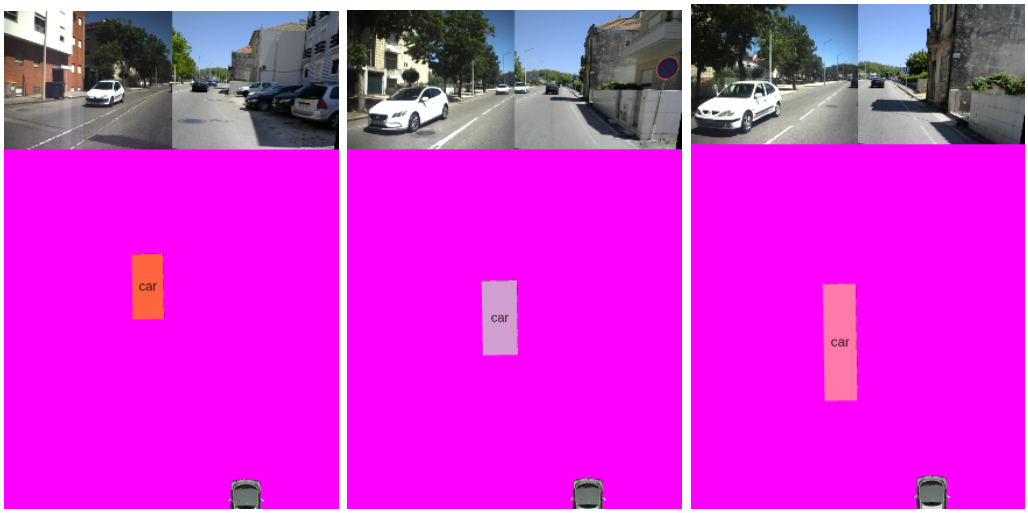


Figure 5.25: Wide opposite lane pass scenario risk evolution



(a) frame 12 (b) frame 38 (c) frame 50

Figure 5.26: Three representative frames of the wide opposite lane pass scenario

5.4.2.2 Tight lanes

This scenario is indicated by number 3 in Figure 5.22. It involves the ATLASCAR2 passing by a truck coming in the opposite lane. The ATLASCAR2's speed ranges from 5.5 to 6.6 $\mathrm{m\,s^{-1}}$ by the end of the scenario. Figure 5.27 shows the risk evolution. Figure 5.28 shows three representative frames of the Risk Maps and in Appendix A, Figure A.7 shows the correspondent Temporal Occupancy Maps.

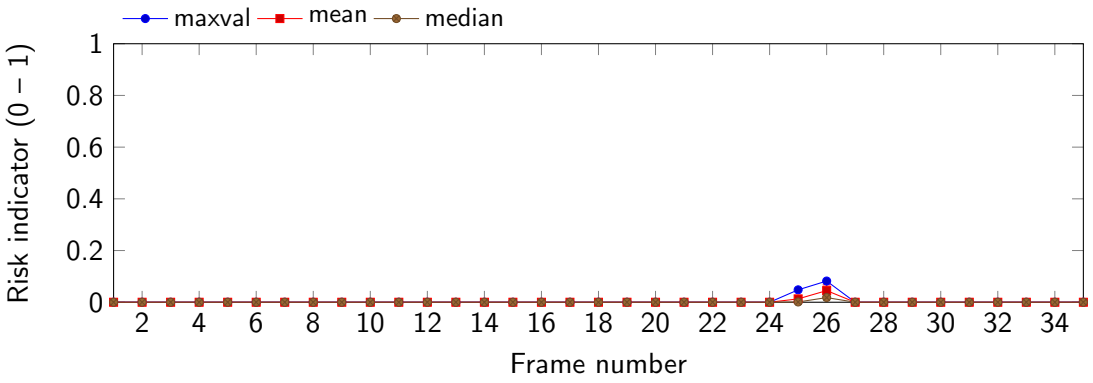
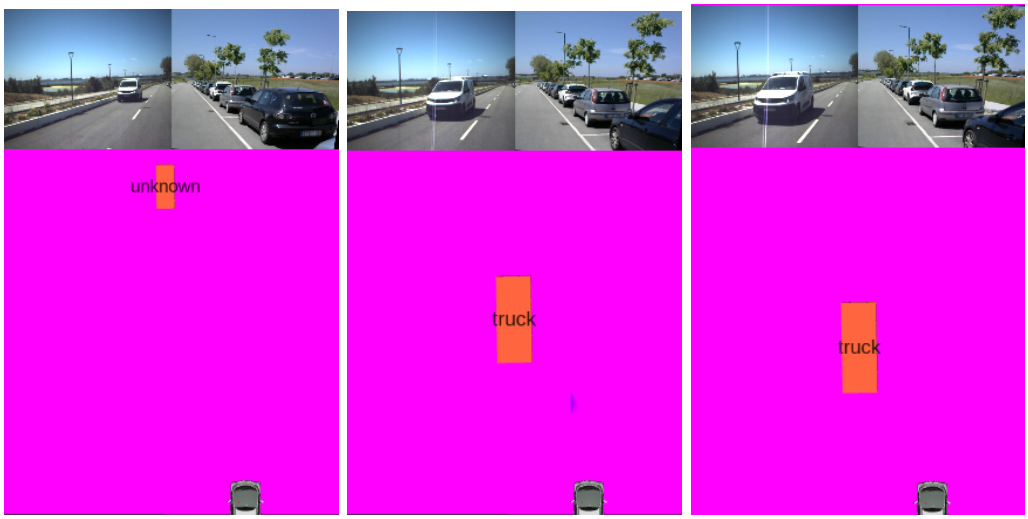


Figure 5.27: Tight opposite lane pass scenario risk evolution

The risk is practically null throughout the whole scenario, except for the transition from frame 24 to frame 27 (Figure 5.28c). This is due to a fluctuation in the orientation as illustrated in Figure A.7b. The maximum value reached was at frame 26 with value 0.082 (Figure 5.28b). Frame 21 represents the moment the truck is detected by the LiDAR. Despite the orientation fluctuation, the Risk Maps consistently show that there is no collision in this scenario.



(a) frame 21 (b) frame 26 (c) frame 27

Figure 5.28: Three representative frames of the tight opposite lane pass scenario

5.4.3 Junction approach scenario

This scenario is indicated by number 4 in Figure 5.22. It demonstrates the ATLASCAR2 approaching a junction with a car in front and some stopped obstacles detected on the sidewalk on the left. When close to the junction, another car comes from the left making the ATLASCAR2 to slow down and wait for him to pass. The ATLASCAR2's velocity ranges from 5.8 to 1.3 $\mathrm{m\,s^{-1}}$ by the end of the scenario. Figure 5.29 shows the risk evolution where A represents the car in the front and B the car coming from the left. Figure 5.30 shows the representative frames of the Risk Maps and in Appendix A, Figure A.8 shows the correspondent Temporal Occupancy Maps. In Figure 5.30 the front car marker text is not well visible but it is the light orange rectangle. This can be confirmed with the images present in Figure A.8.

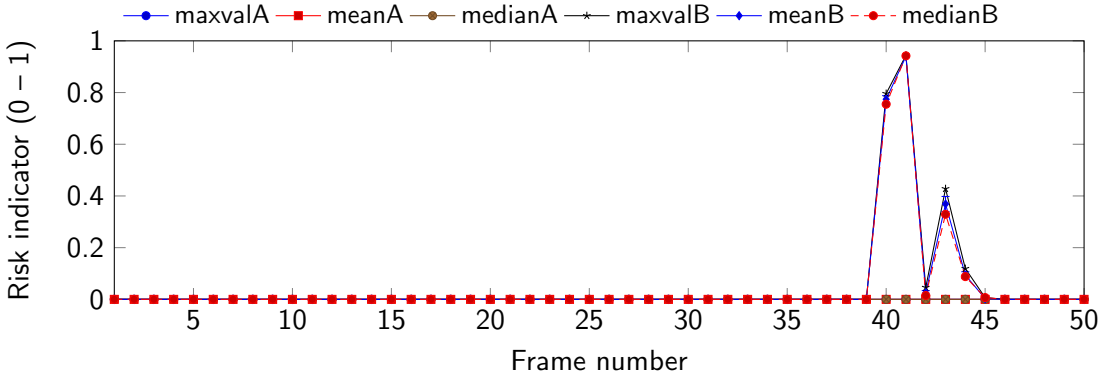


Figure 5.29: Junction approach scenario risk evolution

The car in front does not contribute to the risk which is expected, since it is moving away from the ATLASCAR2. At frame 40 the risk rises abruptly to the value of 0.796

reaching its peak in the next frame (0.944). A slight variation in the speed and acceleration made the risk drop at frame 42 (Figure A.8d). After that, the risk raises again to the value of 0.428 at frame 43, decreasing to 0 in the following frames.

The Risk Maps convey a potential collision with the car approaching from the left, which would happen in reality if ATLASCAR2 was not slowing down. Additionally, the risk decreases as the car passes by the ATLASCAR2 which also correspond to the reality. Therefore, the Risk Maps transmit effectively the risk in this scenario.

(a) frame 11 (b) frame 16 (c) frame 40

(d) frame 42 (e) frame 43 (f) frame 48

Figure 5.30: Representative frames of the junction approach scenario

5.4.4 Crosswalk approach scenario

This scenario is indicated by number 5 in Figure 5.22. It demonstrates the ATLASCAR2 approaching a crosswalk where a pedestrian is crossing through. When the pedestrian is reaching the end of the crosswalk, a motorcyclist passes by the ATLAS-

CAR2 in the opposite lane. The ATLASCAR2's velocity ranges from 6 to 2.6 $\mathrm{m\,s^{-1}}$ by the end of the scenario. Figure 5.31 shows the risk evolution where A is the pedestrian and B is the motorcyclist. Figure 5.32 shows the representative frames of the Risk Maps and in Appendix A, Figure A.9 shows the correspondent Temporal Occupancy Maps.

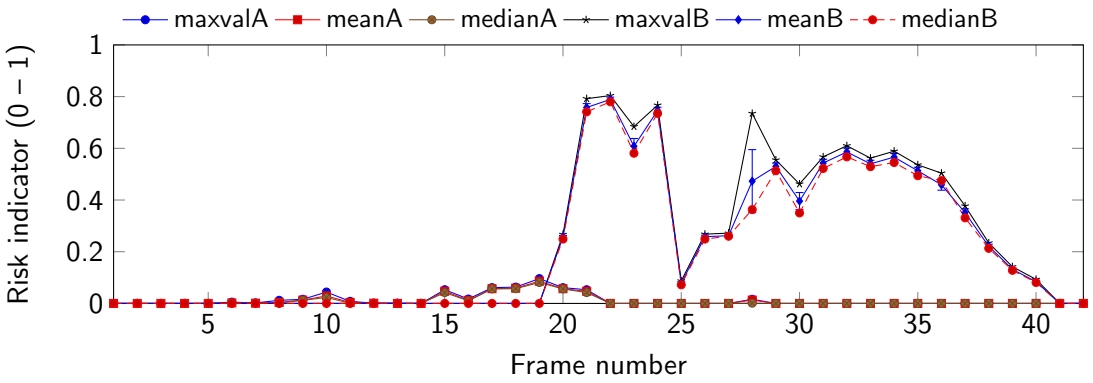


Figure 5.31: Crosswalk approach scenario risk evolution

The risk with the pedestrian is relatively low throughout the scenario, reaching its maximum value (0.096) at frame 19. As for the motorcyclist, the risk is overall high, starting to increase at frame 21 and reaching its peak (0.804) at frame 22. At frame 25 the risk decreases abruptly due to a fluctuation in the orientation, as illustrated in Figure A.9d. After that, the risk increases again gradually until frame 32, decreasing till 0 in the following frames. At frame 28, the risk is not supported by the mean and median of the twenty highest values, indicating that the maximum value is not accurate. Frames 3, 17 represent the moments that the pedestrian and the motorcyclist are detected, respectively.

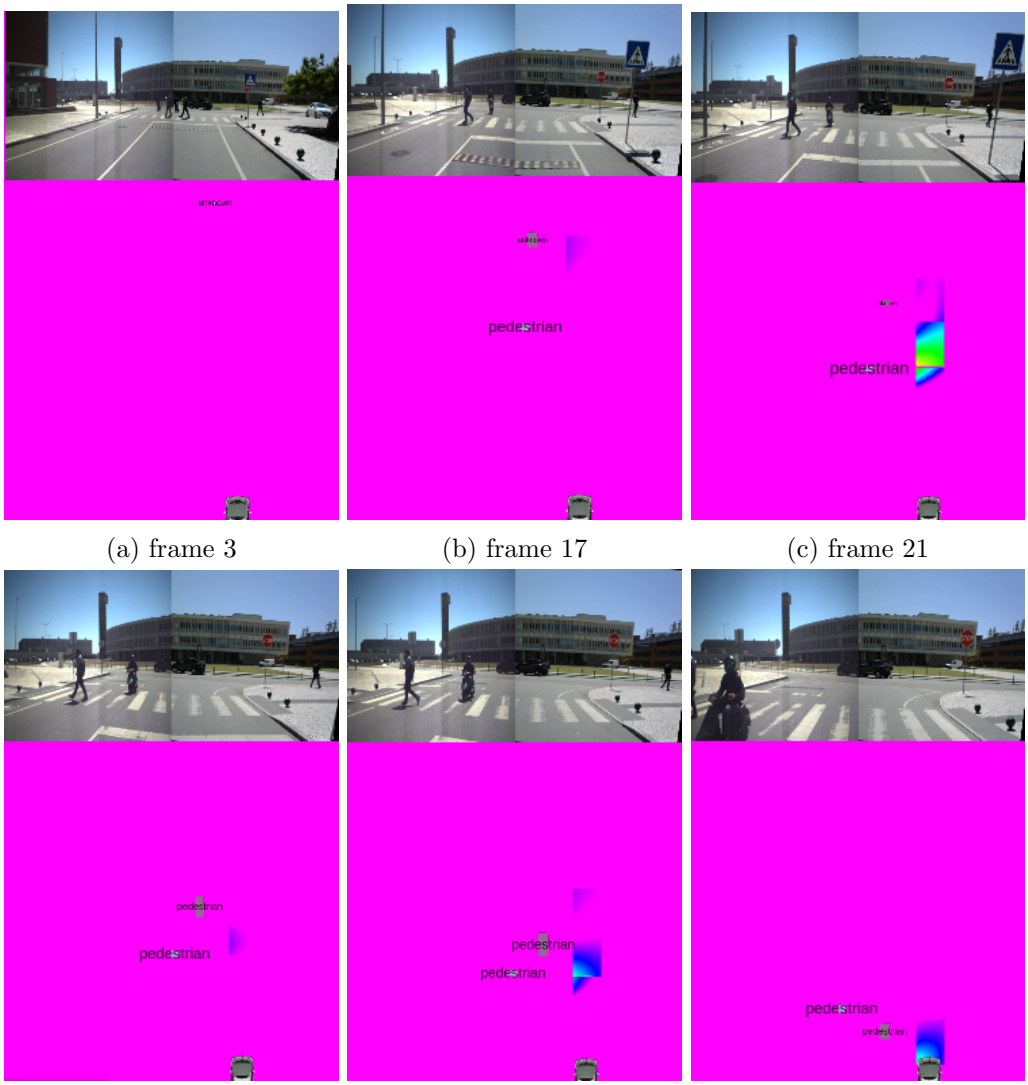
The Risk Maps transmit the collision risk between the ATLASCAR2 and the pedestrian relatively well, but fails to represent the risk of the motorcyclist. The main problem lies in the YOLO classification, as it identifies the motorcyclist as a pedestrian, thereby enabling the pedestrian model of the Risk Maps instead of the vehicle model. Therefore, the Risk Maps fail to transmit effectively the risk in this scenario.

5.5 Summary

The Risk Maps were tested on both simulated and real world scenarios. In both cases, common urban situations were analyzed to determine the collision risk between the ATLASCAR2 and other road agents such as pedestrians and other vehicles. The evaluation of the risk was achieved by representing the maximum value within the Risk Maps cells and supporting it with the median and mean values of the twenty highest values. Additionally, the standard deviation was also utilized to demonstrate how close the dataset is to the mean value.

The scenarios analyzed in simulation world are as follows:

Junction collision: ATLASCAR2 collides with other car coming from the left at junction. The risk increases gradually as the car gets closer to the ATLASCAR2, eventually reaching the maximum probability value (1).



(a) frame 3 (b) frame 17 (c) frame 21

(d) frame 25 (e) frame 28 (f) frame 36

Figure 5.32: Representative frames of the crosswalk approach scenario

Opposite lane pass: ATLASCAR2 passes by a truck coming in the opposite lane and, right after, by a car parked at his right. The risk is practically null, reached a peak value of 0.093 at specific frame when passing by the truck.

Lane collision: ATLASCAR2 approaches a junction with a car stopped at the entrance. Additionally, ATLASCAR2 passes by a pedestrian walking in the opposite direction at its right. The risk between ATLASCAR2 and the pedestrian was slightly exaggerated as it reached a peak value of 0.566 with the pedestrian being at 4 m distance away from the ATLASCAR2. As for the stopped car, the risk reached the peak value of 1, as it should.

Crosswalk approach: ATLASCAR2 approaches a crosswalk with a pedestrian crossing through. ATLASCAR2 ends up passing really close to the pedestrian. The risk increases abruptly to the maximum value (1), remaining there till the end.

The scenarios analyzed in real world are as follows:

Roundabout approach: ATLASCAR2 approaches a roundabout with another car inside it. The risk begins high at the top, decreasing as the car passes by the ATLASCAR2.

Wide lanes opposite pass: ATLASCAR2 passes by three vehicles coming in the opposite lane. The risk is null throughout the whole scenario.

Tight lanes opposite pass: ATLASCAR2 passes by a truck coming in the opposite lane. The risk is practically null except for two frames where the orientation of truck fluctuated.

Junction approach: ATLASCAR2 approaches a junction with a car on his front. When close to the junction, another car appears from the left. There is no risk associated with the vehicle in front as it moves away from the ATLASCAR2. As for the car coming from the left, the risk starts very high reaching the value of 0.944 and then decreases as it passes by the ATLASCAR2.

Crosswalk approach: ATLASCAR2 approaches a crosswalk with a pedestrian crossing through. When the pedestrian is reaching the end of the crosswalk, a motorcyclist appears in the opposite lane. The risk related to the pedestrian is very low throughout the scenario reaching a peak value of 0.096. As for the motorcyclist, the risk is very high which does not relate with the reality. Risk Maps failed to represent this scenario.

Another aspect covered in this chapter is the software adaptations made for real world data collection. This includes calibration, panorama stitching, LiDAR object detection and segmentation, sensor synchronization and the achievement of the ATLASCAR2's speed.

Intentionally blank page.

Chapter 6

Conclusions and future work

This chapter summarizes the work developed within this dissertation and presents the conclusions taken from it. Additionally, it addresses future work ideas that can be implemented using this work as a starting point.

6.1 Conclusions

This dissertation started with the aim of developing Risk Maps utilizing artificial vision (cameras) and LiDAR. The purpose of these maps is to assist drivers, human or machine, in detecting road obstacles and provide information about the potential risk of colliding into them.

To achieve this goal, the dissertation starts by exploring the state of the art of relevant concepts and methodologies. This involves the following fields: object detection and classification, panorama stitching, LiDAR segmentation or clustering, LiDAR and camera data combination, and collision risk assessment.

The development of the proposed solution began with exploring the Occupancy Maps collision risk assessment method, identifying the necessary inputs and the outputs it provides. It proved to be a promising solution, as it provides a map with future occupancy predictions of road agents over a given time horizon based on their instantaneous properties namely, speed, acceleration, size, orientation and nature. From there, the idea was to use the available hardware on the ATLASCAR2, along with the newly installed 3D LiDAR (other goal of this dissertation), to obtain these properties in a simulation environment. This approach ensures proximity to real world conditions which is the destination.

During this phase, perception solutions were found for the cameras and the LiDAR. As there were two cameras present on the ATLASCAR2, it made sense to fully utilize them maximizing the global field of view by stitching each camera image together into a panorama. The stitching process involves finding the homography matrix that ensures a smooth transition from the left camera image to the right camera image. The homography was calculated once and then used for the entire recording. This approach ensured speed and stability; however, it only works well for the specific distance at which the homography was calculated. For perfect panoramas, the homography would need to be calculated for every frame using a more sophisticated algorithm.

Moreover, to detect and classify obstacles in the panorama image, YOLOv7 was utilized within Ribeiro's application [11]. This approach ensures good detections, but

requires time for inference and incurs significant computational cost; therefore, there is room for optimization, perhaps utilizing a lighter model. For the LiDAR segmentation, the LiDAR obstacle detector was utilized [8]. This application has great capabilities namely, 3D bounding box formation, real time parameter definition and bounding box tracking. The only downside is that it could take some time to find the best parameters. An optimization procedure could enhance the application experience by choosing the best parameters for every LiDAR scan.

Associating the LiDAR boxes with the YOLO detections was another challenge. The method utilized relies on the IoU between the projected LiDAR boxes and the YOLO bounding boxes. The only downside of this method is that the distance of an object from the LiDAR sensor affects the size of its LiDAR box, making IoU matching more difficult.

To better suit the dissertation goals, the Occupancy Maps method was modified by incorporating a model for stopped obstacles and by redefining the ego vehicle's trajectory to represent its predicted future occupancy. The intersection between the ATLASCAR2's future occupancy with other agent's future occupancy over a given time horizon creates the Intersection Maps. By summing the Occupancy and Intersection Maps across an ideal number of time horizons, Temporal Occupancy and Risk Maps are generated. The Risk Maps, which are the proposed solution, also provide the collision risk corresponding to the maximum probability value of its cells. Moreover, to validate the maximum value metric, the mean, the standard deviation and the median values are calculated for the highest twenty values.

During Tests and results chapter (Chapter 5), the Risk Maps were analyzed in various scenarios within simulated and real world environments. Overall, the Risk Maps provided an accurate representation of the collision risk, with exceptions in two scenarios: the lane collision scenario in simulation and the crosswalk approach in the real world. In the former, the Risk Maps overstate the risk of a pedestrian passing by ATLASCAR2, and in the latter, they overstate the risk of a motorcyclist passing by ATLASCAR2. The pedestrian model is slightly over conservative and can be refined. The metrics utilized to support the maximum value allowed for a better understanding of the high risk probabilities and avoid overemphasizing a single extreme value. Additionally, the fluctuations caused by the variations in the speed, acceleration and orientation derive mainly from the fact that the LiDAR bounding box size changes as there are more or less detected points. This leads to the change of position of the centroid. Furthermore, the Risk Maps are short for the real world scenarios because the obstacles are detected by the LiDAR when they are relatively close to the ATLASCAR2 not giving enough time for a more detailed Risk Maps.

Overall, the stitching method is not perfect but succeeded in providing a richer, wider image. The YOLOv7 model is somewhat slow during inference, which was not an issue given the camera's low frame rate (10 fps). Additionally, YOLOv7 accurately detected all obstacles in the scenarios except for the crosswalk approach, where it misclassified the motorcyclist as a pedestrian. The LiDAR segmentation could be optimized but still succeeded in segmenting the LiDAR's point cloud with scenario specific parameters. The LiDAR-camera data combination also successfully associated the obstacle's position and size with its nature. Moreover, the collision risk successfully transmitted probability of collision scenarios. Focusing on the maximum probability value, allow decision-makers

to prioritize actions that prevent or mitigate the highest risk identified, which is crucial in road environments.

To conclude, the goals of this thesis were fulfilled, leading to the enhancement of the ATLASCAR2's perception by integrating 3D LiDAR to its infrastructure and developing a new concept, the Risk Maps. The Risk Maps are an upgrade of the existing Occupancy Maps [21], with the main differences as follows:

- Pedestrian model minimum speed was set to $0.9\,\mathrm{m\,s^{-1}}$
- Ego vehicle's predicted trajectory was changed to ego vehicle's predicted occupancy.
- Introduction of the stopped model.
- Creation of Intersection Maps that represent the intersecting area between ego vehicle and obstacles for a specified time horizon.
- Creation of Temporal Occupancy Maps that represent the predicted distributions for several time horizons simultaneously.
- Creation of Risk Maps that represent the intersecting area between ego vehicle and obstacles for several time horizons simultaneously.
- The collision risk is the maximum probability value within the Risk Maps, supported by the mean, median and standard deviation of the twenty highest values.

6.2 Future work

This dissertation leaves multiple paths open to improve the perception and the collision risk assessment of the ATLASCAR2. These are listed in the following topics regarding the improvement of the Risk Maps approach and the overall perception of the ATLASCAR2:

- Improve ATLASCAR2 vision system by installing more cameras or a 360° camera ensuring surround view of the environment.
- Implement more sophisticated sensor fusion techniques to enhance the 3D detection of the obstacles.
- Implement obstacle orientation estimation within the point cloud or the image.
- Refine the Risk Maps pedestrian model to accurately transmit the collision risk.
- Improve the Risk Maps into a 360° surrounding grid.
- Include the potential damage of the collision in the risk evaluation.

Intentionally blank page.

References

[1] Automation and R. group, *Atlas project*. [Online]. Available: https://atlas.web.ua.pt/.

[2] World Health Organization, *Road traffic injuries*, Organizational author. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[3] *28 top self-driving car companies to know in 2023 — built in*. [Online]. Available: https://builtin.com/transportation-tech/self-driving-car-companies.

[4] Cruise, *Cruise recalls all self-driving cars after grisly accident and california ban — self-driving cars — the guardian*. [Online]. Available: https://www.theguardian.com/technology/2023/nov/08/cruise-recall-self-driving-cars-gm.

[5] M. Gomes, M. Oliveira, S. Pombinho, B. Lourenço, A. Castro, and D. Rato, *Core packages for ATLASCAR2*, 2022. [Online]. Available: https://github.com/lardemua/atlascar2.

[6] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3d lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," *Robotics and Autonomous Systems [serial on the internet]*, vol. 83, no. 4, pp. 299–311, 2016. [Online]. Available: https://www.researchgate.net/.

[7] *Envision the future — velodyne lidar*. [Online]. Available: https://velodynelidar.com/.

[8] S. Sun, *Lidar obstacle detector*, version 1.0.0, Dec. 2021. [Online]. Available: https://github.com/SS47816/lidar_obstacle_detector.

[9] *Rqt reconfigure - ROS wiki*. [Online]. Available: http://wiki.ros.org/rqt_reconfigure.

[10] G. M. C. Ribeiro, "Perception using multi-tasked neural networks on ATLASCAR2," Master's Thesis, Universidade de Aveiro, 2023.

[11] G. Ribeiro and V. Santos, "Single and multi-tasked neural networks: Selection and deployment," in *2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2024, pp. 41–46. DOI: 10.1109/ICARSC61747.2024.10535918.

[12] J. Solawetz and Francesco, *What is yolov8? the ultimate guide*. 2023. [Online]. Available: https://blog.roboflow.com/whats-new-in-yolov8/.

[13] C. Han, Q. Zhao, S. Zhang, Y. Chen, Z. Zhang, and J. Yuan, "Yolopv2: Better, faster, stronger for panoptic driving perception," *ArXiv e-prints*, 2022. [Online]. Available: http://arxiv.org/abs/2208.11434.

[14] T. Reaney, *Video stitcher*, version 1.0.0, 2018. [Online]. Available: https://github.com/Toemazz/VideoStitcher.

[15] M. A. R. de Oliveira *et al.*, "Atom: A general calibration framework for multi-modal, multi-sensor systems," *Expert Systems with Applications*, p. 118 000, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422012234.

[16] *Atom documentation*. [Online]. Available: https://lardemua.github.io/atom_documentation/.

[17] G. A. Kumar, J. H. Lee, J. Hwang, J. Park, S. H. Youn, and S. Kwon, "Lidar and camera fusion approach for object distance estimation in self-driving vehicles," *Symmetry*, vol. 12, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/2/324.

[18] X. Xiong, S. Zhang, and Y. Chen, "Review of intelligent vehicle driving risk assessment in multi-vehicle interaction scenarios," *World Electric Vehicle Journal*, vol. 14, no. 12, p. 348, 2023. [Online]. Available: https://www.researchgate.net/.

[19] W. M. D. Chia, S. L. Keoh, C. Goh, and C. Johnson, "Risk assessment methodologies for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 16 923–16 939, 2022. [Online]. Available: https://ieeexplore.ieee.org/.

[20] L. Rummelhard, A. Nègre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 2485–2490.

[21] U. Patil, A. Renzaglia, A. Paigwar, and C. Laugier, "Real-time Collision Risk Estimation based on Stochastic Reachability Spaces," in *ICAR 2021 - International Conference on Advanced Robotics*, Ljubljana, Slovenia, Dec. 2021, pp. 1–6. DOI: 10.1109/ICAR53236.2021.9659485. [Online]. Available: https://inria.hal.science/hal-03416222.

[22] T. Stolte, G. Bagschik, A. Reschka, and M. Maurer, "Hazard analysis and risk assessment for an automated unmanned protective vehicle," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1848–1855.

[23] D. Ni, "A unified perspective on traffic flow theory part i: The field theory," *Applied Mathematical Sciences*, vol. 7, no. 39, pp. 1929–1946, 2013. [Online]. Available: www.m-hikari.com.

[24] B. Wu, Y. Yan, D. Ni, and L. Li, "A longitudinal car-following risk assessment model based on risk field theory for autonomous vehicles," *International Journal of Transportation Science and Technology*, vol. 10, no. 1, pp. 60–68, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2046043020300423.

[25] A. Arun, M. M. Haque, A. Bhaskar, S. Washington, and T. Sayed, "A systematic mapping review of surrogate safety assessment using traffic conflict techniques," *Accident Analysis and Prevention*, vol. 153, p. 106 016, 2021. [Online]. Available: https://www.researchgate.net/.

[26] J. Zhu, Y. Ma, and Y. Lou, "Multi-vehicle interaction safety of connected automated vehicles in merging area: A real-time risk assessment approach," *Accident Analysis and Prevention*, vol. 166, p. 106 546, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457521005777.

[27] M. Strickland, G. Fainekos, and H. Ben Amor, "Deep predictive models for collision risk assessment in autonomous driving," in *2018 IEEE International Conference on Robotics and Automation (ICRA) : May 21-25, 2018, Brisbane, Australia.*, 2018, pp. 1–8.

[28] J. Reich, M. Wellstein, I. Sorokos, F. boril, and K.-U. Scholl, "Towards a software component to perform situation-aware dynamic risk assessment for autonomous vehicles," in *Dependable Computing - EDCC 2021 Workshops*. Springer International Publishing, 2021, pp. 3–11.

[29] D. Shin, B. Kim, J. Seo, and K. Yi, "Effects of wireless communication on integrated risk management based automated vehicle," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 1767–1772.

[30] H. Wang, Y. Huang, A. Khajepour, Y. Zhang, Y. Rasekhipour, and D. Cao, "Crash mitigation in motion planning for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3313–3323, 2019. [Online]. Available: https://ieeexplore.ieee.org/.

[31] M. Parseh, F. Asplund, L. Svensson, W. Sinz, E. Tomasch, and M. Torngren, "A data-driven method towards minimizing collision severity for highly automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 4, pp. 723–735, 2021. [Online]. Available: https://ieeexplore.ieee.org/.

[32] *Iglad data description*. [Online]. Available: http://www.iglad.net/web/page.aspx?refid=2.

[33] Mitsubishi, *Mitsubishi i-MiEV*. [Online]. Available: https://ev-database.org/car/1096/Mitsubishi-i-MiEV.

[34] P. Grey, *Point grey fl3-ge-28s4c-c*. [Online]. Available: https://www.avsupply.com/ITM/29694/FL3-GE-28S4C-C.html.

[35] Logitech, *Logitech hd webcam c270 technical specifications*. [Online]. Available: https://support.logi.com/hc/en-my/articles/360023462093-Logitech-HD-Webcam-C270-Technical-Specifications.

[36] S. C. Pombinho, "Integrated software architecture in ATLASCAR2," Master's Thesis, Universidade de Aveiro, 2022.

[37] Farnell, *Rotary encoder, optical, incremental, 1000 ppr, 0 detents, horizontal, without push switch*. [Online]. Available: https://pt.farnell.com/hengstler/ri32-0-1000er-14kb/encoder-rotary/dp/615985.

[38] Arduino, *Arduino uno wifi rev2*. [Online]. Available: https://store.arduino.cc/products/arduino-uno-wifi-rev2.

[39] Botnroll, *Can-bus shield v2.0*. [Online]. Available: https://www.botnroll.com/pt/comunicacao/2470-can-bus-shield-v20.html.

[40] K. Kuchera, *Canalyze - native can interface for linux*. [Online]. Available: https://kkuchera.github.io/canalyze/.

[41] Asus, *Asus rog strix g15 g512li-70at5pb1*. [Online]. Available: https://www.fnac.pt/Computador-Portatil-Gaming-Asus-ROG-Strix-G15-G512LI-70AT5PB1-Computador-Portatil-Computador-Portatil-Gaming/a7788764.

[42] M. Quigley *et al.*, "ROS: An open-source robot operating system," vol. 3, Jan. 2009.

[43] R. D. Team, *RVIZ: 3d visualization tool for ROS*. [Online]. Available: http://wiki.ros.org/rviz.

[44] C. Aguero *et al.*, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," *Automation Science and Engineering, IEEE Transactions on*, vol. 12, no. 2, pp. 494–506, Apr. 2015.

[45] Velodyne, *Velodyne vlp-16 specifications*. [Online]. Available: https://velodynelidar.com/wp-content/uploads/2019/12/63-9229_Rev-K_Puck-_Datasheet_Web.pdf.

[46] *Collision risk estimation using stochastic motion models*. [Online]. Available: https://github.com/patilunmesh/collision_risk_estimation.

[47] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[48] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[49] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, A. Koubaa, Ed., Springer, 2016, ch. 5, ISBN: 978-3-319-26052-5. DOI: 10.1007/978-3-319-26054-9{_}5. [Online]. Available: http://www.springer.com/de/book/9783319260525.

[50] Healthline, *What is the average walking speed of an adult?* [Online]. Available: https://www.healthline.com/health/exercise-fitness/average-walking-speed#what-is-a-brisk-pace.

[51] *Ackermann steering controller*. [Online]. Available: http://wiki.ros.org/ackermann_steering_controller.

[52] *Geometry twist ROS messages*. [Online]. Available: http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Twist.html.

[53] *Navigation odometry ROS messages*. [Online]. Available: http://docs.ros.org/en/noetic/api/nav_msgs/html/msg/Odometry.html.

[54] R. D. Team, *Rqt graph*. [Online]. Available: http://wiki.ros.org/rqt_graph.

[55] Ultralytics, *YOLOv5: A state-of-the-art real-time object detection system*, 2021. [Online]. Available: https://docs.ultralytics.com.

[56] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[57] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[58] ROS, *Message filters for python api.* [Online]. Available: `https://docs.ros.org/en/api/message_filters/html/python/#message_filters.TimeSynchronizer`.

[59] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: A llvm-based python jit compiler," in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 2015, pp. 1–6.

[60] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: `10.1038/s41586-020-2649-2`. [Online]. Available: `https://doi.org/10.1038/s41586-020-2649-2`.

[61] R. developer team, *Marker array ROS messages.* [Online]. Available: `http://docs.ros.org/en/noetic/api/visualization_msgs/html/msg/MarkerArray.html`.

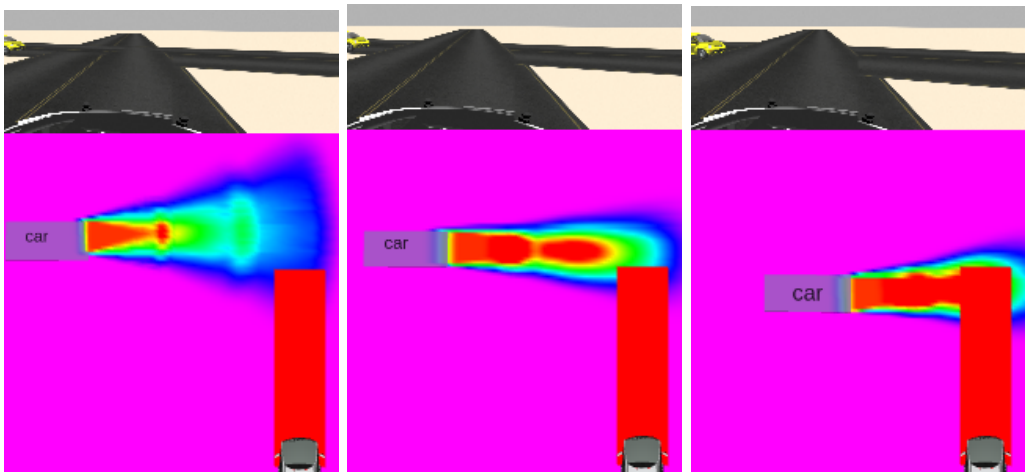
[62] J. Bowman and P. Mihelich, *Ros camera calibration.* [Online]. Available: `http://wiki.ros.org/camera_calibration`.

Intentionally blank page.

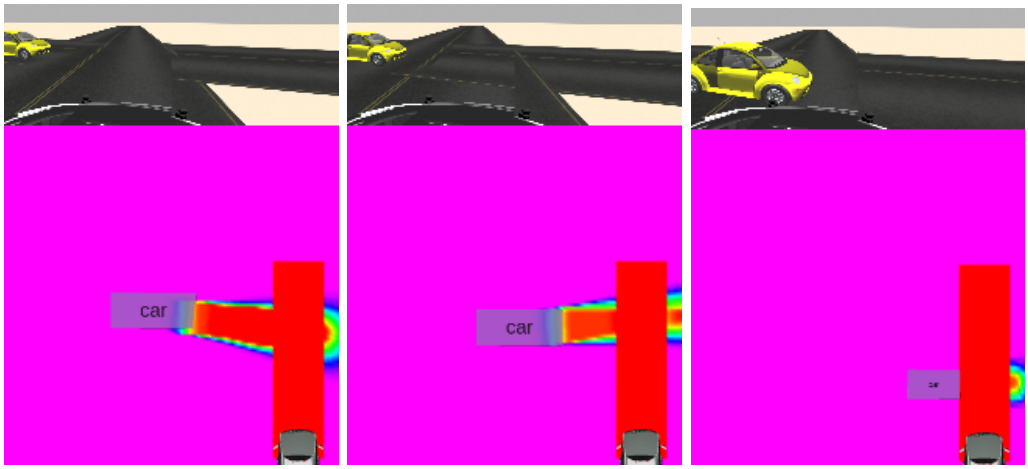
Appendix A

Temporal Occupancy Maps

This appendix contains figures of the Temporal Occupancy Maps, which correspond to the Risk Maps figures in Chapter 5.

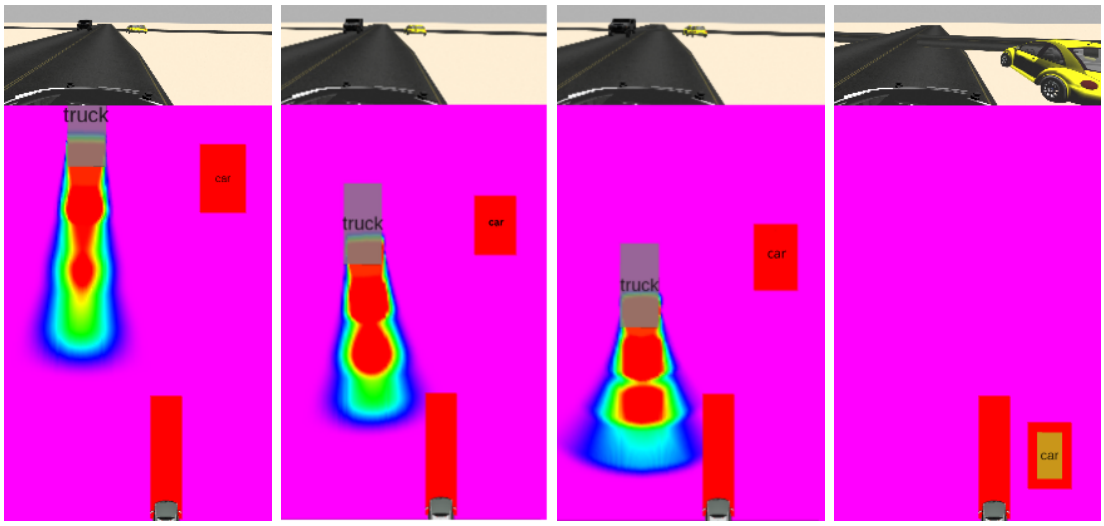


(a) frame 7 (b) frame 9 (c) frame 17



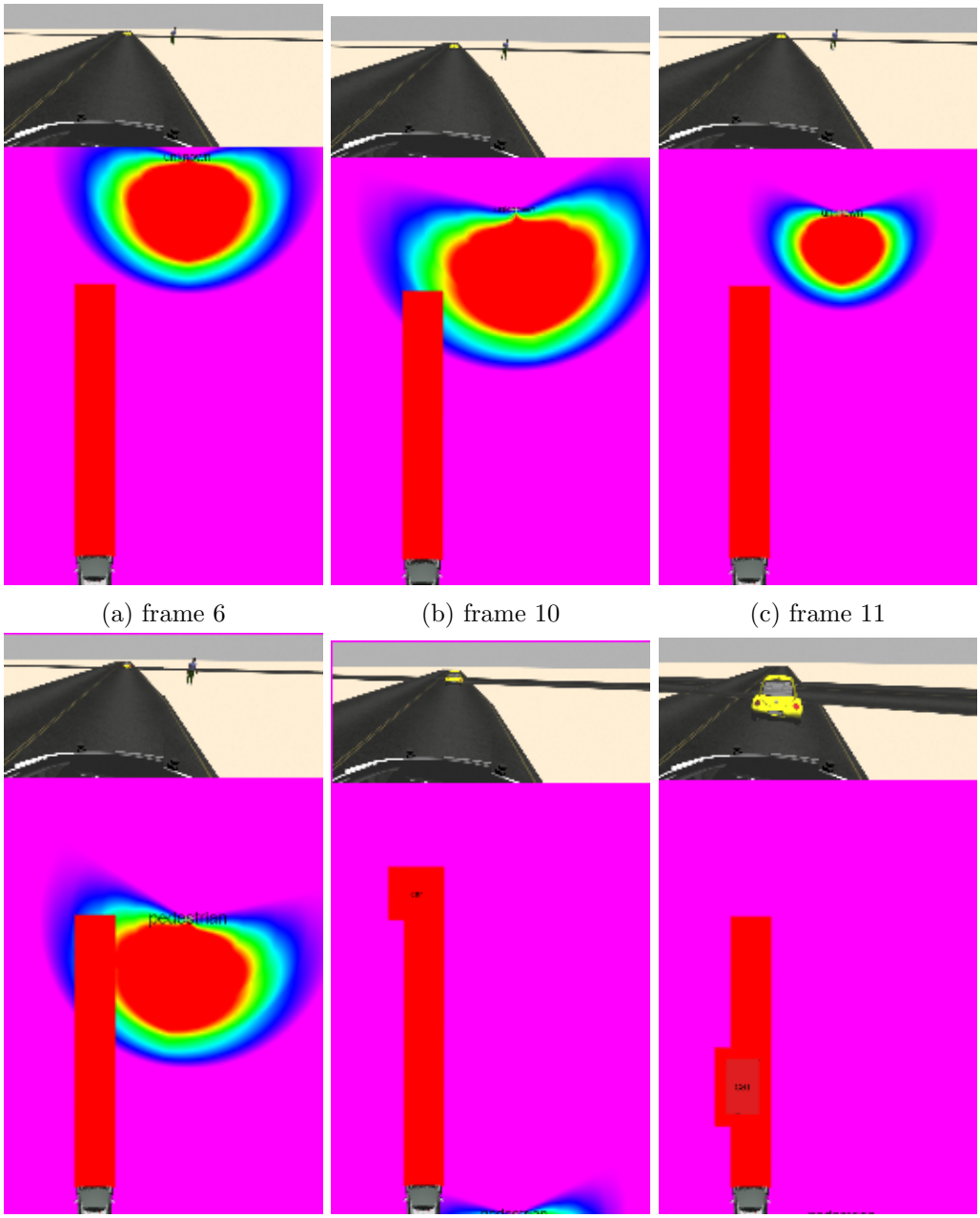
(d) frame 21 (e) frame 24 (f) frame 34

Figure A.1: Correspondent Temporal Occupancy Maps frames of the junction scenario



(a) frame 23 (b) frame 35 (c) frame 42 (d) frame 781

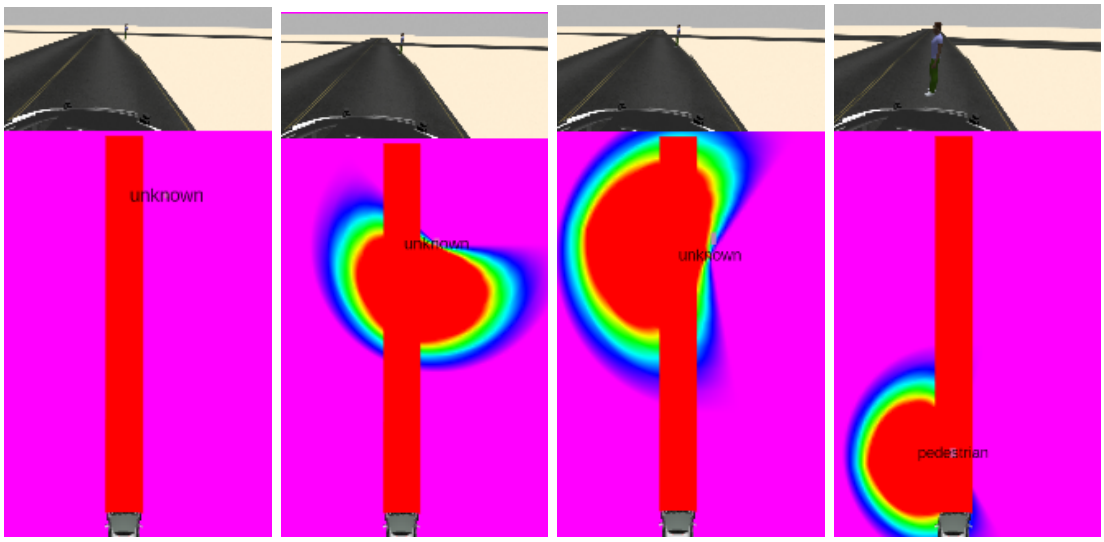
Figure A.2: Correspondent Temporal Occupancy Maps frames of the opposite lane pass scenario



(a) frame 6 (b) frame 10 (c) frame 11

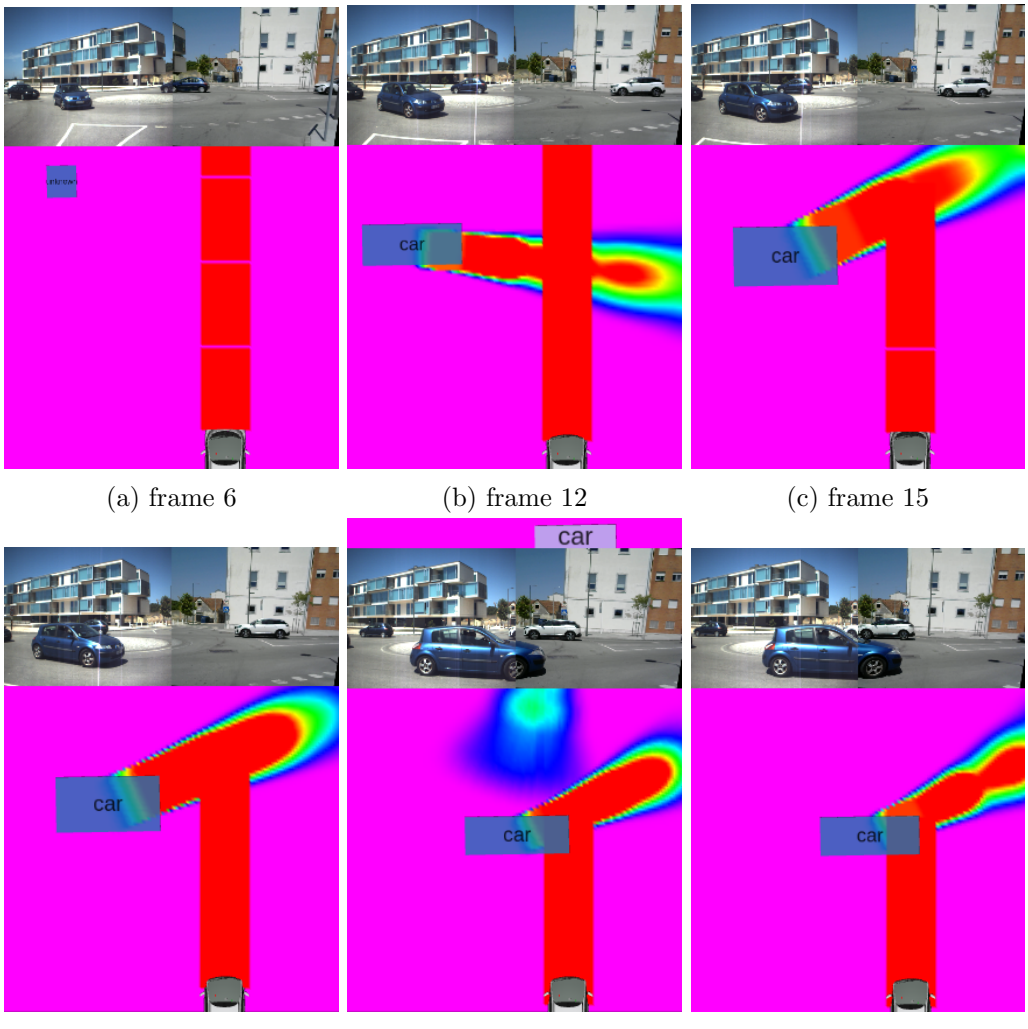
(d) frame 18 (e) frame 45 (f) frame 68

Figure A.3: Correspondent Temporal Occupancy Maps frames of the lane collision scenario



(a) frame 7 (b) frame 11 (c) frame 12 (d) frame 28

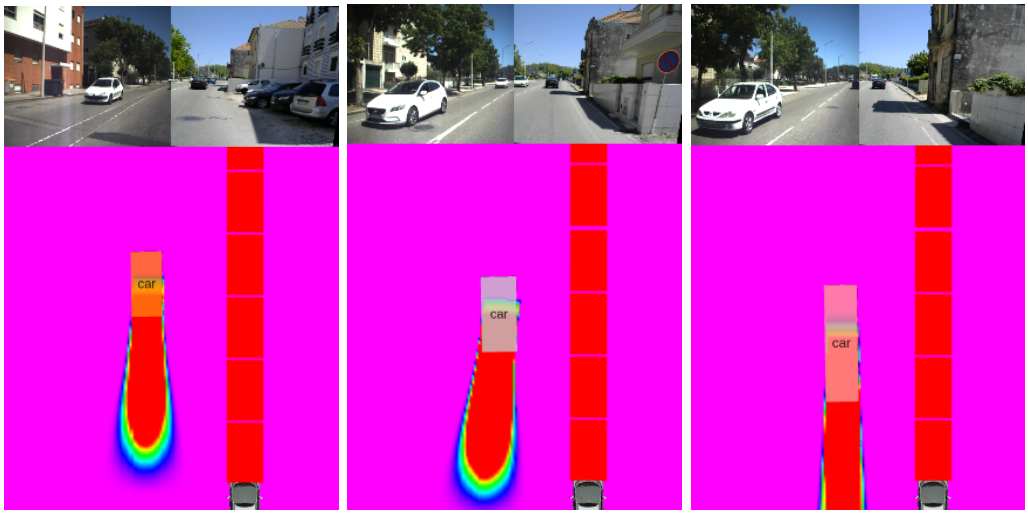
Figure A.4: Correspondent Temporal Occupancy Maps frames of the crosswalk approach scenario



(a) frame 6 (b) frame 12 (c) frame 15

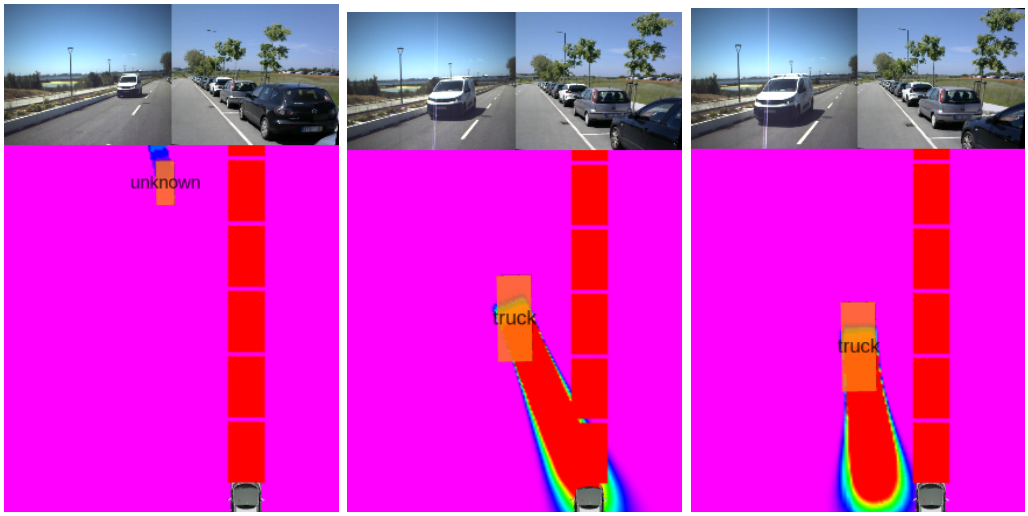
(d) frame 16 (e) frame 23 (f) frame 24

Figure A.5: Correspondent Temporal Occupancy Maps frames of the roundabout approach scenario



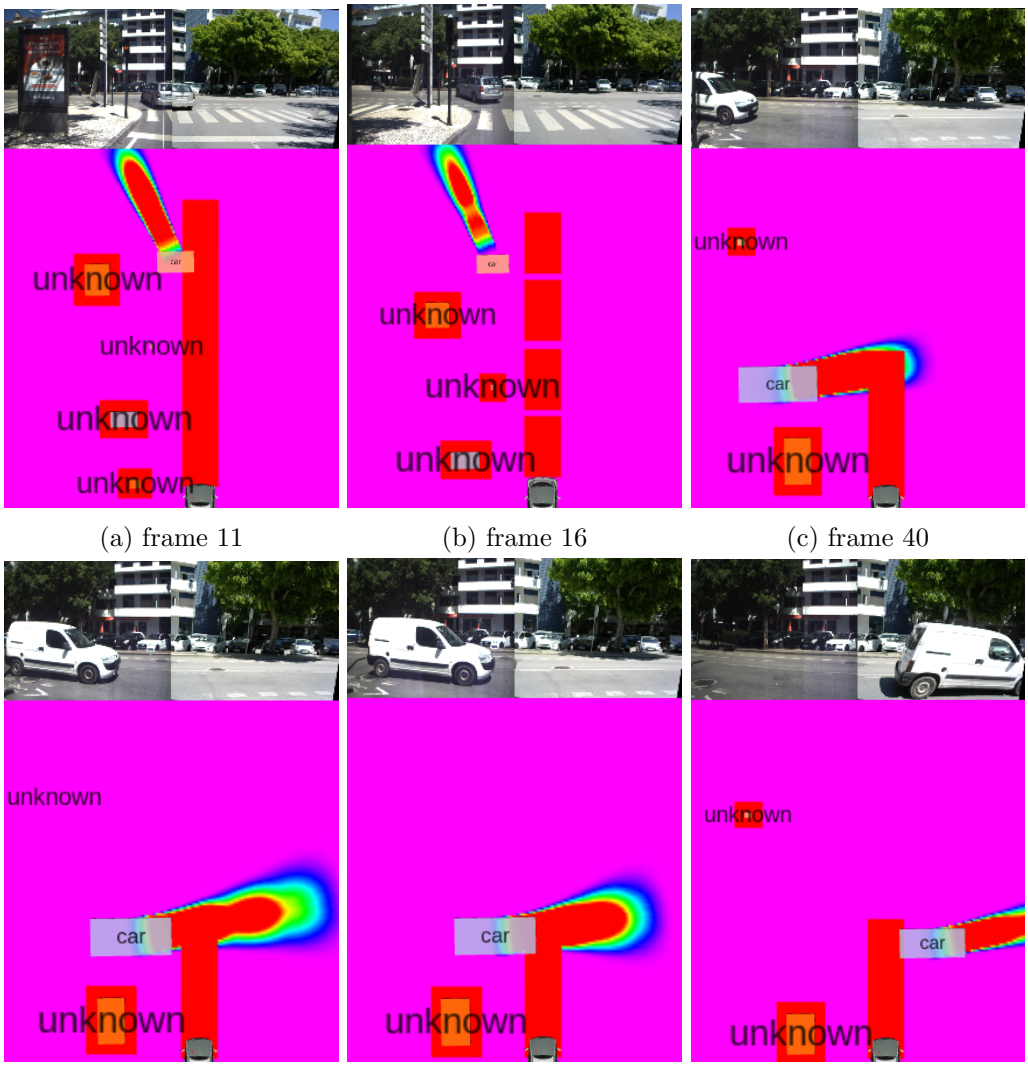
(a) frame 12 (b) frame 38 (c) frame 50

Figure A.6: Correspondent Temporal Occupancy Maps of the wide opposite lane pass scenario



(a) frame 21 (b) frame 26 (c) frame 27

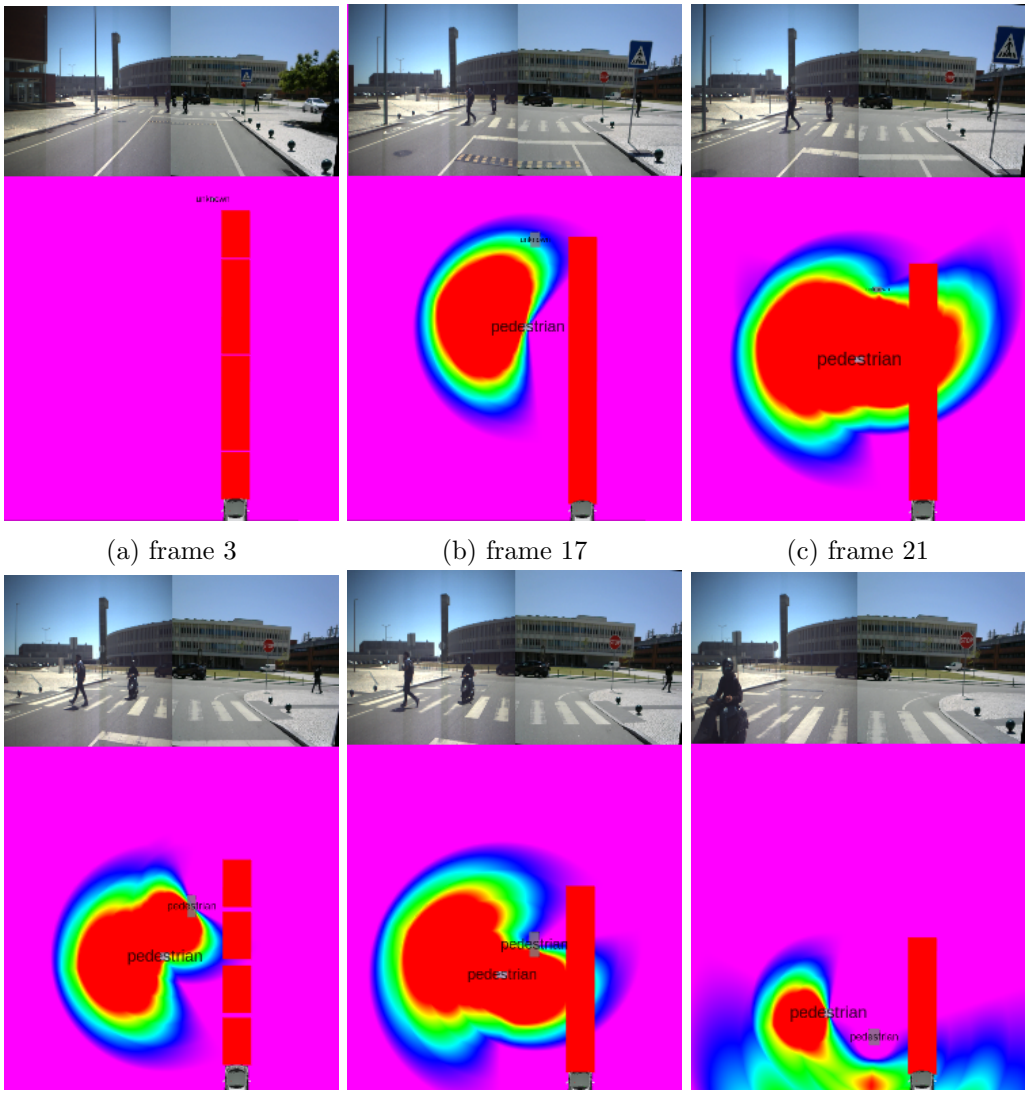
Figure A.7: Correspondent Temporal Occupancy Maps frames of the tight opposite lane pass scenario



(a) frame 11 (b) frame 16 (c) frame 40

(d) frame 42 (e) frame 43 (f) frame 48

Figure A.8: Correspondent Temporal Occupancy Maps frames of the junction approach scenario



(a) frame 3 (b) frame 17 (c) frame 21

(d) frame 25 (e) frame 28 (f) frame 36

Figure A.9: Correspondent Temporal Occupancy Maps frames of the crosswalk approach scenario