Navigating the Uncharted: Leveraging Modular Architectures and End-to-End Learning for Autonomous Driving in Unmapped Environments

Luis Alberto Rosero Rosero

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

Luis Alberto Rosero Rosero

Navigating the Uncharted: Leveraging Modular Architectures and End-to-End Learning for Autonomous Driving in Unmapped Environments

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Fernando Santos Osório

USP – São Carlos
August 2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

```
       Rosero Rosero, Luis Alberto
R795n     Navigating the Uncharted: Leveraging Modular
       Architectures and End-to-End Learning for
       Autonomous Driving in Unmapped Environments / Luis
       Alberto Rosero Rosero; orientador Fernando  Santos
       Osório. -- São Carlos, 2024.
          112 p.


          Tese (Doutorado - Programa de Pós-Graduação em
       Ciências de Computação e Matemática Computacional) --
        Instituto de Ciências Matemáticas e de Computação,
       Universidade de São Paulo, 2024.


          1. Autonomous Driving. 2. Modular Architectures.
       3. Perception system. 4. End-to-End learning. 5.
       Intelligent and Autonomous Vehicles. I. Santos
       Osório, Fernando , orient. II. Título.
```

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Luis Alberto Rosero Rosero

Navegando pelo desconhecido: utilizando arquiteturas modulares e aprendizado para direção autônoma em ambientes não mapeados

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Prof. Dr. Fernando Santos Osório

USP – São Carlos
Agosto de 2024

Este trabalho é dedicado aos meus pais Cardenio e Lucía, aos meus irmãos Carlos e Darío, que sempre me deram seu amor.

ACKNOWLEDGEMENTS

I am grateful to everyone who me throughout my doctoral journey. Foremost, I thank my advisor, Professor Fernando Osório, for giving me the opportunity to join the group and for his unwavering friendship and encouragement throughout.

I extend my heartfelt thanks to my family, especially my parents, Cardenio and Lucía, and my brothers, Carlos and Darío. Their constant support, encouragement, and motivation have been the driving force behind my daily progress.

I am equally grateful to all the members of the Mobile Robotics Laboratory (LRM) for their invaluable help and collaboration during this time.

Finally, I express my sincere thanks to FAPESP (PIPE), CAPES, and Rota 2030 for providing the financial support that made this research possible.

"I have learned that everybody wants to live at the top of the mountain
without realizing that true happiness lies
in the way we climb the slope. "

(Gabriel García Márquez)

RESUMO

ROSERO, L. A. **Navegando pelo desconhecido: utilizando arquiteturas modulares e aprendizado para direção autônoma em ambientes não mapeados**. 2024. 112 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

A condução autônoma promete uma revolução na área de transportes, proporcionando benefícios sociais e econômicos significativos. Apesar dos avanços notáveis na tecnologia de veículos autônomos em ambientes mapeados, a navegação em áreas não mapeadas continua a ser um desafio persistente.

Esta tese investiga o desenvolvimento de arquiteturas de condução autônoma para navegação em tempo real, com e sem mapas. Três abordagens são propostas, implementadas, comparadas e avaliadas:

Arquitetura modular: Um agente personalizado realiza a coleta de dados e serve como linha de base para navegação baseada em mapa. Algoritmos tradicionais e novos módulos para percepção, tomada de decisão e previsão são integrados para garantir uma navegação segura em ambientes mapeados. Este agente atua como “professor” para os agentes de navegação sem mapa.

Aprendizagem *end-to-end*: As redes neurais aprendem políticas a partir de dados por meio de técnicas de aprendizagem por imitação. A simplicidade é priorizada para operação em tempo real em ambientes sem mapas. Diferentes tipos de sensores e métodos de fusão são explorados para melhorar o desempenho.

Arquitetura Híbrida: Combinando a interpretabilidade de sistemas modulares com a capacidade de aprendizagem de modelos *end-to-end*, esta abordagem integra o planejamento de trajetória baseado em dados com módulos de percepção, localizacao, tomada de decisao e controle. Oferece robustez, flexibilidade e adaptabilidade.

Além disso, um *framework* baseado em ROS denominada "*CaRINA agent*" é desenvolvido para implementar *pipelines* modulares e facilitar a incorporação de métodos *end-to-end* e a construção de arquiteturas híbridas. Para avaliar de forma abrangente nossas metodologias, aproveitamos os Leaderboards do CARLA, alcançando resultados competitivos tanto no Leaderboard 1 quanto no Leaderboard 2, classificando as nossas abordagens especificamente entre os primeiros nas categorias SENSORES e MAP. Além disso, a nossa arquitetura modular e agente híbrido garantiram o 1º e o 2º lugar no CARLA Autonomous Driving Challenge (CADCH) 2023, mostrando a eficácia das abordagens propostas.

Palavras-chave: Condução autônoma; arquitetura híbrida; arquitetura modular; end-to-end; planejamento de trajetória; percepção; tomanda de decisão; predição; disparidade; Simulador CARLA; Veículos Autônomos e Inteligentes.

ABSTRACT

ROSERO, L. A. **Navigating the Uncharted: Leveraging Modular Architectures and End-to-End Learning for Autonomous Driving in Unmapped Environments**. 2024. 112 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

Autonomous driving promises a revolution in transportation, unlocking significant social and economic benefits. Despite notable advancements in autonomous vehicle technology tailored for mapped environments, navigating in unmapped areas remains a persistent challenge. The limited utilization of developments in modular pipelines exacerbates this issue, impeding progress towards map-free navigation.

This thesis delves into the development of autonomous driving architectures for real-time navigation, both with and without maps. Three approaches are proposed, implemented, compared, and evaluated to create new and robust methodologies:

Modular Pipeline: A custom agent performs data collection and serves as a baseline for map-based navigation. Traditional algorithms and new modules for perception, decision-making, and prediction are integrated to ensure safe navigation in mapped environments. This agent acts as the "teacher" for the mapless navigation agents.

End-to-End Learning: Neural networks learn driving policies from data through imitation learning techniques. Simplicity is prioritized for real-time operation in map-free environments. Different sensor types and fusion methods are explored to enhance performance.

Hybrid Architecture: Combining the interpretability of modular systems with the learning capabilities of end-to-end models, this approach integrates data-driven path planning with modular perception and control modules. It offers robustness, flexibility, and adaptability.

Furthermore, a ROS-based framework named "CaRINA agent" is developed to implement modular pipelines and facilitate incorporating end-to-end methods and constructing hybrid architectures. To comprehensively evaluate our methodologies, we leverage the CARLA Leaderboards, achieving competitive results in both Leaderboard 1 and Leaderboard 2, specifically ranking among the top in the SENSORS and MAP categories. Moreover, our modular architecture and hybrid agent secured 1st and 2nd place in the 2023 CARLA Autonomous Driving Challenge (CADCH), underscoring the effectiveness of our proposed approaches.

Keywords: Autonomous driving; hybrid architecture; modular architecture; end-to-end; path planning; perception; decision-making; prediction; disparity; CARLA simulator; Intelligent and Autonomous Vehicles.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance
BEV	Bird's Eye View
CADCH	CARLA Autonomous Driving Challenge
CaRINA	Carro Robótico Inteligente para Navegação Autônoma
CNN	Convolutional neural network
CRA	Collision Risk Assessment
DCNN	Deep Cascaded Neural Network
FSM	Finite State Machine
GAT	Graph Attention Network
GRU	Gated Recurrent Unit
HD	High-Definition
HMI	Human-Machine Interface
IoU	intersection over union
LiDAR	light detection and ranging
LRM	Mobile Robotics Lab.
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
MOT	multi-object tracking
NHTSA	National Highway Traffic Safety Administration of the United States
OpenCV	Open Source Computer Vision Library
R-CNN	Region-based CNN
ROS	Robotic Operating System
RPN	Region Proposal Network
RTOS	real-time operating system
SAD	sum of absolute differences
SAE	Society of Automotive Engineers
SGBM	semi global block matching
SGM	semi global matching
SORT	Simple Online and Realtime Tracking
SSD	sum of squared differences

TTC	time to collision
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2X	Vehicle to Anything
YOLO	"You Only Look Once"

CONTENTS

CHAPTER 1

INTRODUCTION

Autonomous driving has the potential to revolutionize transportation, offering a multitude of societal benefits beyond mere convenience. The reduction of human error, a significant contributor to road accidents, holds the promise of significantly decreasing crashes and fatalities. Moreover, this technology provides mobility solutions for individuals with disabilities who may encounter challenges driving themselves. By facilitating communication among autonomous vehicles and infrastructure, coordinated traffic patterns could be established, thereby mitigating congestion and reducing fuel consumption.

Various methodologies exist for developing autonomous systems, encompassing software components and algorithms from fields such as machine learning, computer vision, decision theory, and probability theory. The standard approach employs modular pipelines, proven effective in scenarios with access to detailed High-Definition (HD) maps or dense waypoints. This approach, widely adopted by both companies and research groups (TENG *et al.*, 2023), decomposes the navigation problem into specific tasks such as localization, object detection, tracking, prediction, decision-making, path planning, and control (TAMPUU *et al.*, 2020; JO *et al.*, 2015; LIU *et al.*, 2017). On the other hand, recent advancements in autonomous driving have introduced end-to-end learning, aiming to directly map sensor input to driving actions without explicit task decomposition (CHEN *et al.*, 2023).

1.1 Context

Levels of Driving Automation: The Society of Automotive Engineers (SAE) defines six levels of vehicle driving automation ranging from Level 0 (no driving automation) to Level 5 (full driving automation) in the context of motor vehicles and their operation on roadways. Where Level 0 means there is no driving automation and the human is responsible for driving. Level 1 is driver assistance. Level 2 offers partial driving automation, where the vehicle can control acceleration and deceleration but the driver must be attentive and ready to take control at any time. At Level 3 the vehicle can perform most driving tasks under certain conditions but the driver must be prepared to intervene if necessary. At Level 4 the vehicle has a high level of automation where it can perform all driving tasks in specific scenarios without human intervention, the human has the option to intervene. Finally, Level 5 represents full driving automation, where vehicles are capable of driving under any conditions without the need for human intervention[1].

It is known that there are currently commercial models at Level 2 (Tesla) and Level 3 (Mercedes-Benz, BMW), but there are no commercial vehicles at Level 4 (Mercedes offers Level 4 only for autonomous parking). In recent years, various automobile manufacturers and related companies have cooperated to develop autonomous driving technology worldwide. This has begun in some areas, such as Level 4 autonomous driving taxis operating in limited areas in the United States, for example, Waymo, Motional, and Cruise.

This work is part of the research conducted at the Mobile Robotics Lab. (LRM) of ICMC-USP in São Carlos. Historically, this laboratory began its research and initial tests with intelligent and autonomous vehicles in 2009, fifteen years ago, in a project funded by INCT-SEC. During this period, three generations of intelligent and fully autonomous vehicles were developed, namely Carro Robótico Inteligente para Navegação Autônoma (CaRINA) 1 (electric vehicle "Club Car") (FERNANDES *et al.*, 2012) (KLASER; OSóRIO; WOLF, 2014), CaRINA 2 (Fiat Pálio Adventure) (FERNANDES *et al.*, 2014), and Smart Truck (Scania partnership) [2]. The work presented in this Thesis began during the master's degree and follows up the development of this Thesis, with significant contributions to the architecture of the intelligent systems of the vehicles and their perception modules (Vision, Radar, LiDAR), as well as participating in the projects developed by the LRM team in partnership with industry (Scania and Vale) (CALDAS *et al.*, 2023).

1.2 Problem Statement

While the advantage of modular architectures lies in its interpretability, extensive coupling of numerous components increases the risk of error propagation, resulting in heightened complexity in maintaining the entire architecture, associated costs. This mix of components adds complexity to both the development and evaluation process. Furthermore, reliance on high-definition (HD) maps for navigation poses limitations, including restricted coverage, privately held maps, and potential processing issues or access errors.

To overcome these limitations, end-to-end methods for autonomous driving have been used. Although this approach streamlines the system architecture and eliminates manual feature

[1] Available at: <https://www.sae.org/binaries/content/assets/cm/content/blog/sae-j3016-visual-chart_5.3.21.pdf>

[2] Available at: <http://lrm.icmc.usp.br/web/index.php?n=Eng.ProjSTruck>

engineering, it often requires substantial amounts of diverse training data, which is costly and time-consuming to collect in real scenarios. Additionally, the opaque decision-making process within end-to-end models raises concerns about safety and accountability, particularly in critical situations.

Another challenge in autonomous driving involves evaluating the architecture employed, whether it follows a modular, end-to-end, or hybrid methodology. Kalra and Paddock (KALRA; PADDOCK, 2016), Koopman and Wagner (KOOPMAN; WAGNER, 2016), and Huang et al. (HUANG *et al.*, 2016) suggest that to comprehensively assess an autonomous system, it is important to combine real-road and simulation tests. In this regard, simulators offer advantages by creating repeatable scenarios for component performance assessments. They simulate diverse driving situations with realistic dynamics, including weather conditions, sensor malfunctions, traffic violations, hazardous events, traffic jams, and crowded streets. Simulations also serve as effective benchmarks, enabling the evaluation of different system approaches under the same conditions for comparative analysis.

Considering all the advances in autonomous driving achieved by modular pipelines in mapped environments and the recent success of machine learning algorithms for autonomous driving, the scientific question arises: Is it possible to combine the advantages of these two approaches to build a new hybrid approach that navigates more safely in both mapped and unmapped environments? Additionally, can we benchmark these approaches in realistic and repeatable scenarios under different weather and traffic conditions?

1.3 Contributions

Based on the aforementioned considerations, this thesis focuses on developing autonomous driving architectures capable of navigating without relying on maps in real-time[3]. To achieve this objective, we propose, implement, compare, and evaluate three approaches:

Modular pipeline: We design a modular agent from scratch to automate data collection and serve as a baseline for map-based navigation. Drawing from previous work in our laboratory, we integrate traditional algorithms and propose new modules for perception, decision-making, and prediction, ensuring safe navigation in mapped environments. These enhancements establish the modular agent as the "teacher" for our end-to-end, mapless navigation agents.

End-to-end agents based on different types of sensors and fusion: For this, we propose autonomous agents based on cloning and imitation learning techniques, where a neural network learns driving policies from data. We prioritize the simplicity of our methods to achieve real time operation while ensuring their effectiveness in navigating environments without map.

[3] In this context, we consider soft real-time, which aim to meet deadlines for data delivery or processing as often as possible. However, occasional missed deadlines are permissible without severe consequences.

We contribute a new hybrid method for mapless autonomous driving by integrating data-driven path planning with modular perception and control modules. This approach offers robustness, flexibility, and adaptability. Our method and framework represent a novel contribution to the development of autonomous driving, ADAS, and mobile robotics, and can be integrated with both existing and new perception and control methods developed by the LRM Lab or external contributors. Our hybrid architecture achieves results comparable to current state-of-the-art algorithms, combining the interpretability of modular systems with the learning capabilities of end-to-end models.

Figure 1 illustrates the main difference between the three approaches for architecture, being modular, end-to-end, and hybrid architecture.

Finally, we develop a framework based on ROS named “CaRINA agent” that implements modular pipelines for autonomous driving, facilitating the replacement of existing modules with others performing perception, planning, localization, decision-making, etc. Additionally, the framework allows for the incorporation of end-to-end methods and construction of hybrid architectures for testing on CARLA leaderboards for both tracks: MAP and SENSORS. The code is available online[4].

1.3.1 Honors and Awards

During the doctoral studies, the author actively participated in international competitions aimed at advancing the fields of autonomous driving and perception. Two notable experiences directly tied to this research are highlighted as follows:

Argoverse 2022 Challenge (Stereo Depth Estimation): This competition tasked participants with developing algorithms for stereo camera disparity calculation, crucial for depth perception in autonomous driving. Leveraging deep learning techniques, the author implemented an algorithm that achieved third place. Results were presented at the esteemed CVPR 2022 Workshop on Autonomous Driving[5]. The technical report is available on the competition website[6].

CARLA Autonomous Driving Challenge (CADCH): Since 2019, the author has participated in the prestigious CARLA Autonomous Driving Challenge, an annual international competition testing self-driving algorithms in simulated urban environments. As a member of Team LRM, the author achieved outstanding results:

CADCH 2019 [7,8]:

[4] Available at: <https://github.com/lrmicmc/CaRINA-agent>.
[5] Available at: <https://www.youtube.com/watch?v=Z1q9ijuLLvU&t=1440s>.
[6] Available at: <https://www.argoverse.org/priorCompetitions.html>.
[7] Available at: <https://web.archive.org/web/20201107230101/https://carlachallenge.org/results-challenge-2019/>.
[8] Available at: <https://agenciabrasil.ebc.com.br/en/geral/noticia/2019-08/brazilians-win-intl-self-driving-car-challenge>.

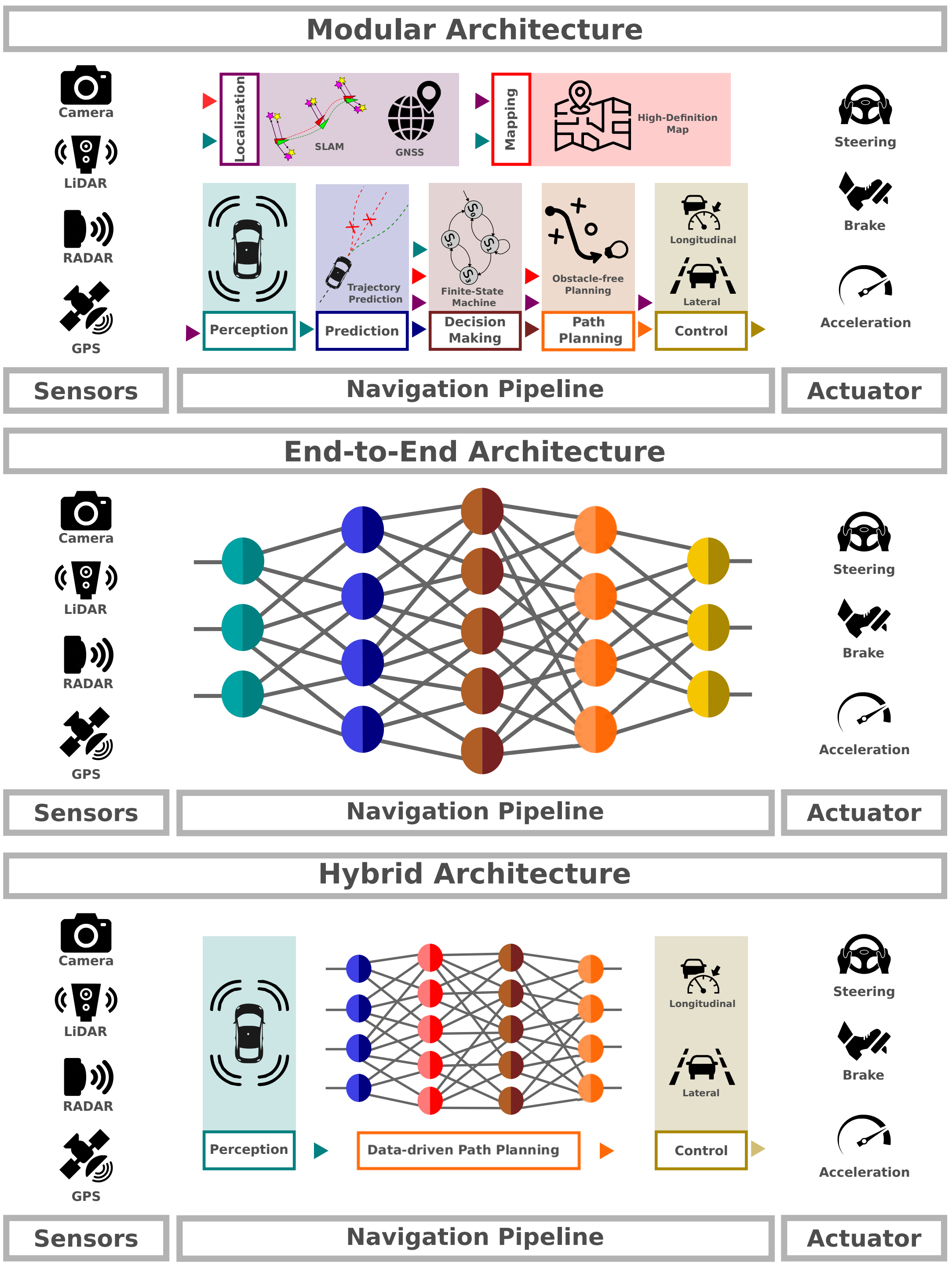


Figure 1 – Example that illustrates the main differences between modular, end-to-end, and hybrid architectures.

Source: Elaborated by the author.

1st Place in 3 out of 4 categories: LiDAR-only, All Sensors, and Map.

2nd Place in Camera-only category.

We made available a technical report (ROSERO *et al.*, 2020) and these results were presented at the Workshop on Autonomous Driving at CVPR 2019.

CADCH 2023 [9]**:**

1st Place in the SENSORS category.

2nd Place in the MAP category.

We publish a article describing our approach in the journal SENSORS (ROSERO *et al.*, 2024) and the author had the opportunity to present developments and results at the NeurIPS 2023 Machine Learning for Autonomous Driving Workshop[10].

1.3.2 Thesis Outline

This thesis is organized as follows: Chapter 2 delves into various deep learning tools utilized in autonomous driving. It covers Convolutional Neural Networks (CNNs) and their applications in object detection, behavior cloning, and imitation learning. Additionally, it provides insights into Long Short-Term Memory (LSTM) networks and reinforcement learning. Chapter 3 provides a critical overview of related-works; Chapter 4 describes the modular software architecture developed in this research, and describe our modules for parsing the OpenDrive map, perception, prediction, risk assessment, and decision-making. Chapter 5 presents our end-to-end implementations for maples autonomous driving. Chapter 6 presents a hybrid architecture approach; Chapter 7 presents our experimental setup and created dataset for training our agents, discusses the results and other experiments; Finally, Chapter 8 addresses the final remarks and suggests some future work.

[9] Available at: <https://leaderboard.carla.org/challenge/>.
[10] Available at: <https://ml4ad.github.io>.

CHAPTER

2

BACKGROUND

In this chapter, we cover theoretical concepts and terminology related to computer vision and machine learning. We provide an overview of the most important concepts in each area to understand our research proposal.

2.1 Convolutional Neural Networks

Convolutional neural network (CNN) (LECUN *et al.*, 1990) is a neural network for processing data that has a grid topology, for example a RGB image. CNN employs a specialized linear mathematical operation called convolution in place of general matrix multiplication. The use of CNN was consolidated with the results obtained by (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) the 2012 ImageNet LSVRC-2012 competition.

Convolutions

In convolutional network terminology, the first argument (in this example, the function x) to the convolution is often referred to as the input, and the second argument (in this example, the function w) as the kernel. The output is some times referred to as the feature map. If we now assume that x and w are defined only on integer t, we can define the discrete convolution:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.1}$$

In machine learning applications, the input is usually a multidimensional array of data, and the kernel is usually a multidimensional array of parameters that are adapted by the learning algorithm.

Pooling layers

The typical layer of a convolutional network consists of three stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage, each linear activation is run through a nonlinear activation function, such as the rectified linear activation function. This stage is sometimes called the detector stage. In the third stage, we use a pooling function to modify the output of the layer further. A pooling function replaces the output of the net at a certain location with a summary statistic of the nearby outputs. For example, the max pooling (ZHOU; CHELLAPPA, 1988) operation reports the maximum output within a rectangular neighborhood. Other popular pooling functions include the average of a rectangular neighborhood, the $L2$ norm of a rectangular neighborhood, or a weighted average based on the distance from the central pixel.

2.1.1 CNNs for classification and detection

Residual Networks

Increasing the depth leads to an increase the accuracy of the network, as long as over-fitting is taken care of. But the problem with increased depth is that the signal required to change the weights, which arises from the end of the network by comparing ground-truth and prediction becomes very small at the earlier layers, because of increased depth. It essentially means that earlier layers are almost negligible learned. This is called vanishing gradient. The second problem with training the deeper networks is, performing the optimization on huge parameter space and therefore naively adding the layers leading to higher training error. Residual networks (HE *et al.*, 2016a) allow training of such deep networks by constructing the network through modules called residual models.

YOLO

The "You Only Look Once" (YOLO) (REDMON *et al.*, 2015) detector is a little bit less precise (Improved on v2) but it is a really fast detector. The idea of this detector is using a CNN model to get the detection on a single pass. First the image is resized to 448×448, then fed to the network and finally the output is filtered by a Non-max suppression algorithm.

Region Proposals R-CNNs

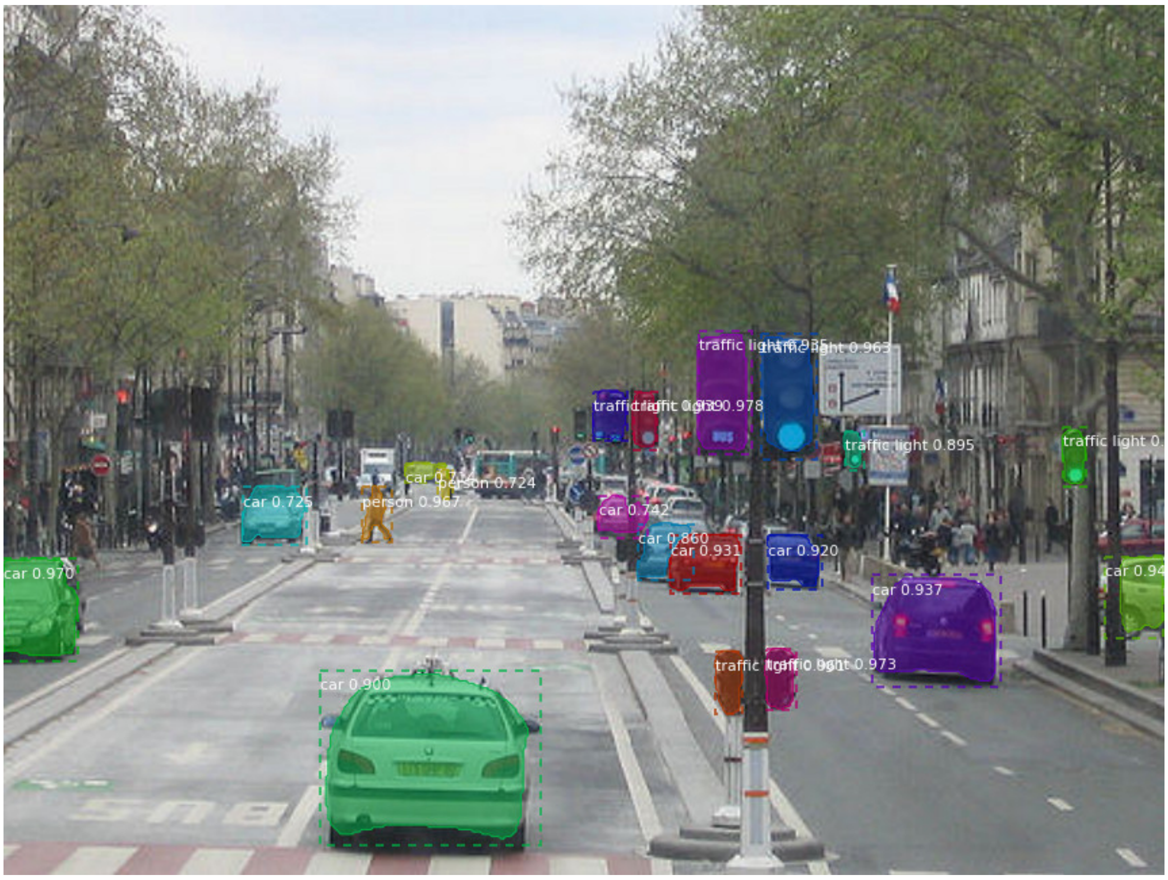
The Region-based CNN (R-CNN) (GIRSHICK *et al.*, 2016) approach performs bounding-box object detection to attend to a manageable number of candidate object regions and evaluate convolutional networks independently on each RoI.

R-CNN was extended by Girshick (2015)(Fast R-CNN) to allow attending to RoIs on feature maps using RoIPool, leading to fast speed and better accuracy. Faster R-CNN (REN *et* *al.*, 2015) advanced this stream by learning the attention mechanism with a Region Proposal Network (RPN). Faster R-CNN is flexible and robust to many follow-up improvements.

Mask R-CNN

The Mask R-CNN method (HE *et al.*, 2017b) detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. This approach extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Figure 2 shows the final predictions from Mask R-CNN for a urban scene.

Figure 2 – Predictions using Mask R-CNN



Source: Abdulla (2017).

2.2 Behavior Cloning

Behavior cloning, also known as imitation learning, is a machine learning technique where an agent (e.g., a robot, game AI, etc.) to learn a task by observing demonstrations from experts, can be a human or another agent demonstrating that task successfully. Instead of explicitly programming rules, it involves learning a policy or function that maps observations of the expert's actions to the actions the agent should take itself.

One of the key components of behavior cloning is the dataset containing pairs of input observations and corresponding actions taken by the expert. This dataset is used to train a supervised learning model, typically a neural network, to map input observations to corresponding actions. During training, the model learns to generalize from the observed data, allowing it to make decisions in new, unseen situations similar to those encountered during training.

Behavior cloning has been applied in various domains, including autonomous driving, robotic manipulation, and video game playing. In autonomous driving, for example, behavior cloning can be used to train a vehicle to mimic the driving behavior of a human driver by learning from a dataset of human driving demonstrations. The trained model can then navigate the vehicle autonomously, making decisions based on the learned behavior. Representative works on behavior cloning for autonomous driving learning from human demonstrations are (POMERLEAU, 1988), (BOJARSKI *et al.*, 2016),(CODEVILLA *et al.*, 2018)

The foundation of behavior cloning lies in gathering high-quality data showcasing the desired behavior. This typically involves:

Expert Demonstrations: Recordings of an expert performing the task successfully. This could include joystick movements, keyboard inputs, or visual/sensor data for autonomous systems.

Data Preprocessing: The raw data might need filtering, cleaning, and formatting to match the format required by the learning algorithm.

Based on the expert demonstrations, a learning algorithm is employed to translate observations into actions. Several options exist, including: Supervised Learning: Techniques like regression learn the mapping between the observed state and the expert's action at that moment.

Modern neural networks like convolutional neural networks (CNNs) can capture complex relationships and patterns in the data, improving imitation accuracy (POMERLEAU, 1988).

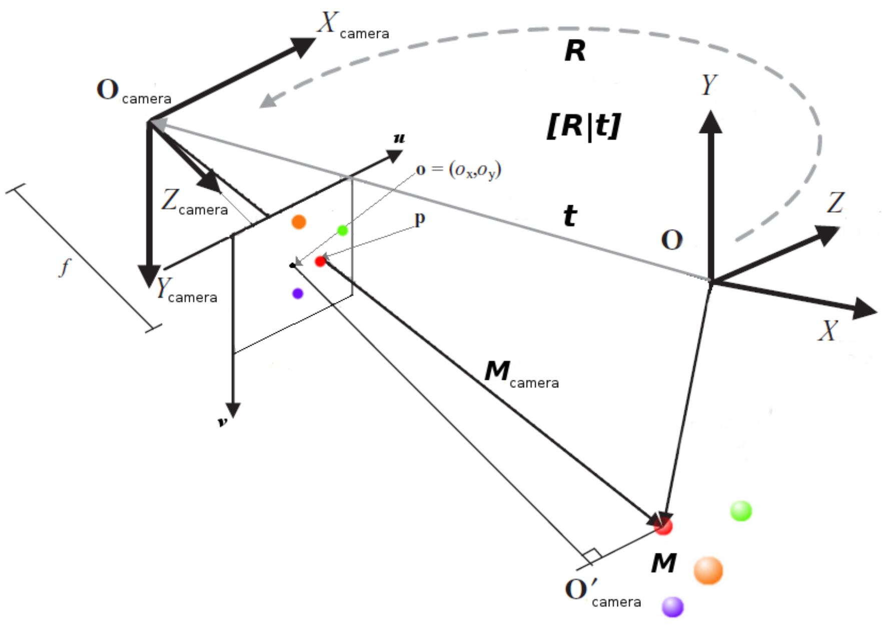
2.3 Computer vision

In this section we explain the fundamental concepts and problems of computer vision. We start with a simple camera model and later we explain what is related to stereo vision and visual motion.

2.3.1 Perspective camera model

The simplest model of a camera is the pinhole (BRADSKI; KAEHLER, 2013). This model assume that a ray reflected by a point in the scene enters through the pinhole. That point is projected onto the image surface. The result of all the projected points is called the projective plane. The size of the image relative to distant objects is given by the focal length (f).

Figure 3 – Pinhole camera model



Source: Elaborated by the author.

The Figure 3 represents the projective transformation that maps M points in the physical world with coordinates (X_i, Y_i, Z_i) to points in the projective plane with coordinates (u_i, y_i). The Equation 2.2 represents the projective transformation where we have a 3×3 matrix of intrinsic parameters (f_x, f_y, o_x, o_y) and a 4×3 matrix of extrinsic parameters $[R|t]$ (R: Rotation matrix r_{11} to r_{33} and t: Translations t_1 to t_3). We can transform 3D data and image frames to other poses in the space (positions and orientations), manipulating these parameters.

$$S \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{2.2}$$

2.3.2 Stereo vision

Stereo imaging systems are based on the capability that our eyes give us to perceive the scene depth. A computational system can emulate that capability by finding correspondences between pixels of two images and computing a three-dimensional image triangulating the correspondences and the 3D point baseline and a known baseline separation between cameras.

Traditional approaches use correlation-based and feature-based techniques to find the correspondences between the pixels onto the images, estimating their disparity (displacement of the pixels that represent the same scene element in each image). Also, it is necessary the:

- Estimation of the fundamental and essential matrix from point correspondences.

- Determination the epipolar geometry and rectify a stereo pair.

- Finally to recover 3D scene from image correspondences using intrinsic and extrinsic parameters.

In order to build a stereo imaging system with two cameras we need to know the geometry of the imaging system and to perform four important steps: undistortion, rectification, correspondence and reprojection.

Epipolar geometry

Epipolar geometry is the basic geometry of a stereo imaging system, this geometry combine a pinhole camera model (one for each camera) and the epipole points. Each camera have a center of projection (O_l for left camera and O_r for right camera) and a π_l and π_r projective planes. A epipole point (e_l or e_r) is defined as the image of the center of projection point (O_l or O_r respectively) on the plane. The plane formed by the point P in 3D space and the two epipole points is called epipolar plane and the lines formed by the intersection of the epipolar plane and the image planes are called epipolar lines. In this way, for a image feature of one of the cameras of the stereo rig, its matching view lies along the epipolar line. This fact is known as epipolar constraint.

Due to the epipolar constraint, searching for matching features across the image plane, from two-dimensional becomes one-dimensional along the epipolar lines once we know the epipolar geometry of the stereo rig.

Stereo calibration and rectification

Stereo calibration is the process of determining the geometric relationships between cameras in space. The stereo calibration finds the rotation matrix R and the translation vector T between the two cameras.

Stereo rectification is the process of individually correcting each image so that the images appear as if they were taken by two cameras with aligned rows and aligned image planes. With the rectification of the optical axes of the two cameras are parallel intersecting at infinity. In the Open Source Computer Vision Library (OpenCV) (BRADSKI, 2000) we can find the methods to calibrate and rectify with stereo cameras (HARTLEY, 1999) where the focus is to get the simplest where the main rays intersect at infinity. The rectification process resides in reprojecting the image camera planes in the same plane, with the rows of the images perfectly aligned within a parallel frontal line.

Finally, the result of horizontally aligning rows with a common image plane containing each image is that the epipoles are located at infinity.

Another result of the calibration is the alignment of the planes, which results in eight terms, four for each image. Additionally, we get a vector containing the distortion coefficients, a rotation matrix R_{rect}, rectified camera matrix M_{rec} and unrectified camera matrix M.

Stereo correspondence

Stereo correspondence is related to a three-dimensional point projected in two different cameras. This method finds pixels in the image on the left that correspond to the same 3D object in the image on the right. Assuming a pair of undeformed and rectified stereo images we can handle depth from the triangulation using disparity measurements $d = x_l - x_r$. In the literature we can find different algorithms for the calculation of disparity some of the most known are sum of absolute differences (SAD) and Census (ZABIH; WOODFILL, 1994). Some computer vision frameworks such as OpenCV implement variations of these algorithms such as Block matching using SAD and a variation of semi global matching (SGM) (HIRSCHMULLER, 2008) that they call semi global block matching (SGBM)

Reprojection

To perform the projection process, we assume that perfect distortion, systematic system. We assume that the planes of the images are coplanar with parallel optical axes and with equal focal lengths $f_l = f_r$. Others assumed that the main points c_x^{left} and c_x^{right} have been calibrated and have their own coordinates in pixels in their respective left and right images.

2.3.3 Visual motion

Optical flow

Optical flow is the estimation of independent motion at each pixel and generally involves minimizing the brightness or color difference between corresponding pixels summed over the image. Equation 2.3 is the motion estimation using sum of squared differences (SSD):

$$S_{SSF-OF}(\{u_i\}) = \sum_i [I_1(x_i + u_i) - I_0 x_i]^2 \tag{2.3}$$

Since the number of variables $\{u_i\}$ is twice the number of measurements, the problem is underconstrained. The two classic approaches to this problem are to perform the summation locally over overlapping regions (the patch-based or window-based approach) or to add smoothness terms on the $\{u_i\}$ field using regularization or Markov random fields (SZELISKI, 2010).

Some classical methods were implemented and are available at OpenCV tool, as for example the Lucas-Kanade method and Dense Optical Flow. Also, there are important datasets that allow us to test, evaluate and compare different algorithms of optical flow, as for example the Middlebury Optical Flow Dataset [1] and the MPI Sintel Flow Dataset [2].

2.3.4 Disparity estimation using deep learning

Stereo matching aims to estimate the disparity map between a pair of rectified images. Disparity refers to the horizontal distance between a pair of corresponding pixels in the left and right images. The disparity ($disparity$) of a pixel can be converted to depth (Z) by $Z = Bf/disparity$. In this way, the accuracy of the depth improves with the prediction of the disparity. Disparity estimation is an essential task for computer vision applications, such as autonomous driving, 3D reconstruction, and robot navigation.

Significant progress of deep learning in the field of computer vision has been also extended to geometric problems such as stereo matching. Currently, deep learning based approaches have reached state of the art performance in most stereo disparity benchmarks, such as Middlebury (SCHARSTEIN *et al.*, 2014), KITTI (GEIGER; LENZ; URTASUN, 2012), ETH3D (SCHöPS *et al.*, 2017), and Argoverse (CHANG *et al.*, 2019).

However, scenarios of the real world not only require state of the art algorithms but also real time inference, and domain adaptation. Argoverse has released a stereo dataset in different lighting and weather conditions containing images at ten times the resolution relative to images in the KITTI dataset, and 16 times as many training frames, making it a much larger and more challenging dataset. The ground-truth depth is derived from LiDAR point cloud accumulation. The Argoverse team has organized the 2022 stereo challenge to motivate researchers to test algorithms that work well, run fast, and generalize to new scenes at the same time in the Argoverse stereo dataset.

2.4 Final Considerations

Convolutional neural networks (CNNs) have enjoyed great success, particularly in computer vision tasks like object classification and detection. This close relationship between CNNs and computer vision has fostered mutual progress in both fields. In this work, we leverage these advancements in both computer vision and deep learning to tackle the challenging task of mapless navigation. In this work, computer vision, computational intelligence, and machine learning, become our key tools. Subsequent chapters will showcase how we apply these technologies to address the problem of autonomous driving.

[1] Available at: <http://vision.middlebury.edu/flow/>
[2] Available at: <http://sintel.is.tue.mpg.de/>

CHAPTER 3

LITERATURE REVIEW AND RESEARCH WORK

As mentioned before, there are different approaches to developing autonomous systems, depending on the technologies used and how the components are structured. This section offers a brief and critical review of modular, end-to-end, and hybrid software architectures. Table 1 summarizes the related works.

3.1 Modular Navigation Architecture

A modular architecture typically organizes software components in a hierarchical manner based on specific criteria. Each group, referred to as a layer, operates at a distinct level of abstraction and provides services to its adjacent layers. This structure follows a descending order of abstraction, where higher-level layers handle more abstract tasks, while lower-level layers manage finer controls in the architecture. For example, following the hierarchical navigation stack proposed by Paden et al. (PADEN *et al.*, 2016), the initial layer is responsible for road and lane-level route planning, determining the roads and lanes the vehicle must follow to reach its destination. Subsequently, a behavior layer makes tactical decisions for the vehicle during navigation, such as interactions with other traffic participants, adherence to traffic rules, and high-level maneuver choices (e.g., lane following, lane change, U-turn, overtaking, and emergency stop). In addition to the route, this layer also receives perception information, including obstacle position and velocity, and traffic light status. Once the behavior is determined, a motion planning layer calculates short-term, feasible, and collision-free trajectories, that are translated into low-level commands, such as throttle, brake, and steering, by low-level controllers within the control layer (KATRAKAZAS *et al.*, 2015).

This design pattern is widely utilized in autonomous systems and has demonstrated notable success in both industrial and research vehicle applications (TENG *et al.*, 2023). Key

Table 1 – Summary of Related Works.

Primary Study	Name	Type	Layers or Methods	Sensors or Inputs
(TAŞ *et al.*, 2018)	BerthaOne	Modular	Sensing, Perception, Planning, Control, HMI, Communication	Radar, LiDAR, GNSS, IMU, Cameras, Stereo Camera, V2V, V2I
(FAN *et al.*, 2018)	Apollo	Modular	Perception, Prediction, Planning, Control, HMI, Guardian, Localization, HD-Map, CANBus	Radar, LiDAR, GNSS IMU, Cameras, V2X, Untrasonic
(AUTOWARE, 2024)	Autoware	Modular	Sensing, Perception, Planning, Control, HMI, Localization, Map	Radar, LiDAR, GNSS, IMU, Cameras, Ultrasonic
(WEI *et al.*, 2013)	CMU	Modular	Hardware, Perception, Mission Planning, Behavior Generation, Motion Planning	Radar, LiDAR, GNSS IMU, Cameras, V2V, V2I, Wheel speed sensor
(JO *et al.*, 2015)	A1	Modular	Sensor Interface, Autonomous Driving Algorithm, Actuator Interface, Development Interface	Lasers, GNSS IMU, Cameras
(SHAO *et al.*, 2023)	ReasonNet	End-to-End	ResNet (2D backbone), Transformer Encoder, PointPillars (3D backbone), GAT, CNN, MLP, GRU	Images (left, front, right and rear), LiDAR
(SHAO *et al.*, 2022)	InterFuser	End-to-End	ResNet (2D and 3D backbone), Transformer, GRU, MLP	Images (left, front, and right), LiDAR
(WU *et al.*, 2022)	TCP	End-to-End	ResNet (2D backbone), MLP, GRU	Image (front), Speed, High-level command, Goal
(CASAS; SADAT; URTASUN, 2021)	MP3	End-to-End	CNN blocks (3D backbone), CNN (Map decoders), Probabilistic Reasoning	LiDAR, High-level command
(XIAO *et al.*, 2022)	Multimodal CIL	End-to-End	CNN (2D backbone), MLP	Image (front), speed, Depth Image (front), High-level command
(ZHANG *et al.*, 2022)	MMFN	End-to-End	ResNet (2D and 3D backbone), VectorNet (3D backbone), GAT, MLP, GRU	LiDAR, Radar, Map, Goal
(CAI *et al.*, 2020)	PMP-net	End-to-End	ResNet (2D and 3D backbones), Attention Mechanism, Gaussian Mixtute Model, MLP	Image (front), LiDAR, Radar, Map, Position, Goal, Speed
(VITELLI *et al.*, 2022)	SafetyNet	Path Planning	PointNet (trajectory backbone) Transformer Encoder, MLP	Historical trajectory, Map, Goal
(SONG *et al.*, 2018)	IVGG LSTM	Path Planning	DCNN, CNN LSTM, MLP, VGG	Image (front, left and right rear view mirror)
(MORAES *et al.*, 2020)	DeepPath	Path Planning	WideResNet38, DeepLabV3, MLP	Image (front), Position
(WANG *et al.*, 2021)	CNN RawRNN	Path Planning	ResNet50 (2D backbone) LSTM, MLP	Image (front)
(HU *et al.*, 2022)	ST-P3	Path Planning	EfficientNet (2D backbone), DeepLabV3 (head), Attention Mechanism, GRU, MLP	Image (front - center, left, right; back - center, left, right), High-level command

Human-Machine Interface (HMI) ; Vehicle to Vehicle (V2V) ; Vehicle to Infrastructure (V2I); Vehicle to Anything (V2X) - ; Convolutional Neural Network (CNN); Multi-Layer Perceptron (MLP); Gated Recurrent Unit (GRU); Graph Attention Network (GAT); Deep Cascaded Neural Network (DCNN); Long Short-Term Memory (LSTM).
Source: Research data.

studies typically adopt similar layers, including sensing, perception, planning, control, and human-machine interface as fundamental components (TAŞ *et al.*, 2018; FAN *et al.*, 2018; JO *et al.*, 2015; WEI *et al.*, 2013; AUTOWARE, 2024). However, there are variations, such as communication between vehicles (e.g., Vehicle-to-Anything - V2X) (TAŞ *et al.*, 2018), Health Management Systems focusing on hardware and software component monitoring, diagnosis, prognosis, and fault recovery (FAN *et al.*, 2018; WEI *et al.*, 2013; JO *et al.*, 2015), behavior or mission planning (WEI *et al.*, 2013; JO *et al.*, 2015; AUTOWARE, 2024), and mapping strategy

(TAŞ *et al.*, 2018; JO *et al.*, 2015; AUTOWARE, 2024). In the latter case, Wei et al. proposed an alternative to the hierarchical navigation stack. This alternative organizes components of the behavior and control layers in parallel order, based on the functioning of Advanced Driver Assistance (ADAS) system. According to the authors, this approach enhances the flexibility of the autonomous system, enabling it to operate at a higher frequency compared to alternative methods.

Nevertheless, the parallel design faces challenges in coordinating components during complex tasks and maneuvers due to asynchronous communication. Additionally, both parallel and hierarchical approaches share issues related to error propagation between components and the intricate management of components with the increase in vehicle autonomy. This occurs because, as autonomy increases, the vehicle also performs more tasks (e.g., maneuvers) and encounters a broader range of traffic scenarios. Therefore, the number of components adversely affects the system's performance.

3.2 End-to-End Autonomous Driving

End-to-End is a navigation approach where neural networks and deep learning models are trained to map sensory input (e.g., images or point clouds) to control outputs (e.g., steering, throttle, brake) or intermediate outputs (e.g., trajectory segment). This eliminates the need for manual feature tuning in modular navigation pipelines. The advantage lies in leveraging deep learning generalization to simplify and enhance the adaptability of navigation stacks across different traffic scenarios. There are various approaches to classifying end-to-end models, ranging from the degree of the deep learning model's involvement in tasks to the technology applied. In the former category, methods range from pure end-to-end architectures, where the deep learning models handle the entire mapping and decision-making process, to hybrid approaches that integrate different algorithms, such as probabilistic models, control theory, fuzzy inference systems, etc. The latter category divides models based on techniques, such as imitation learning and reinforcement learning.

In addition to the presented taxonomy, studies on end-to-end navigation also focus on input representation aspects and the model design. This includes considerations in the number of cameras (e.g., single or multi-camera setups) (SHAO *et al.*, 2023; SHAO *et al.*, 2022; WU *et al.*, 2022), methods for 3D data representation (e.g., point cloud or Bird's Eye View images) (SHAO *et al.*, 2023; SHAO *et al.*, 2022; CASAS; SADAT; URTASUN, 2021; ZHANG *et al.*, 2022), sensor fusion and multimodality (e.g., different sensors and feature fusion methods) (XIAO *et al.*, 2022; ZHANG *et al.*, 2022; CAI *et al.*, 2020), interaction with traffic agents (e.g., interaction graphs or grid maps) (SHAO *et al.*, 2023; ZHANG *et al.*, 2022), deep learning technologies (e.g., transformers, graph neural networks, deep reinforcement learning, attention mechanisms, generative models, etc.) (SHAO *et al.*, 2023; SHAO *et al.*, 2022; ZHANG *et al.*, 2022; CAI *et* *al.*, 2020), decision-making within the network (e.g., high-level commands input or inference) (WU *et al.*, 2022; CASAS; SADAT; URTASUN, 2021), and the accuracy or feasibility of the output (e.g., using standard controllers to estimate final outputs or filtering the output of the deep learning model) (SHAO *et al.*, 2023; CASAS; SADAT; URTASUN, 2021; CAI *et al.*, 2020).

In summary, end-to-end models primarily rely on RGB images and LiDAR-generated point clouds, represented in 3D as points, voxels, or Bird's Eye View (BEV) images. While the ResNet network is commonly used for feature extraction from images and BEV (SHAO *et al.*, 2022; WU *et al.*, 2022; CAI *et al.*, 2020; ZHANG *et al.*, 2022; CASAS; SADAT; URTASUN, 2021), some studies also explore the use of specialized deep learning models for 3D data, such as PointPillars (SHAO *et al.*, 2023) and VectorNet (ZHANG *et al.*, 2022). Two significant challenges for deep learning models include multimodality fusion and how to handle tactical decisions within the network (or when incorporating decisions from external sources). In the former case, early-fusion and middle-fusion approaches are noteworthy (XIAO *et al.*, 2022), they often involve attention mechanisms or concatenation of feature vectors. In the latter case, tactical decisions (i.e., high-level commands) can be treated as an input modality (WU *et al.*, 2022) or a conditional variable (CASAS; SADAT; URTASUN, 2021; XIAO *et al.*, 2022), particularly in approaches exploring multiple-expert designs. However, both pure and hybrid end-to-end navigation methods still face challenges related to the lack of transparency and explainability in decision-making and the requirement for extensive training data.

3.3 Data-driven Path Planning

Autonomous vehicles rely on path planning algorithms to navigate through dynamic and complex environments. Data-driven approaches have gained prominence in recent years, representing a shift from traditional rule-based methods (XU *et al.*, 2021). In data-driven path planning, algorithms leverage machine learning techniques to learn collision-free paths from large datasets (REDA *et al.*, 2024). These datasets typically include information from various sensors, historical driving experiences, and diverse environmental conditions. Similar to end-to-end navigation architectures, data-driven path planning also inherits the adaptability features from deep learning models, which make them able to plan under diverse road geometry and traffic scenarios.

Techniques for data-driven path planning typically emphasize the representation of spatial and temporal features. However, to address the challenges of dynamic driving scenarios, they also consider the representation of traffic rules, interaction among traffic participants, output trajectory smoothness and comfort, high-level commands (e.g., maneuvers), and variations in road geometry. Spatial features are commonly derived from frontal camera images or Bird's Eye View (BEV) projections, using CNN-based networks for feature embedding (SONG *et al.*, 2018; MORAES *et al.*, 2020; WANG *et al.*, 2021; HU *et al.*, 2022). Some works also use the historical trajectory of the ego-vehicle and surrounding agents (VITELLI *et al.*, 2022). Temporal features are traditionally addressed by recurrent neural networks (e.g., GRU and LSTM) (SONG *et al.*, 2018; WANG *et al.*, 2021; HU *et al.*, 2022), although recent studies have explored the application of Transformer networks (VITELLI *et al.*, 2022). Semantic and abstract data, such as traffic rules and high-level commands, are integrated as feature vectors or conditional variables (HU *et al.*, 2022). Finally, ensuring trajectory smoothness typically involves the application of a post-processing algorithm or the penalization term in the loss function (HU *et al.*, 2022). Nevertheless, methods employing machine learning for path planning face challenges in terms of transparency and explainability. Moreover, there is room for improvement in addressing global planning, high-level commands, managing dangerous and unexpected driving scenarios, and ensuring dynamic and kinematic feasibility of planned trajectories.

3.4 LRM Lab - fourteen years of research in Autonomous and Intelligent Vehicles

Within the LRM Mobile Robotics Laboratory[1] at ICMC/USP, where this work was conducted, several projects are being developed in the area of autonomous mobile robots, with an emphasis on projects involving small and medium-sized robots used in indoor environments and research projects on autonomous vehicles for outdoor environments. The autonomous vehicles under development are capable of navigating urban environments and even semi-structured environments, such as those in agricultural applications.

This thesis is directly related to the CaRINA project, Intelligent Robotic Car for Autonomous Navigation, which is under development at LRM-ICMC/USP. This project, initiated in 2010, aims to develop an intelligent autonomous vehicle capable of navigating urban environments without the need for a human driver. The CaRINA project currently has two experimental research platforms, both of which are already automated and capable of performing autonomous navigation. The objectives of the CaRINA project include reducing the number of accidents on streets and highways, increasing mobility for the elderly and people with special needs, and enhancing overall traffic efficiency. In the past, the CaRINA I (KLASER; OSóRIO; WOLF, 2014) electric car was able to travel more than 1.0 km in autonomous mode on the USP campus, and the CaRINA II (FERNANDES *et al.*, 2014) vehicle traveled more than 20 km without a driver in urban environments [2].

In addition to the CaRINA project vehicles, the LRM Lab also developed an autonomous truck in partnership with Scania (LRM-CROB/USP project in partnership with Scania Latin America)[3]. The autonomous truck was designed to use radar perception and stereo vision as

[1] CaRINA Project – website: <http://www.lrm.icmc.usp.br/carina/>
[2] Available at: <http://www.lrm.icmc.usp.br/web/index.php?n=Port.Midia>
[3] Available at: <http://www.lrm.icmc.usp.br/web/index.php?n=Port.ProjSTruck>

its main obstacle detection devices and has been presented on several occasions to various companies and the media. The obstacle detection system of the Scania truck was developed during the author's master's degree (ROSERO, 2017), which also involved creating a system for the fusion of LiDAR, radar, and computer vision for use on these platforms (ROSERO; OSóRIO, 2017). Subsequently, a similar solution was developed together with the Vale company for mining applications (CALDAS *et al.*, 2023).

Much other research related to machine learning, computer vision, control, decision-making, and simulation has been developed at LRM Lab, though not all of it has been integrated and tested in the CaRINA project. Our work aims to contribute to the CaRINA project by providing a framework that allows for the integration and testing of new algorithms for autonomous driving in a unified framework, first in simulation using the CARLA simulator and later on the real platform. Our stack is currently ready to work with the CARLA simulator using ROS for transparent future use on the CaRINA platforms.

To evaluate the performance of our developments since 2019, we have participated in challenges related to autonomous driving, particularly the CADCH, where we won the first edition (ROSERO *et al.*, 2020).

In 2023, four members of our laboratory joined the LRM team, working in different areas: Iago Gomes (agent architecture and map), Carlos Braile (localization), Junior Silva (control), and Luis Rosero (architecture, perception, navigation, End-to-End driving). Under the direction of Prof. Dr. Denis Wolf and Prof. Dr. Fernando Osório, we win first place in the Sensors track and second place in the Map track.

3.5 Final Considerations

The study of related works on modular architectures, end-to-end learning, and data-driven path planning presented in this chapter provides us with context and highlights the advantages and disadvantages of each approach. This understanding allows us to propose a novel hybrid architecture for autonomous driving that combines the best of each approach. Additionally, the past experiences of our researchers at the LRM Lab serve as the basis for improving previous work. We have developed a framework that integrates prior research from the LRM Lab and incorporates new methods in all areas of autonomous driving. This framework supports new perception, control, and decision-making modules, as well as modular, end-to-end, and hybrid architectures.

In this context, we address the three forms of autonomous driving described in this section. Firstly, we propose a modular pipeline, inspired by previous work, following a hierarchical structure, similar to others in the literature and also inspired by an earlier architecture proposed in our group. This serves as baseline and teacher for our learning-based methods

On the other hand, we implement models using both single and multimodal sensor inputs. Our approaches differ from others because we integrate tactical decisions (high-level commands) as graphical information within the input structure.

Additionally, we propose an early multi-sensor fusion structure, transforming point clouds (LiDAR, stereo) to BEV space and using it as input to a neural network. To our knowledge, fusion in bird's-eye view space with that specific structure has not been proposed. Additionally, we explore multimodal fusion combining camera and bird's-eye perspectives.

Furthermore, we propose a multitask learning method for autonomous driving that uses a CNN architecture for disparity estimation. We add a head for path and velocity estimation. To our knowledge, there is no methods in the literature leveraging disparity as an auxiliary task for autonomous driving. Considering the above, this research introduces a hybrid autonomous vehicle architecture that integrates modular pipelines with data-driven path planning, offering a comprehensive comparison of these approaches. Our modular and hybrid architectures were developed and evaluated in the 2023 CARLA Autonomous Driving Challenge (CADCH), securing 1st and 2nd place in the SENSORS and MAP tracks, respectively.

Finally, as evident in the reviewed literature, many algorithms and methods for perception, decision-making, or full end-to-end autonomous driving do not run in real time with limited resources, as they use complex neural network architectures that, despite having good accuracy, may not be applicable to real-time scenarios in many cases. This research aims to use simple and effective architectures for real-time operation, enabling their implementation in real autonomous platforms such as CaRINA 2.

CHAPTER

4

A MODULAR PIPELINE FOR AUTONOMOUS DRIVING

An autonomous system requires several components and its architectural design provides an abstract view of the system operation and organization. In a layered architecture, the components have public and well-defined communication interfaces through which they exchange information with other components. This characteristic enables the definition of a common architecture for all tracks in this challenge, through adjustments of few components for the maintenance of the same communication interface. This strategy reduces the time spent on the development of the agents, and enables the evaluation of the autonomous navigation performance with different sensors and algorithms for a specific task.

Figure 4 shows the general software architecture designed for all agents of LRM team in the 2023 CARLA Autonomous Driving Challenge (ROSERO *et al.*, 2024). The name *"CaRINA Agent"* is used to refer to this architecture throughout this Thesis. The layers of the architecture are *sensing*, *perception*, *map*, *risk assessment*, *navigation*, *control*, and *vehicle*. Robotic Operating System (ROS)[1] supported the communication interface between components with Publish/Subscribe pattern for messages passing (QUIGLEY *et al.*, 2009).

4.1 Map and planning

A map is an essential component enabling the autonomous vehicle to execute its tasks safely and efficiently, storing diverse information, beneficial for various components of the autonomous system (BRUMMELEN *et al.*, 2018). This includes the road geometry description for path planning and the topological representation of roads and intersections, commonly referred to as the road network, used for route planning. In addition to static object positions, navigable areas, positions of traffic signs and lights, traffic rules, and semantic information

[1] Available at: <https://www.ros.org/>.

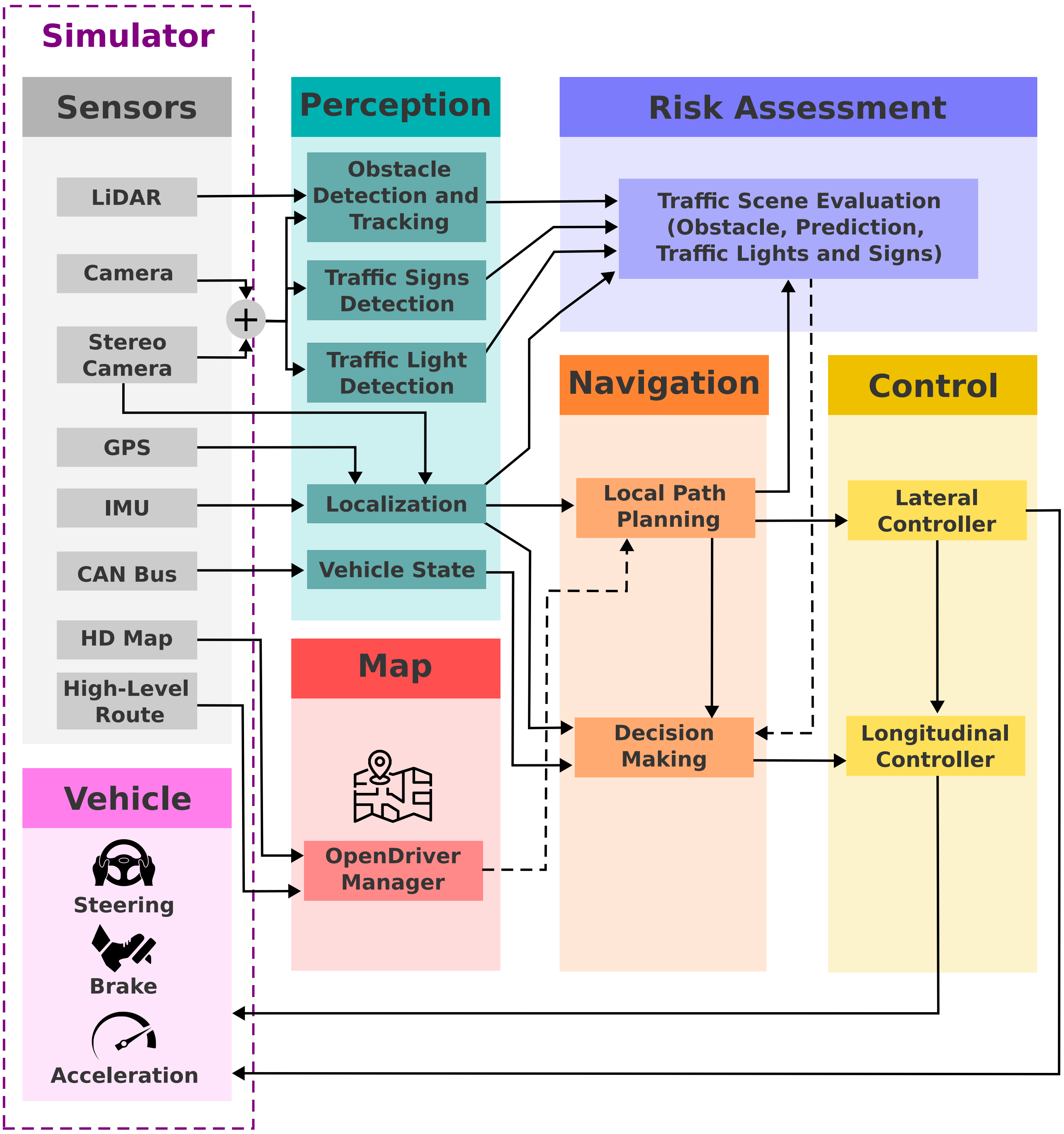


Figure 4 – General design of the proposed Modular architecture.

Source: Elaborated by the author.

related to the road. In this architecture, we employed the OpenDRIVE (OPENDRIVE, 2023; DIAZ-DIAZ *et al.*, 2022) map standard to assist the navigation and perception components.

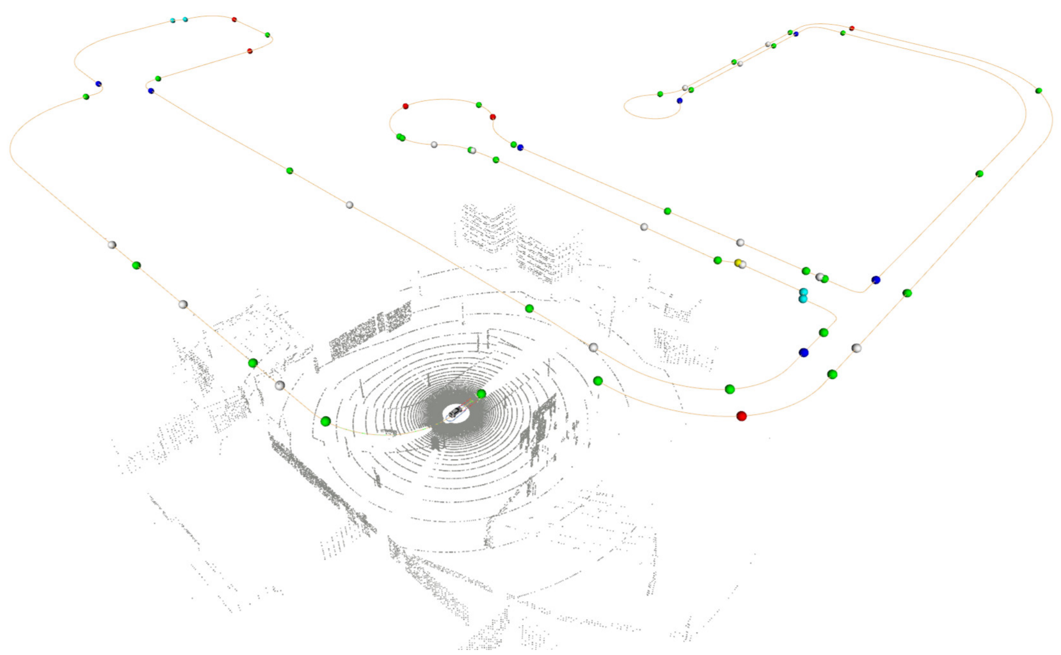


Figure 5 – OpenDRIVE Map. The dots represents high-level commands with red (turn left), blue (turn right), green (keep lane), and white (go-straight).

Source: Elaborated by the author.

4.1.1 OpenDRIVE

The OpenDRIVE is an open format to describe road networks, using XML version 1.0, which is able to represent the road geometry as well as the context information of roads that may influence the behavior of vehicles driving in it, such as traffic signs, traffic lights, and the type of roads and lanes (e.g. highway and sidewalk) (DUPUIS; STROBL; GREZLIKOWSKI, 2010; DIAZ-DIAZ *et al.*, 2022). The format description is built on a hierarchical structure with four main elements: *header*; *road*; *junction*; and, *controller*. Figure 5 shows the visualization of the OpenDRIVE map after being parsed by the *map manager* in the architecture.

The *header*, is the first element of the description and holds metadata of the map, such as the name or a geographic reference for transformations between the Cartesian and Geodesic coordinate systems. The *road* element encompasses geometry description and additional properties (e.g., elevation profile, lanes, and traffic signs). The properties such as traffic signs are placed with respect to the distance to the initial point of the lane, using a local coordinate system. Besides that, *roads* can connect directly or through intersections using the *junction* component, preventing ambiguities in connections. Lastly, a *controller* is employed for signalized junctions or other road elements imposing control on the vehicle.

4.1.2 Path Planning

Path planning is responsible for determining a feasible and optimal path from a starting point to a destination in a given environment. This planned path considers factors such as road geometry, static and dynamic obstacles, vehicle physical constraints, and criteria like time, speed, fuel efficiency, or safety. Various algorithms, including rule-based, gradient-based, graph-based, and optimization methods, can perform these tasks. In addition, recent studies have explored the integration of machine learning into path planning. Moreover, the task can be divided into two steps: global planning and local planning. Global planning estimates a reference path, considering static features (e.g., road geometry and static obstacles) and the intended route. Local planning adjusts this global reference path based on dynamic variables in the traffic scene, such as dynamic obstacles.

In the modular navigation pipeline strategy, *global reference planning* incorporates a local speed profile based on the dynamic scene, along with local lane-change planning due to traffic events. The *global reference planning* involves sparse waypoints representing the intended route. Subsequently, lane-level localization determines the corresponding lane and road ID for each waypoint using the OpenDRIVE map manager. This information enables the system to estimate a lane-level route, identifying the roads and lanes the vehicle must traverse to reach its destination. Finally, segments of the reference path, each spanning 50 meters, are published in the ROS ecosystem based on the vehicle's speed and current position.

4.2 Perception and Sensing

The autonomous driving systems proposed in this thesis relies on two types of sensors for robust environmental perception: a stereo camera and a light detection and ranging (LiDAR) sensor. Our proposed perception system, depicted in Figure 6, adopts a multi-sensor fusion approach to achieve accurate and robust object detection in 2D/3D images and point clouds. This approach employs distinct detection modules and their fusion, each capitalizing on the strengths of different sensor modalities:

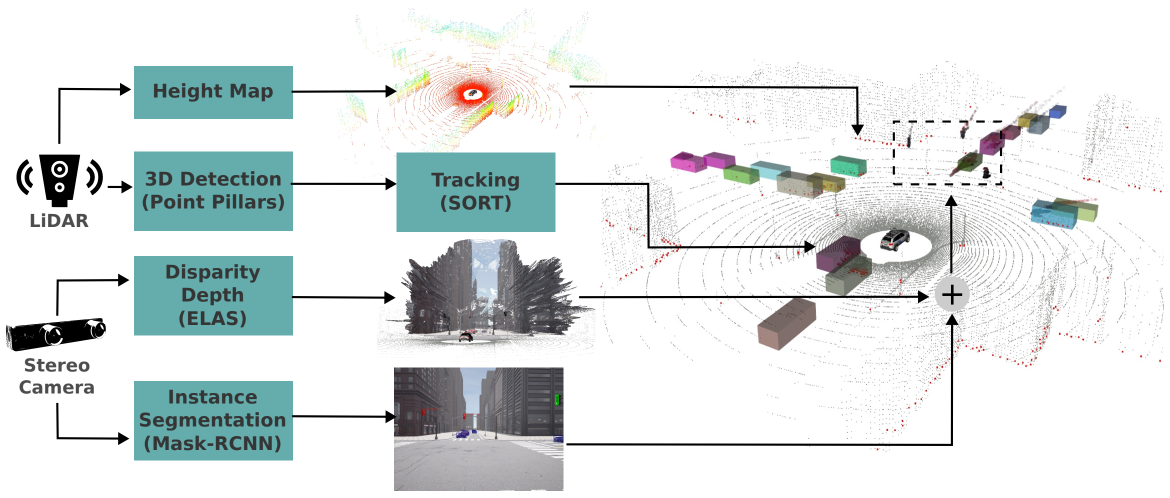


Figure 6 – Perception module.

Source: Elaborated by the author.

4.2.1 Height maps as obstacle detectors

As a basic obstacle detector, we employ a height map generated from the LiDAR point cloud, similar to the one presented in (THRUN *et al.*, 2007). This method analyzes height differences in a grid, identifying obstacles exceeding a threshold. Grid cells with no such differences are deemed part of a plane. We use a polar grid map and a 20 cm as threshold. This detection mechanism serves as a backup for emergency situations. The first branch in Figure 6 depicts a height map-based perception process, where points are assigned colors based on their height to create a visual representation of the surrounding environment's topography then grid cells within that contain points exceeding the predetermined vertical distance threshold are identified. Finally, points situated within these designated grid cells are marked in red to clearly indicate potential obstacles. Our obstacle detector using height map in a polar grid is open source and available online[2].

4.2.2 Instance segmentation

For object detection, we employ Mask R-CNN (HE *et al.*, 2017a), a powerful instance segmentation algorithm that extracts coordinates of bounding boxes and masks for each object instance detected in the image. Our system categorizes these objects into eight categories: car (including vans, trucks, and buses), bicycle (including motorcycles), pedestrian, red traffic light, yellow traffic light, green traffic light, stop, and emergency vehicle.

4.2.3 Fusion with stereo camera

Since the image used for instance segmentation is also the left image of the stereo camera, we leverage this for 3D detection and classification. The bounding box coordinates and pixel mask of each object instance detected in 2D are mapped to corresponding 3D points in the organized point cloud. This 3D point cloud inherits the color and the image's row, column structure but expands it with 3D/depth information. This fusion process creates an RGBD point cloud with color and 3D/depth information for each object instance, enabling accurate classification and positioning.

The third branch, Figure 6, illustrates an example of a point cloud constructed from the stereo camera, while the fourth branch shows an example of instance segmentation corresponding to the same scenario. Finally, the fusion of these two branches enables the detected objects to be mapped onto the RGBD point cloud, providing a comprehensive visualization of their positions within the 3D environment.

While this method can detect both static and dynamic objects, it is not the primary system detector in our architecture due to its non-real-time operation with a maximum of 5 frames per second. Nevertheless, the detailed information it provides on traffic light states, unavailable from

[2] Available at: <https://github.com/luis2r/Polar-Height-Map>.

LiDAR, justifies its inclusion despite the latency. Dynamic object detection (cars, bicycles and pedestrians) in this fusion serves as a backup for emergencies, and we currently do not track these objects.

4.2.4 3D detection in point clouds (dynamic objects):

For 3D detection in LiDAR point clouds, we leverage the PointPillars algorithm (LANG *et al.*, 2019). This algorithm provides regression of 3D bounding boxes of objects and their orientation related to the ego vehicle. PointPillars demonstrates commendable performance in real-world autonomous driving scenarios. Its pillar representation retains valuable spatial information while maintaining computational efficiency, a critical factor for real-time applications. Moreover, it effectively leverages the strengths of LiDAR data, including its ability in handling occlusions and performing reliably in diverse lighting conditions.

4.2.5 Tracking

The perception stack (in point clouds) detailed in this section, which supplies inputs to the risk assessment module responsible for determining Finite State Machine (FSM) graph states, consists of two integral components: (i) pose estimation (bounding boxes) and (ii) multi-object tracking (MOT) modules. Having a stable and precise state estimation, both for the ego-vehicle and dynamic objects in the surroundings, is important to transition across the states.

Regarding the **multi-object tracking** module, we based on the one proposed by (BEWLEY *et al.*, 2016). This approach, known as Simple Online and Realtime Tracking (SORT), divides the tracking task into three sub-tasks: detection, data association, and state estimation.

As detailed earlier, objects are detected in the LiDAR frame using PointPillars. This detection model differs from the ones adopted originally by the SORT tracker. The detected objects' bounding boxes are projected into the world frame using the ego-vehicle estimated pose, and their respective poses are matched in the data association step. At this point, we keep the SORT tracker data association in the 2D space, in which we compute the intersection over union (IoU) of the top-down projection of the bounding boxes. Our assumption is that two dynamic objects do not overlap in the X-Y plane.

As for the state estimation, we use a Kalman Filter-based approach, in which our goal is to estimate the 3D position of the bounding box center (x, y, z), 3D dimensions of the bounding box (s_x, s_y, s_z), the yaw angle ψ, and the 3D velocities $(\dot{x}, \dot{y}, \dot{z})$ of the tracked object in the International System of Units and in the global reference frame. This state space differs from the SORT paper state, in which the estimated state is performed in the pixel space. Also, notice that the vehicle pose estimation is important in this step, as we are tracking in the global reference frame.

As we represent this state vector as $S_{obj} = (x, y, z, s_x, s_y, s_z, \psi, \dot{x}, \dot{y}, \dot{z})^T$, we need to define both the state propagation and observation model matrices.

The state propagation we adopted assumes constant linear velocity between detections so that we can propagate the position using the propagation matrix F from Equation 4.1.

$$F = \begin{bmatrix} I_{3x3} & 0_{3x4} & \Delta T \cdot I_{3x3} \\ 0_{4x3} & I_{4x4} & 0_{4x3} \\ 0_{3x3} & 0_{3x4} & I_{3x3} \end{bmatrix}, \tag{4.1}$$

where I represents the identity matrix, 0 the zero-filled matrix, and ΔT represents the period between predictions. The subscripts indicated with MxN represent the matrix number of rows M and columns N, respectively.

Regarding the observation model matrix, since we obtain directly the 3D position, dimensions, and yaw angle from our detection module, our observation matrix is defined by $H = I_{7x10}$.

The third branch in Figure 6 illustrates detection and tracking in 3D, with LiDAR serving as input to the 3D detector. In our case, the 3D detections of point pillars are fed into the SORT algorithm, which we have modified for tracking. Ultimately, for visualization, each tracked object is assigned a distinct color.

4.2.6 Prediction

4.2.6.1 Linear Prediction

Our system employs a prediction-based approach to ensure safe navigation by anticipating the movements of surrounding objects. This approach utilizes a simple motion model based on the object's current speed, as estimated by the tracking system and this model assumes constant velocity for each object, providing a first-order approximation of their trajectories.

The prediction formula for a linear motion model is defined by the Equation 4.2.

$$x(\Delta t) = x_o + v\Delta t \tag{4.2}$$

Where x_o is the current pose, v the actual velicity of the surrounding object and Δt represents the time interval for prediction (5 seconds). Finally $x(\Delta t)$ is the predicted pose in the interval Δt.

While more complex prediction models exist, opting for a simple linear model ensures computational efficiency. This linear motion model for future pose prediction demands fewer computational resources, making them suitable for real-time applications in autonomous driving.

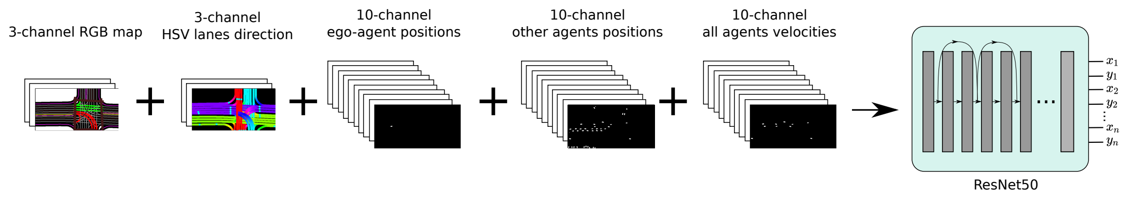

3-channel RGB map
3-channel HSV lanes direction
10-channel ego-agent positions
10-channel other agents positions
10-channel all agents velocities
x_1
y_1
x_2
y_2
x_n
y_n
ResNet50

Figure 7 – CNN-MultiRegressor architecture.

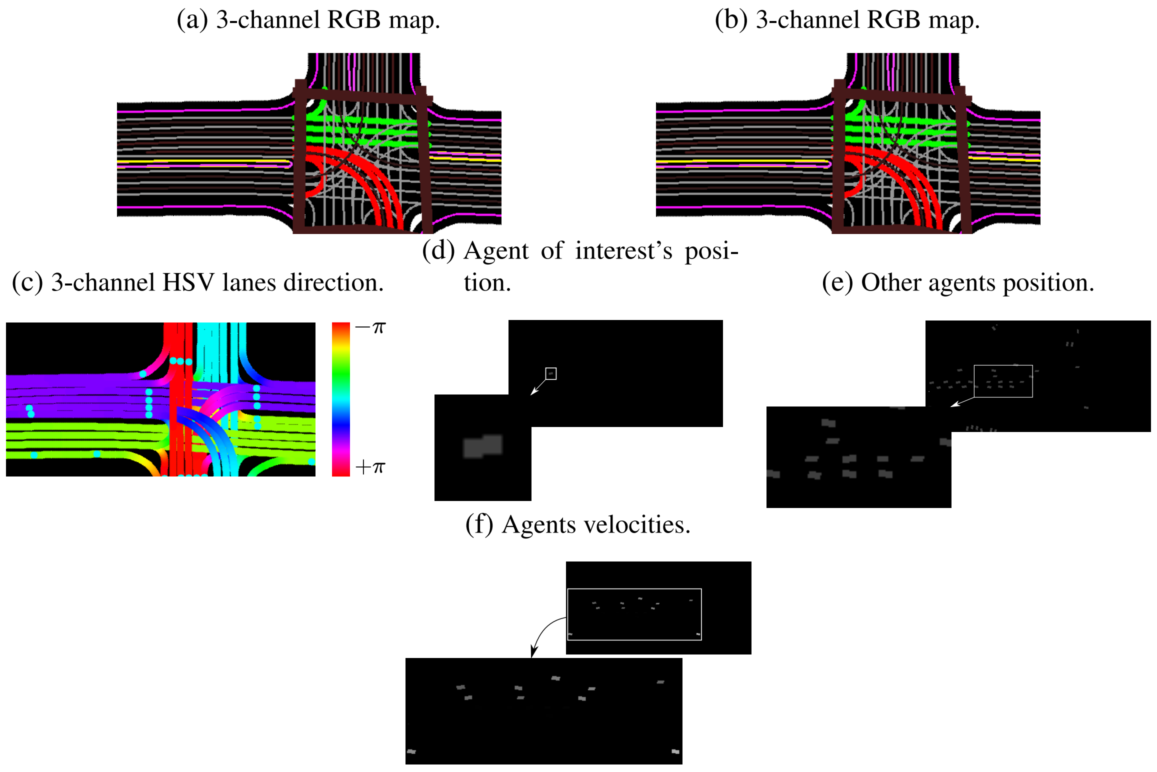

(a) 3-channel RGB map.
(b) 3-channel RGB map.
(c) 3-channel HSV lanes direction.
$-\pi$
$+\pi$
(d) Agent of interest's position.
(e) Other agents position.
(f) Agents velocities.

Figure 8 – ResNet-50 inputs. Each channel is fed with different rasterized images from the data in order to track the evolution of the scene.

Source: Research data.

4.2.6.2 *Motion Prediction Using Deep Learning and Multimodal Data Fusion*

This section presents a novel approach to motion prediction, framing it as a machine learning (ML) problem where traffic participant (TP) behavior is learned from data. To capture the diverse behaviors of different TPs (cars, cyclists, pedestrians) and leverage the detailed structure of the environment (high-definition map), substantial data is required for reliable and safe navigation. Fortunately, large-scale datasets like the Lyft Level 5 Open Data and the Waymo Open Motion Dataset facilitate ML application in motion prediction.

We propose a predictor named CNN-MultiRegressor, inspired by previous work (HOUSTON *et al.*, 2020), for predicting future trajectories. This pipeline feeds a ResNet-50 backbone (HE *et al.*, 2016b) with rasterized data from the Waymo Open Motion Dataset (agent states and semantic map), as shown in Figure 7. It predicts future trajectories for an agent over the next 8 seconds and employs a loss function to forecast 6 joint trajectories with their respective probabilities. Our source code and models are publicly available[3].

Data Processing and Rasterization: The pipeline, tested in the Motion Prediction Challenge, includes a module that transforms Waymo dataset data into a raster format suitable for convolutional networks. The neural network is a customized ResNet-50 for regressing 6 joint trajectories. We utilize a Bird's Eye View (x and y coordinates only) from the dataset, with data relative to the world coordinate system and potentially large variations between scenes. To create rasters, we transform map points, current and past poses of all agent cuboids, to the agent of interest's coordinate system. Subsequently, we convert coordinates to the image coordinate system (Figure 7).

Channel Inputs and Feature Encoding: The map is projected as a 3-channel RGB image (Figure 8a), including all map features, center lanes, road edges, stop signs, traffic lights, etc., in their current state. Figure 8c exemplifies a rasterized lane orientation map. Each map feature point's orientation is represented by a unit direction vector with its value mapped to an HSV color map between $-\pi$ and $+\pi$. Figure 8d displays rasters for the agent of interest's poses. The current and past 9 poses are drawn on grayscale images (Figure 8e). Ten additional grayscale images encode the current state and past 9 poses for surrounding agents. Finally, another 10 grayscale images encode agent speeds (Figure 8f), where intensity is proportional to speed at each time step. Each raster is 256x128, and all are stacked into a single 256x128x36 tensor representing the scene for feeding into the ResNet-50 backbone.

Network Architecture and Training: We use a ResNet-50 backbone with a 20% dropout for the last dense layer and no data augmentation during training. We employ the Adam optimizer with a step learning rate scheduler, starting at 0.01 for the first two epochs and reaching 0.001 by the last epoch.

For training, we randomly select one agent per scene in batches of 32 across 3 epochs, amounting to 768,000 frames per epoch. The ResNet-50 was trained at three resolutions: 0.5cm/pixel for larger map coverage, 0.125m/pixel for capturing short-range details, and finally 0.25m/pixel, which was also used for inference.

Evaluation and Metrics: Our implementation was evaluated in the Waymo Prediction Challenge 2020, which used various metrics like minADE, minFDE, Miss Rate, and Overlap Rate, with the main metric being mAP (described on the challenge website). mAP defines precision and recall based on threshold-based predictions as specified in the challenge.

In regression tasks, common loss functions like Mean Squared Error, Mean Squared Logarithmic Error, and Mean Absolute Error are typically used, but these are suited for unimodal outputs. For certain tasks and datasets, using the main evaluation metric as the loss function for training can be beneficial. However, mAP is non-differentiable and cannot be directly optimized with gradient descent methods (BROWN *et al.*, 2020). Therefore, we adopted the negative

[3] CNN-MultiRegressor code: <https://github.com/luis2r/CNN-MultiRegressor.git>.

Table 2 – Results: Motion Prediction Challenge

Method Name	minADE	minFDE	Miss Rate	Overlap Rate	mAP
Anonymous610	1.0387	1.5514	0.1573	0.1779	**0.3281**
CNN-MultiRegressor	**0.8257**	1.7101	0.2735	**0.1640**	0.1944
Waymo LSTM baseline	1.0065	2.3553	0.3750	0.1898	0.1756

Source: Research data.

Table 3 – Results: Interaction Prediction Challenge

Method Name	minADE	minFDE	Miss Rate	Overlap Rate	mAP
Waymo LSTM baseline	1.9056	5.0278	0.7750	0.3407	0.0524
CNN-MultiRegressor	2.5850	6.3697	0.8910	**0.3076**	0.0341

Source: Research data.

likelihood, which was the main evaluation metric in the Lyft Motion Prediction Challenge[4], represented by Equation 4.3, where we predict c confidences for k hypotheses.

$$Loss = -log\sum_{k} e^{log(c^k)-\frac{1}{2}\sum_t(\bar{x}_t^k - x_t)^2+(\bar{y}_t^k - y_t)^2} \tag{4.3}$$

Exploring Multimodal Feature Fusion with LSTM We further explored a variant architecture that incorporates Long Short-Term Memory (LSTM) networks for multimodal feature fusion. This architecture aims to leverage the strengths of both convolutional neural networks (CNNs) and LSTMs: CNNs efficiently extract spatial features from static elements like the map and agent locations and LSTMs effectively capture temporal dependencies in sequences of agent poses, encoding past movements and potentially predicting future behaviors.

The proposed architecture utilizes the same CNN-based pipeline as before to extract spatial features from static and current data and employs an LSTM network to encode sequences of past agent poses for each traffic participant (TP), capturing temporal dynamics.

We perform feature fusion by combining the extracted spatial and temporal features from both encoders. This fusion was achieved using the concatenation method. This conduct a multimodal prediction using the fused features to predict future trajectories and potentially interaction probabilities, similar to the original CNN-MultiRegressor structure. for training we employ the same loss function for multimodal prediction.

Table 2 shows the results obtained in the test set for the Waymo Motion Prediction Challenge, and Table 3 shows the results obtained for the Waymo Interaction Motion Prediction Challenge.

Real-time Limitations and Further Development: While conceptually promising, this architectures currently does not meet real-time requirements for our autonomous driving pipeline

[4] Available at: <https://github.com/lyft/l5kit/blob/master/competition.md>

due to the computational overhead of rasterization.

By addressing these limitations, we aim to integrate the benefits of multimodal feature fusion with LSTMs into a real-time capable system for enhancing motion prediction and interaction understanding in future iterations of our autonomous driving approach.

4.3 Risk assessment

Our autonomous driving system employs a dedicated Collision Risk Assessment (CRA) module to continuously evaluate potential threats posed by both dynamic (cars, pedestrians, bicycles) and static surrounding objects. This module integrates current and future positions of surrounding objects, obtained from the previous stage, with the ego vehicle's planned path for risk assessment.

• **Zoned Risk Evaluation:** The planned path is divided into two zones, each reflecting different risk levels based on distance from the ego vehicle: a High Risk Zone (0-4m) and a Moderate Risk Zone (4m-40m). For risk evaluation: The path ahead is divided into two zones, each carrying different risk levels based on Euclidean distance. These zones are corridors created from the waypoints of the planned path (essentially buffer zones extending 40 meters ahead of the ego vehicle). The width of these corridors matches the width of the ego car.

Any object (static or dynamic) whose current position intersects either zone is considered a potential collision threat. The intersection point of predicted trajectories with the ego vehicle's path is also considered. We assume that our lateral MPC controller guarantees that the ego car will pass exactly through these corridors.

The identified potential points of collision and object's information, including type, distance, and predicted trajectory, is reported to the decision-making module for determining appropriate speed adaptations. Figure 9 visually illustrates the risk assessment process, showcasing three points as examples. Note that other surrounding objects and their predicted trajectories are currently ignored unless they enter the relevant risk zones or directly influence the planned path.

4.4 Decision-Making

The decision-making module utilizes a synchronous Moore Finite State Machine (FSM) to orchestrate actions based on inputs from the Collision Risk Assessment (CRA) module. The FSM employs a binary encoding scheme for inputs as shown in Table 4.

The FSM comprises four key states, each governing specific speed control behaviors:

• **Drive State (S1)**: No obstacles impede the vehicle's progress. Target speed is set to a maximum of 8.8 m/s (31.68 km/h).

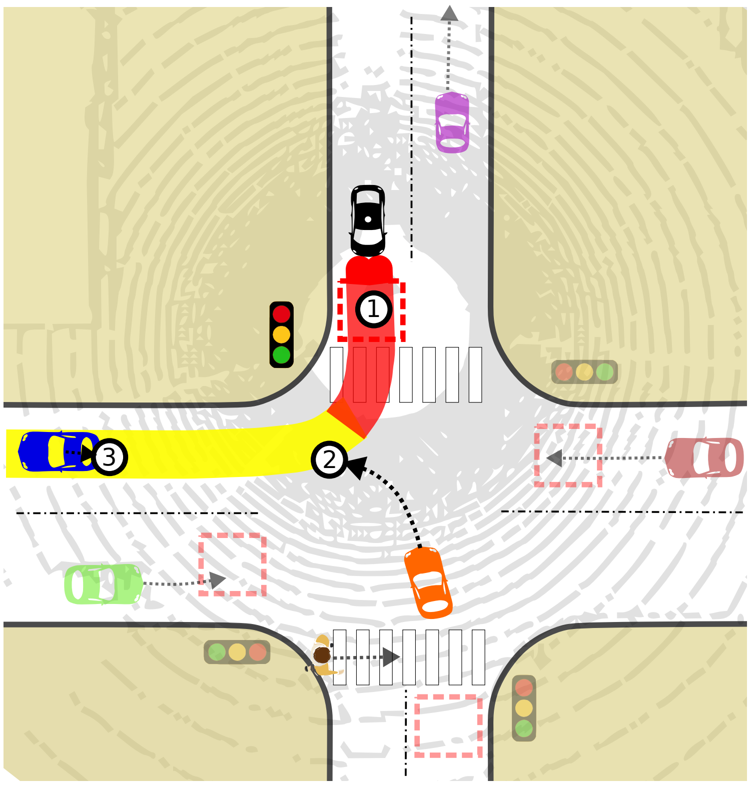


Figure 9 – Risk assessment. *Point 1* - Zone of influence of a red traffic light; *Point 2* - The predicted trajectory of another car intersects the ego vehicle's path in the yellow zone; *Point 3* - A parked car within the yellow zone is identified as a potential obstacle but receives lower priority compared to threats in the red zone.

Source: Elaborated by the author.

Table 4 – FSM inputs.

Input	Description
00	No obstacles detected, indicating a clear path ahead.
01	An obstacle is being tracked, requiring speed adjustments to maintain safe following distances.
10	A red traffic light is ahead, necessitating a controlled stop.
11	A stop sign is detected, also demanding a full stop.

Source: Elaborated by the author.

- **Follow the Leader State (S2)**: The CRA reports an obstacle (static or dynamic) ahead of the ego vehicle, triggering dynamic speed adjustments. Speed is adjusted based on distance and time to collision (TTC), calculated using the ego vehicle's current speed and distance to the obstacle.

- **Red Light State (S3) and Stop Sign State (S4)**: These states mirror the "Follow the Leader" logic, utilizing TTC to achieve controlled stops at designated locations. The vehicle decelerates smoothly, ensuring compliance with traffic rules and safety.

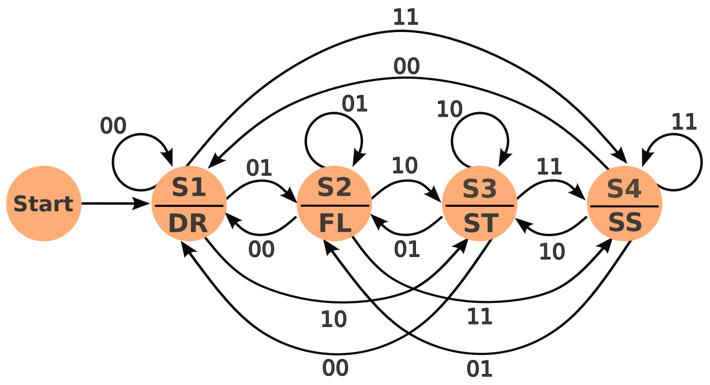


Figure 10 – State transition diagram for the Moore Finite State Machine used in our decision-making module.

Source: Elaborated by the author.

Table 5 – State transition table (based on hand-crafted rules)

Present State	Next State				Output / Description
	Input=00	Input=01	Input=10	Input=11	
S1	S1	S2	S3	S4	DR / Drive
S2	S1	S2	S3	S4	FL / Follow the Leader
S3	S1	S2	S3	S4	ST / Stop Red Traffic Light
S4	S1	S2	S3	S4	SS / Stop Sign

Source: Elaborated by the author.

Figure 10 depicts the state transition diagram, visually representing the FSM's logic. The decision-making module employs a straightforward yet effective FSM structure for robust decision-making. Speed control strategies adapt dynamically to varying conditions, ensuring safe and efficient navigation. The module seamlessly integrates with other components of the autonomous driving system, including perception and control modules. The FSM operates synchronously at 10 Hz, aligning with sensory data capture rates.

The current approach uses the actual velocity of the ego vehicle to calculate TTC, after which velocity adjustments are made to prevent collisions. This method is simple, fast, suitable for quick estimations, easy to implement, and computationally efficient. -However, it assumes constant velocity and ignores potential future trajectory changes. The TTC formula is: $TTC = Distance/RelativeVelocity$

4.5 Localization

In order to perform the ego-vehicle **pose estimation**, we fused the relative transforms obtained using an odometry source (in our case, visual-inertial odometry - VIO), the IMU orientation, and the GNSS position using an Extended Kalman Filter (EKF) approach, as in the Figure 11. The inputs of our stack are the camera image, the IMU orientation, the GNSS coordinates, and the sensor calibration (external reference frames' relative transformation). The

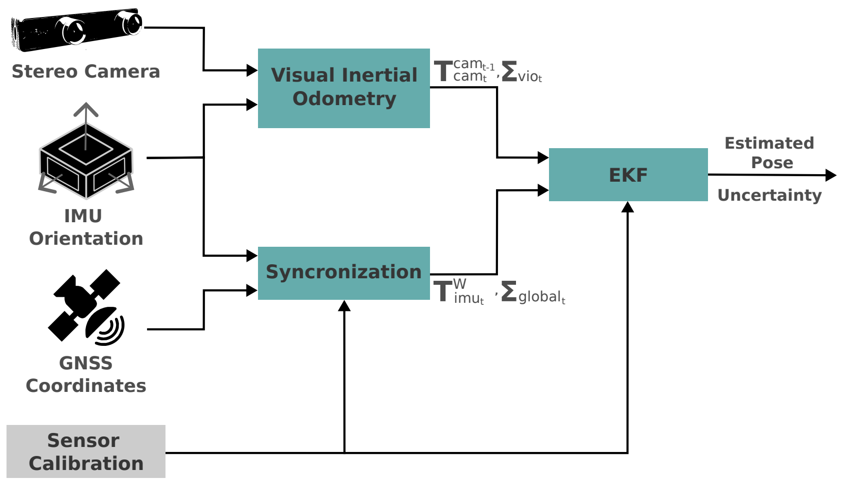


Figure 11 – The pose estimation stack used in our perception module.

Source: Elaborated by the author.

output elements of the pose estimation stack are estimated pose and its uncertainty.

The VIO estimation is responsible for estimating $T^{cam_{t-1}}_{cam_t}$, which represents the pose transformation matrix of the current camera frame w.r.t. the previous, and the estimation uncertainty covariance matrix, Σ_{vio_t}.

While the GNSS is responsible for providing the global geographic coordinates, the IMU provides the linear acceleration, angular velocity, and 3D orientation at a higher frequency. We then synchronize both the 3D orientation and global coordinates in order to provide $T^{W}_{imu_t}$, which represents the transformation matrix of the IMU frame w.r.t. the world frame, and its uncertainty, Σ_{global_t}. In our case, the IMU and the GNSS are represented by the same reference frame, but we left them illustrated in the diagram for the sake of clarity. Also, the geographic coordinates provided are then converted to a plane projection coordinate system.

The input poses, $T^{cam_{t-1}}_{cam_t}$ and $T^{W}_{imu_t}$, and the sensor calibrated, are then provided to the EKF and then converted to a common reference frame internally. The goal is to estimate the 6DoF pose of the agent frame w.r.t. the world frame, $S_{agent} = (x, y, z, q_x, q_y, q_z, q_w)^T$, where: (x, y, z) are the global coordinates, easting, northing, and altitude, respectively; and (q_x, q_y, q_z, q_w) represents the four components of the quaternion that represents our agent's orientation. For each relative pose received, $T^{cam_{t-1}}_{cam_t}$, the EKF performs a system prediction, which implies accumulating drift until a global pose, $T^{W}_{imu_t}$, is received and the state update is performed.

We emphasize that this pose estimation module is also modular, so that the back end (in this case, the EKF), the methods used for estimating the relative transforms, and the source of the global pose estimation do not need to be the same as the ones we used in this project.

In practice, for estimating the relative pose transformations using VIO, we used the RTabMap ROS implementation[5]. RTabMap is known for its estimation robustness and its full

[5] Available at: <https://github.com/introlab/rtabmap_ros>.

functionalities are widely used in SLAM applications. As for the EKF implementation, we used the GTSAM implementation[6]. While GTSAM is known for implementing solutions using factor graphs, it also implements a very convenient interface for representing pose transformations and implements the 3D pose Extended Kalman filter off-the-shelf. Finally, our localization stack is open source and available online[7].

4.6 Control Systems

The *control layer* generates steering, throttle, and brake commands to keep the agent on the planned trajectory. This goal is achieved through two closed-loop control mechanisms that receive desired vehicle trajectory information from the navigation layer's decision-making and local path planning modules. These modules set desired trajectory and velocity into the agent's action space. The closed-loop controls translate reference values into actual control actions for braking, throttle, and steering, which are then sent directly to the simulator for execution.

For longitudinal control, the decision-making module (FSM) calculates the desired agent's velocity, which is used to compute the final velocity. A Proportional-Integral-Derivative (PID) controller then ensures the agent follows this desired reference.

4.6.1 Lateral Control (MPC)

The lateral control, which generates the steering signal, is managed by the Model-Based Predictive Control (MPC), in which a cost function is optimized along a predefined time horizon H, thus resulting in a sequence of actions, one for each time step Δt. The immediate action is executed and the process is restarted in the next time step, leading to a receding horizon optimization.

Develop an optimization problem for Model Predictive Control (MPC) aimed at improving autonomous driving performance. The goal is to minimize the deviation from a desired path while ensuring smooth and safe vehicle operation under varying road conditions and constraints.

The constraints defining the vehicle's motion model are essentially non-holonomic. Car-like robots can assume positions on the 2-D plane, different headings and steering angles, thus adding up to four degrees of freedom. However, it poses the following two kinematic constraints: a) the vehicle is allowed to move only forward and backward and b) the steering angle is bounded (KATRAKAZAS *et al.*, 2015). Therefore, the actual car motion and the planning trajectory can be different when the planner neglects dynamics factors.

Figure 12 shows a bicycle model used to represent car-like vehicles, which are characterized by Ackerman steering geometry (DUDEK; JENKIN, 2010), moving with longitudinal

[6] Available at: <https://gtsam.org/doxygen/4.0.0/a03631.html>.
[7] Available at: <https://github.com/cabraile/LRM-Localization-Stack-2023>.

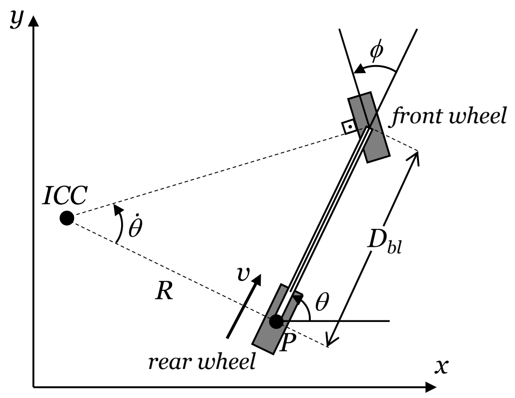


Figure 12 – Geometry of a bicycle model. The ICC is determined from the length of the bicycle body D_{bl} and the steering angle ϕ.

Source: Elaborated by the author.

velocity v. The front wheel is able to turn and gives the steering angle ϕ, whereas the rear wheel is always aligned with the bicycle body. According to Fig. 12, θ represents the heading of the vehicle, and P is the guiding point controlled so as to follow the assigned path. The intersection between the lines that pass through the rear and front wheels axes provides the Instantaneous Center of Curvature (ICC). The distance between ICC and P represents the radius of curvature R. The curvature of the vehicle is given by $\kappa = 1/R$.

By considering that the wheels roll without slipping, only the kinematic equations can be considered and the lateral dynamic effects can be neglected (LIMA *et al.*, 2015). Therefore, the considerations made so far result in the following kinematic model (FRAICHARD; SCHEUER, 2004)

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\kappa} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \kappa \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tau, \tag{4.4}$$

where $\tau = \dot{\phi}/(D_{bl}\cos^2\phi)$. The motion constraints are added to the optimization problem by means of the third power of Δt on the basis of Eq. 4.4 (OBAYASHI; UTO; TAKANO, 2016), where v is computed by the decision-making module (considered constant in the optimization). The cost function is defined as the sum of the quadratic differences between the decision variables and the reference path,

$$L_{ref} = C_x\frac{1}{2}(x - x_{ref})^2 + C_y\frac{1}{2}(y - y_{ref})^2 + C_\theta\frac{1}{2}(\theta - \theta_{ref})^2 + C_\kappa\frac{1}{2}(\kappa - \kappa_{ref})^2, \tag{4.5}$$

and also, the quadratic of τ

$$L_\tau = C_\tau\frac{1}{2}(\tau)^2, \tag{4.6}$$

where C_x, C_y, C_θ, C_κ and C_τ are cost weights manually tuned. The chosen parameters are shown in Table 6. The optimization is performed by *Python* library *scypy.optimize*.

Table 6 – Non-linear MPC parameters.

Δt	H	C_x	C_y	C_θ	C_κ	C_τ
1 s	4 s	5	5	10	100	10

Source: Elaborated by the author.

4.7 Final Considerations

In this chapter, we present a modular pipeline for autonomous driving. It's worth noting that the prediction module, named CNNMultiRegressor, is not currently integrated into our architecture. This is because it does not meet the real-time processing requirement we've established for this work (10Hz). However, we include it here as it may inspire future works utilizing deep learning and rasterized data for prediction. Instead, we employ a simpler prediction method that may not yield optimal results and can be replaced in the future.

We have used well known methods and proposed others for the different layers and modules. The advantage of our framework lies in its modular structure, which enables seamless component replacement in the future, facilitating easy substitution with existing or emerging techniques in all areas (detection, tracking, localization, decision-making, prediction) to enhance performance or for new research.

The code developed for this architecture is organized into packages, simplifying the isolation and adaptation of individual modules for use in other architectures. All code developed for the modular pipeline is publicly available online[8].

It's important to mention that while this code can potentially be adapted for real autonomous driving platforms in the future, the released version is currently tailored to function solely with the CARLA simulator and leaderboard for research purposes.

Automatic control systems for autonomous driving represent a broad and dynamic field, with notable advances but This thesis does not focus on this particular area and not to conduct an exhaustive study or detailed modeling of the simulated vehicle's dynamics.

We utilize a simplified vehicle model (bicycle model) and well-established controllers: an MPC controller for the steering angle and a basic PID for cruise control, both with manually tuned parameters. We see considerable opportunities to enhance our control by exploring other techniques, such as nonlinear model predictive control, robust linear quadratic regulators, model-free control, and quadratic programming.

Additionally, evaluating driving is pertinent, as the current control system likely exhibits deficiencies in this area. This evaluation can help advance the area establishing a benchmark metric to assess and develop control systems closer to those real-world implementations, offering improved efficiency while considering a comfort parameter.

[8] Available at: <https://github.com/lrmicmc/CaRINA-agent>.

A list of videos of our modular pipeline agent going through different scenarios in simulated environments for the track SENSORS in the CARLA leaderboards in validation routes is available online[9].

[9] Available at: <https://www.youtube.com/playlist?list=PLcT95Tv_ZrmJPRu-NhNeIoL56oAjojHOO>

CHAPTER

5

END-TO-END AUTONOMOUS DRIVING

Many current end-to-end approaches for autonomous vehicle motion planning rely on behavioral cloning. Figure 13 depicts two modes of end-to-end learning. In the first mode, the input is directly mapped to actions such as steering, braking, and acceleration. In the second mode, the inputs are mapped to intermediate affordances, which are then interpreted by a low-level controller. In this work, we opt for the second option due to its greater adaptability to different vehicles and its ease of implementation.

Behaviour cloning involves imitating pre-recorded driving behaviors or expert demonstra-

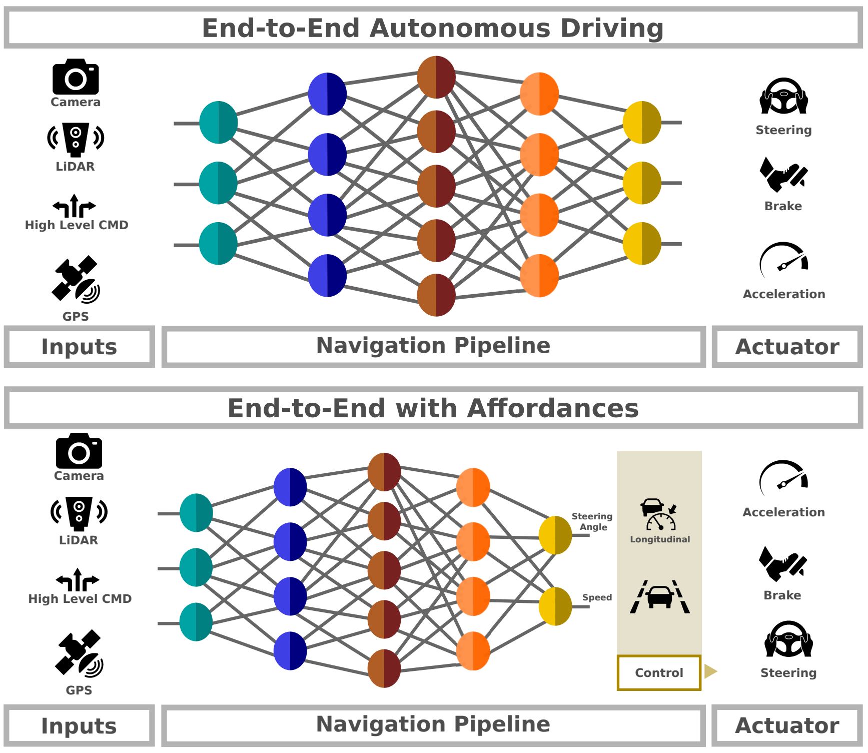


Figure 13 – End-to-end modes.

Source: Elaborated by the author.

tions, where the motion planning problem is modeled as a function $p = f(z)$ that takes an input z and generates a sequence of waypoints $p = \{p_0, ..., p_t\}$. Each point p_i comprises coordinates (x_i, y_i, v_i) representing the future poses and speeds that the vehicle must reach over time. The velocity v_i can also be calculated based on the distance between each waypoint (x_i, y_i) and the previous waypoint (x_{i-1}, y_{i-1}). Alternatively, in control cloning, a set of actions $a = \{a_0, ..., a_n\}$, where a_i represents a value (steering angle, acceleration, brake), is imitated at each future instant of time i. These approaches rely solely on the present step and make decisions based on current state information, ignoring potential future interactions with the environment. This can lead to unsafe or suboptimal behavior, especially in complex scenarios.

These approaches typically wait until the entire planning is executed or new inferences are available to adjust speeds and trajectory. New paths and velocities are only generated at specific intervals, potentially causing delayed reactions to obstacles or other agents. This issue is apparent in datasets like *Nuplan*, where models employing behavioral planning perform well in scenarios without interaction with other vehicles but experience decreased performance when interacting with other agents during execution.

To address these limitations, we propose a behavior cloning approach that models the motion planning problem as a combination of path planning and decision-making. A function $(w, v_0) = f(x)$ receives sensory information x and outputs a set of n equally spaced waypoints w, and a single instantaneous velocity value v_0, where $w = \{w_0, ..., w_n\}$. This signifies that in each step, we need a new value of velocity and we have a long path for following, eliminating errors in navigation.

This path format simplifies navigation and data collection in simulated and real mapped environments, where paths are pre-determined, and only the current vehicle pose and path clipping are necessary to collect the dataset (no future car poses are required). Moreover, this format aligns with traditional pipelines that track paths in a waypoints format equally spaced.

Given these considerations, we prioritize real-time inference, enabling new planning and speed calculation at each step to allow interaction with other actors and the environment, and if necessary, the rapid generation of a new plan. We employ shallower network architectures for improved real-time inference capabilities while maintaining high navigation performance in mapless scenarios.

5.1 Monocular camera for end-to-end driving

Our first end-to-end model uses a monocular camera as input. The RGB image is then encoded using a ResNet 18 backbone to extract features, finally a regression head regresses 401 values corresponding to 200 (x, y) points for the path and a v value for the desired vehicle velocity. For the input we use the RGB image fused with the high-level commands that allow us to avoid ambiguities at intersections. The high level commands are *turn right, turn left, go*

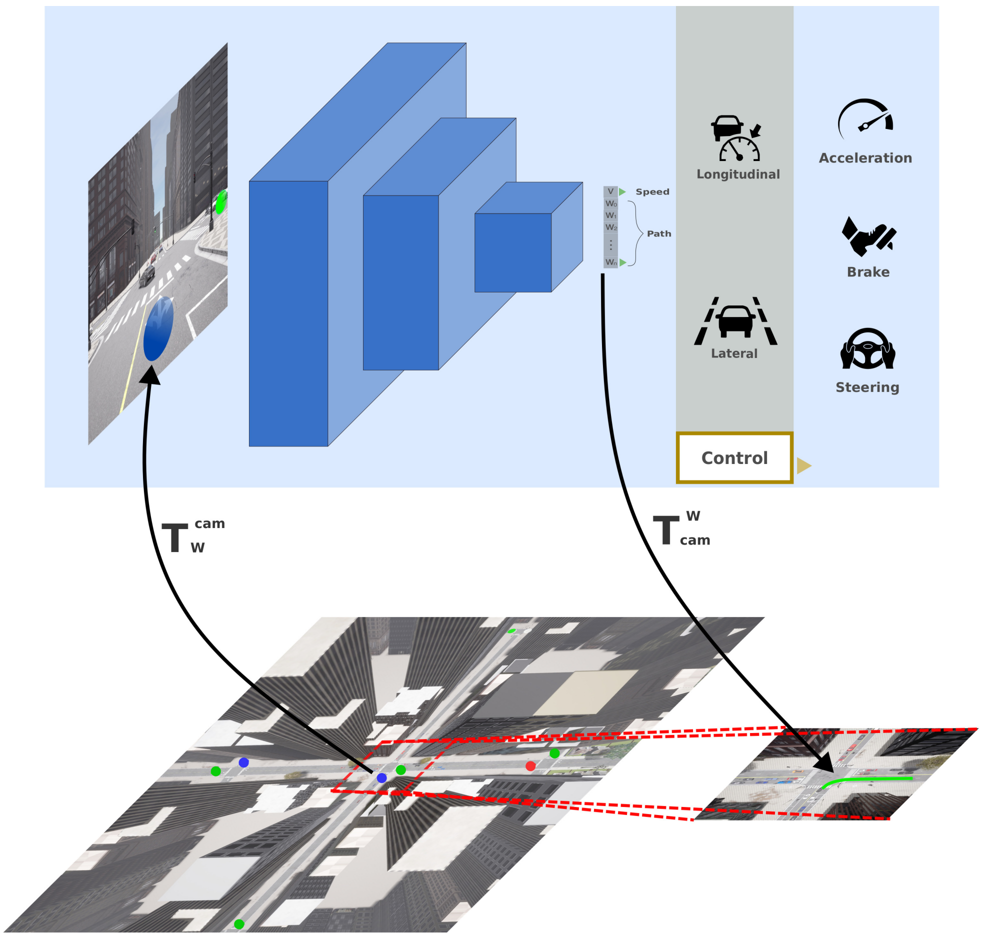


Figure 14 – End-to-end driving using only monocular camera and high level commands on intersections as inputs a CNN for regression of the path and the velocity.

Source: Elaborated by the author.

straight ,or follow the lane, we color code those commands and project them as color points on the RGB image. This commands have a position in the world coordinate system, we use the ego vehicle pose and the camera calibration to transform that point to the image. Figure 14 shows the point transformation process and the proposed architecture for end-to-end driving using a single monocular camera. For training we use the MSE error (Eq. 5.1).

$$L = MSE(y, \hat{y}) = \frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N} \tag{5.1}$$

Where y is the predicted path or desired speed and $\hat{y}$ is the ground truth. The final loss for training is a composed weighted loss as shown in Eq. 5.2.

$$L_f = W_w \times L_w + W_v \times L_v \tag{5.2}$$

Where L_w is the MSE loss for the path composed of 200 waypoints and L_v is the loss for the desired speed.

5.2 Stereo for end-to-end driving

This section presents a novel method for disparity estimation using deep learning in the context of autonomous driving. Our approach leverages the capabilities of the CARLA simulator to generate diverse and realistic depth and disparity datasets for training.

For end-to-end autonomous driving we build upon this disparity estimation algorithm and extend it to a multi-task learning framework. This method simultaneously predicts disparity, vehicle path, and speed. This integration enhances the navigation adding depth information without the necessity of a modular pipeline for obstacle detection.

This innovative approach has yielded good results, benefiting not only from the advantages of multi-task learning but also from the precise disparity estimation. By generating a point cloud and identifying hazardous obstacles more explicitly than traditional single-camera approaches, we have significantly enhanced obstacle detection capabilities.

5.2.1 PWC-Net for stereo disparity estimation

First we describe the proposed method for disparity estimation only. For this task we adapt a method for optical flow estimation named PWC-Net (SUN *et al.*, 2018). According to the authors, PWC-Net is a CNN model for optical flow that has been designed according to simple and well-established principles: pyramidal processing, warping, and the use of a cost volume. Cast in a learnable feature pyramid, PWC-Net uses the current optical flow estimate to warp the CNN features of the second image. It then uses the warped features and features of the first image to construct a cost volume, which is processed by a CNN to estimate the optical flow. PWC-Net outperforms optical flow methods on the MPI Sintel and KITTI 2015 benchmarks, running at about 35 fps on Sintel resolution (1024×436) images.

Considering two rectified RGB images coming from a calibrated stereo camera, we can have an epipolar line across the two images and the stereo matching is the pixel correspondence between the two images along the epipolar line in the horizontal direction on the x-axis. Thus, for this challenge we consider stereo matching as a special case of optical flow where disparities between the stereo pair can be modeled as optical flow on the x-axis of the image. So we can use models used for optical flow (x and y coordinates) to solve the disparity problem as optical flow only at the x coordinate along the epipolar line between the left and right image.

Figure 15 summarizes the key components of PWC-Net adapted for stereo estimation. Feature pyramid 1 and feature pyramid 2 correspond to learnable feature pyramids from a feature pyramid extractor feeded with the left and right RGB rectified images. A warping operation from the traditional optical flow approach is used as a layer in the network to estimate large motion. PWC-Net has a layer to construct a cost volume, which is then processed by CNN layers to estimate the flow (disparity). The warping and cost volume layers have no learnable parameters and reduce the model size. Finally PWC-Net uses a context network to exploit

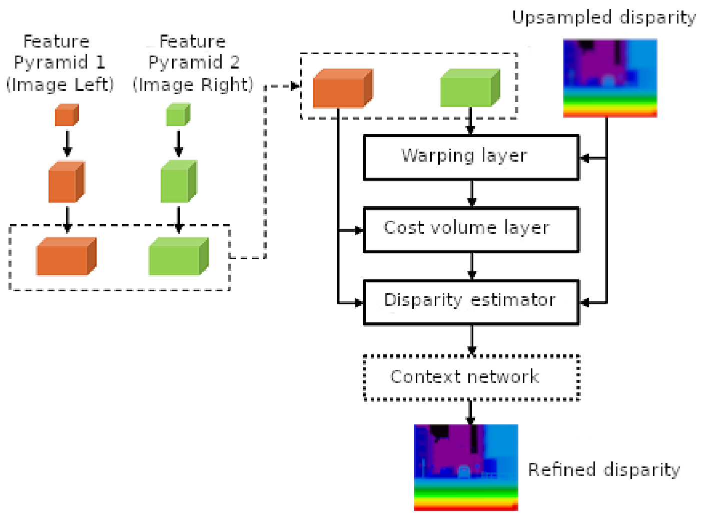


Figure 15 – PWC-Net for disparity estimation.
Adapted from (SUN *et al.*, 2018)

contextual information and refine the disparity.

5.2.2 Implementation

Our adaptation of PWC-Net for disparity estimation is implemented in the MMFlow framework (CONTRIBUTORS, 2021). MMFlow is an open source pytorch based toolbox that is a part of the OpenMMLab project. MMFlow is the first toolbox that provides a framework for unified implementation and evaluation of optical flow algorithms.

For training disparity estimation we create a dense synthetic stereo dataset from CARLA simulator (DOSOVITSKIY *et al.*, 2017). Then, we fine-tune the model using the Argoverse stereo dataset. For evaluation, metrics from the KITTI stereo challenge are adopted.

5.2.3 Synthetic disparity dataset

A dense synthetic stereo dataset was created using the CARLA simulator. We configure a simulated agent using a setup similar to the one used in the Argoverse stereo dataset for data collection. We use two RGB cameras at a height of 1.7m on the roof with a baseline (distance between the two cameras) of 0.2986 m. Each camera has a FOV=30 deg, the images have a size of 2464×2056 pixels.

In the same pose as the left camera we set a depth camera that provides raw data of the scene encoding the distance of each pixel to the camera (also known as depth buffer or Z-buffer) to create a depth map of the elements in the scene.

To create our dense stereo dataset we use the depth (Z) values and the baseline value in pixels to create disparity maps for each left RGB image as shown below.

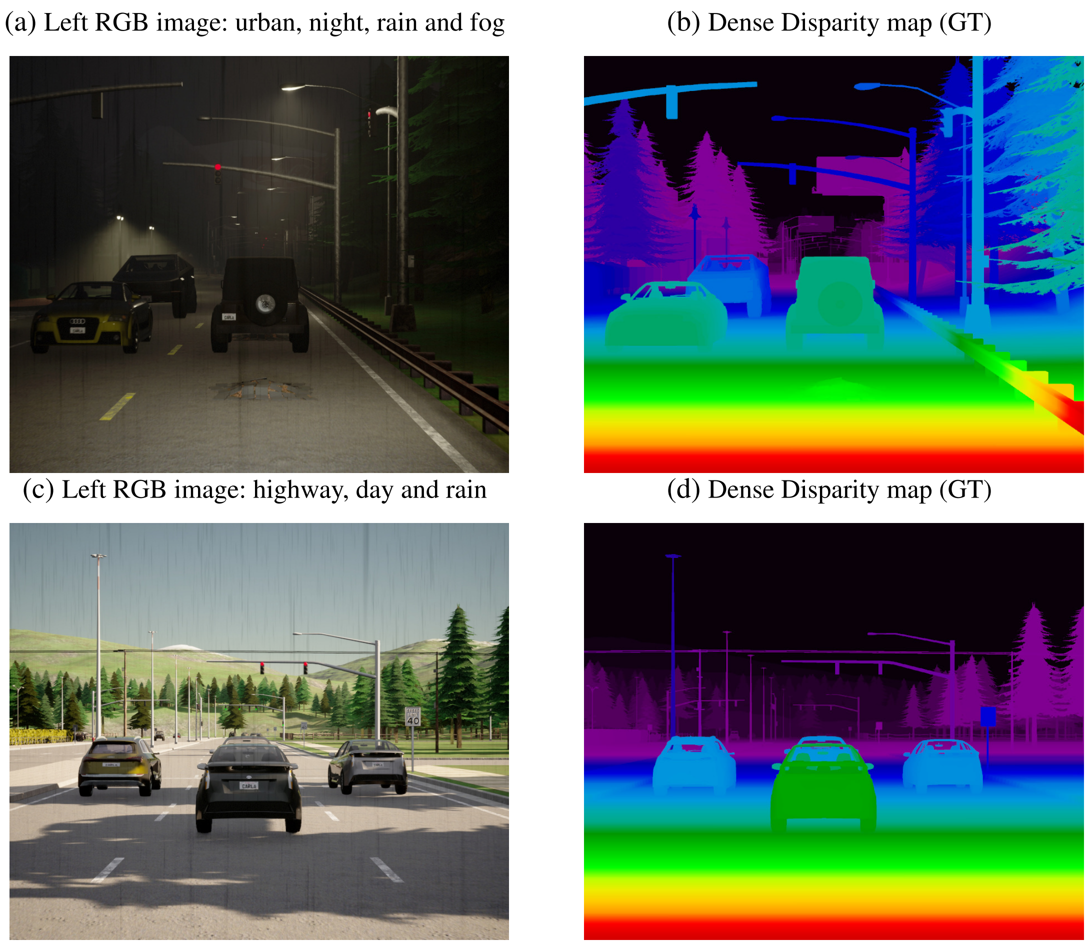

(a) Left RGB image: urban, night, rain and fog
(b) Dense Disparity map (GT)
(c) Left RGB image: highway, day and rain
(d) Dense Disparity map (GT)

Figure 16 – Dense synthetic stereo dataset generated using the CARLA simulator.

Source: Research data.

$$disparity = \frac{Bf}{Z} \tag{5.3}$$

Where B is the baseline and f is the focal length of the camera (calculated from FOV), Z is depth. We collected approximately one hundred thousand frames.

Our dataset includes urban, residential and highway environments, as well as different weather and lighting conditions including rain, fog, day, night, etc. Figure 16 shows two examples of left RGB images and their respective dense disparity map captured from the CARLA simulator.

5.2.4 Training

We use the same parameters used for training in (SUN *et al.*, 2018) and the same loss proposed in FlowNet (DOSOVITSKIY *et al.*, 2015). We use a search range of 4 pixels to compute the cost volume at each pyramid level. We first train the model using our synthetic stereo dataset using the S_{long} learning rate schedule introduced in (ILG *et al.*, 2017), Starting from 0.0001 and reducing the learning rate by half at 0.4M, 0.6M, 0.8M, and 1M iterations. Finally, we fine-tune the model using Argoverse stereo dataset using the S_{fine} schedule (ILG *et al.*, 2017). Batch size 4 was used for all the training process.

For data augmentation, we use a random crop (768 x 2432 patches). Inference is performed in full resolution.

5.2.5 Results in the Argoverse stereo benchmark

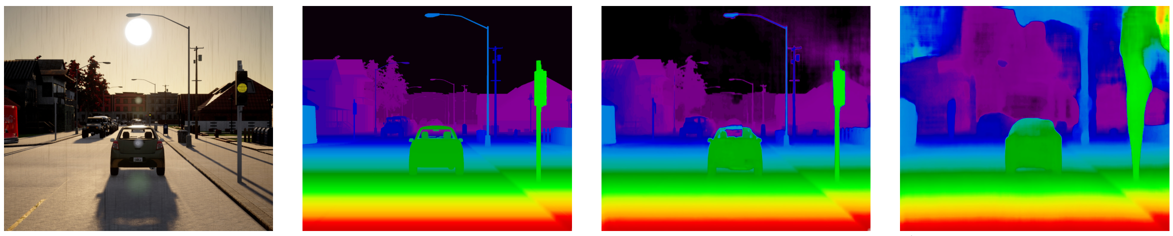
Figure 17 shows two frames: one for our synthetic dataset and the other taken from the validation set of the Argoverse stereo dataset. Figure 17a shows the left RGB image, Figure 17b represents the ground-truth (dense for our dataset and sparse for Argoverse stereo dataset) and then disparity results: first we show inference results for a PWC-Net model trained only on our synthetic dataset (Figure 17c) and finally inference results for the same model but with fine-tuning performed on the Argoverse stereo dataset (Figure 17d).

Note that the ground-truth disparity released together with the Argoverse stereo dataset is sparse (Figure 17b below) and many pixels in the background and foreground do not have ground-truth, mainly in the upper parts of the image, for example: the tops of buildings, traffic lights very close to the camera and lamps. However, PWC-Net trained only on our dataset correctly calculates disparity in the upper parts (Figure 17c above and below). This is because the disparity ground-truth of our dataset is dense and available at training time for all pixels in the image. In the same way as testing on the synthetic dataset, inference on the Argoverse stereo dataset (Figure 17d) also benefited from prior training on the synthetic dataset. Disparity in the upper parts in the Argoverse dataset is correctly estimated.

An important result of training on our synthetic stereo dataset is domain adaptation. The model trained only using synthetic data obtains very satisfactory results on the test set of the Argoverse as shown in Figure 17c. In these figures we can see that the shape of the objects and their edges are well defined and thin objects are correctly differentiated, for example, we can clearly see power cables and other thin objects, while when fine-tuning is performed on the sparse dataset these details tend to disappear.

Argoverse Stereo Competition server computes the percentage of bad pixels averaged over all ground-truth pixels, similar to the KITTI Stereo 2015 benchmark (MENZE; HEIPKE; GEIGER, 2015)(MENZE; HEIPKE; GEIGER, 2018) for all 1,094 test disparities from 15 log sequences.

The disparity of a pixel is considered to be correctly estimated if the absolute disparity error is less than a threshold or its relative error is less than 10% of its true value. Three disparity error thresholds are defined: 3, 5, and 10 pixels. The leaderboard ranks all methods according to the number of bad pixels using a 10 pixels threshold (all:10 is the main metric). We compare our results in Table 7 using all:10, fg:10, and bg:10.



(a) Left RGB image

(b) Ground-Truth

(c) Inference: PWC-Net trained only using our CARLA stereo dataset

(d) Inference: PWC-Net trained using our CARLA stereo dataset + fine-tuning in Agoverse stereo dataset.

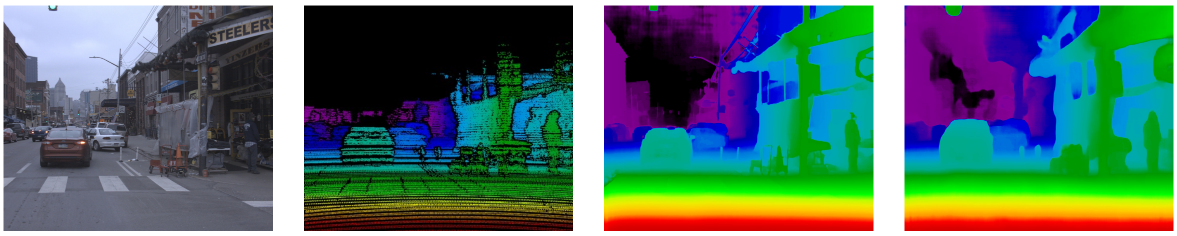


Figure 17 – Comparison between two models (PWC-Net) tested in our synthetic stereo dataset (first row) and Argoverse stereo dataset (second row). The first column is the left RGB image, and the second column is the ground-truth. The firs model is trained only using our CARLA stereo dataset (inference results on third column) and the second model is the same model with further fine-tuning in Argoverse stereo dataset (last column).

Source: Research data.

Table 7 – Result in the Argoverse stereo leaderboard

Participant team	**all:10**	**fg:10**	**bg:10**
GMStereo	1.61	1.71	1.56
MSCLab (DEQ Stereo)	2.39	3.34	2.01
LRM (PWC-Net disparity)	2.47	2.67	2.38
Odepth	3.78	4.57	3.46
ACVNet (Baseline) (XU *et al.*, 2022)	4.06	7.77	2.54

Source: Research data.

According to the online leaderboard[1], as shown in Table 7 the overall ten-pixel-error (all:10) for the PWC-Net is 2.47, we occupy the third place for all:10 and bg:10 metrics and second place for fg:10 metric. We surpassed by a wide margin the results of (XU *et al.*, 2022). For the 2022 Argoverse stereo competition, methods that run in real time are desired and algorithms must run faster than 200 ms per disparity prediction (during forward pass). We achieve an average inference time of 191.60 ms on an Nvidia GeForce RTX 2080Ti GPU.

[1] Available at: <https://eval.ai/web/challenges/challenge-page/1704/leaderboard/4066>

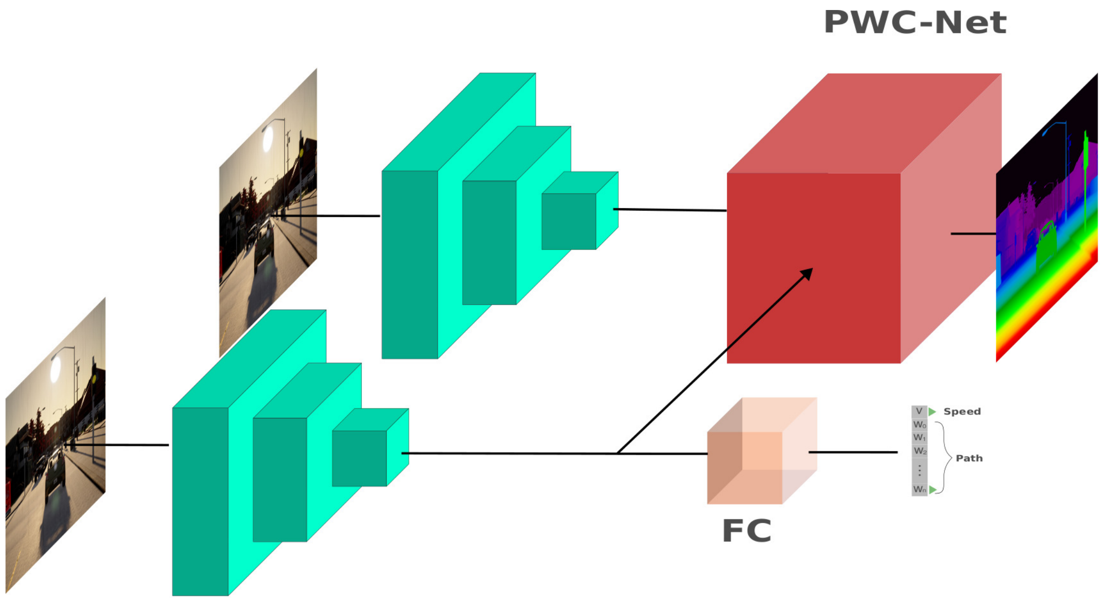


Figure 18 – PWC-Net for disparity, path and velocity estimation

Source: Elaborated by the author.

5.2.6 Joint disparity, path and velocity estimation

Building upon the PWC-Net architecture adapted for disparity estimation, we introduce a new branch dedicated to estimating both the path and the instantaneous speed.

We utilize the embeddings extracted by the backbone which processes the right image, and incorporate new heads that concurrently estimates a set of 200 waypoints for the path along with a velocity value. Our architecture for path, velocity and disparity estimation is depicted in Figure 18.

5.3 Multi-modal end-to-end Driving

This section introduces two end-to-end architectures that leverage multimodal input from three sensors: LiDAR, stereo camera, and monocular camera. These stand apart from previous models in this chapter by utilizing a structure that we name BEVSFusion.

5.3.1 BEVSFusion

This structure fuses information from various sensors into a single, informative representation within birds-eye-view (BEV) space. These implementations directly employ raw sensor data, bypassing any object detection or segmentation pre-processing. Only necessary coordinate transformations are performed.

We named this fusion **BEVSFusion** which is a rich data structure that seamlessly fuses High-level commands and Point clouds.The BEVSFusion structure receives data from three sources that are processed in the following way:

- **Stereo Camera**: We utilize a pair of cameras with specific field of view, resolution, and baseline. Disparity maps calculated with the ELAS algorithm and projected into point cloud. Stereo point cloud is transformed from the camera coordinate system to BEV coordinate system using the transformation matrix T_{cam}^{BEV}.
- **LiDAR**: The LiDAR point cloud is directly transformed to the BEV coordinate system using T_{LiDAR}^{BEV} transformation matrix. LiDAR points are then rasterized in an RGB image where a colormap encodes height information (blue for ground, yellow for above sensor). Empty pixels are filled with black.
- **High-Level Commands**: Global plan commands are converted from the world frame to the BEV frame using T_{W}^{BEV} and rasterized as colored dots in BEV space (blue for turn right, red for turn left, withe for straight and green for lane follow). High-Level commands are poses provided by a noisy GPS and are mainly given at intersections to resolve ambiguities in navigation.

Finally, these three processed elements (rasterized LiDAR, stereo, and high-level commands) are stacked into a single 9-channel image (BEVSFusion). This unified structure integrates spatial, depth, height, color, and high-level command information for robust path planning.

5.3.2 CNN-Planner: A Convolutional neural network for path regression

BEVSFusion serves as the input to a Convolutinal Neural Network that we name CNN-Planner (ROSERO *et al.*, 2022). The CNN-Planner can be represented as a function:

$$w = \text{CNN-Planner}(BEVSFusion), \tag{5.4}$$

This CNN is a architecture for regression of a sequence w of dense waypoints for the ego vehicle's trajectory. Each waypoint in the sequence $w = \{w_1, ..., w_n\}$ represents a point with (x, y) coordinates in the ego-car coordinate system. w_1 corresponds to the closest point to the ego car, while w_n denotes the furthest point on the planned trajectory. w is transformed from the ego coordinate system to the world coordinate system using T_{ego}^{W} transformation matrix.

Figure 19 visually illustrates the process, highlighting the creation of BEVSFusion through sensor fusion and its integration as input for the CNN-Planner and generating the planned trajectory. The path w in the world coordinate system is followed using a MPC controller.

For training the CNN-Planner we use the MSE loss Eq. 5.1.

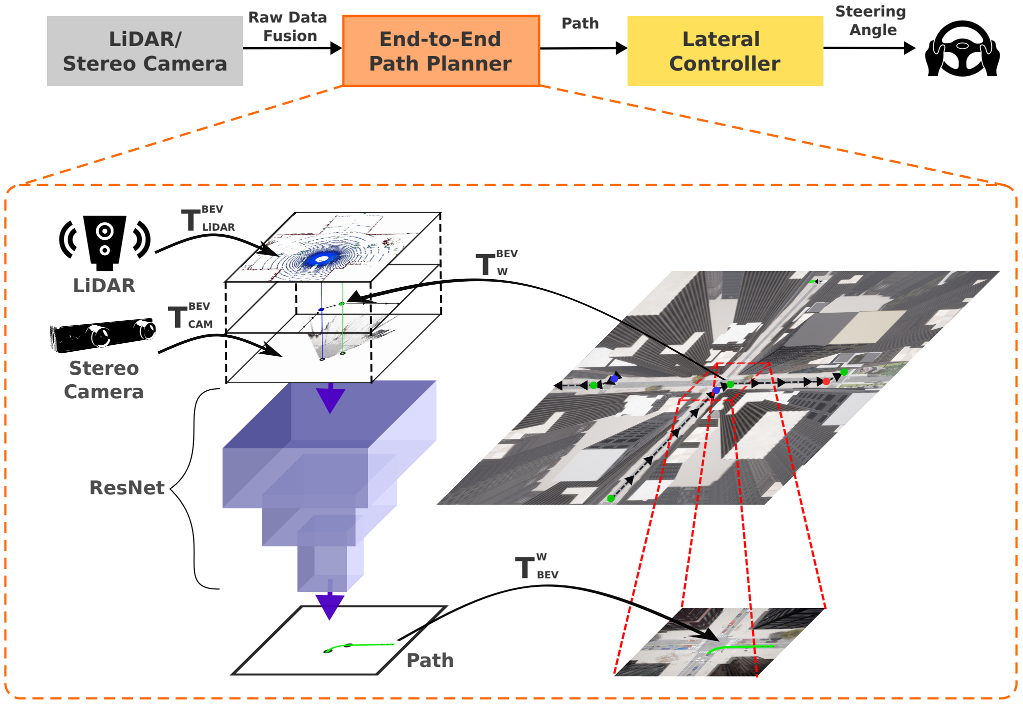


Figure 19 – CNN-Planner: Our neural path planner takes as input BEVSFusion and the output is a list *w* (path) that is followed by the MPC controller.

Source: Elaborated by the author.

5.3.3 Separate path and velocity inference

Our first implementation of a multi modal architecture uses a BEVSFusion to feeds a dedicated end-to-end path planner(CNN-Planner) . Meanwhile, a separate ResNet solely utilizes the front camera image to infer instantaneous speed. Here, sensor fusion exclusively impacts path planning, leaving vehicle speed calculation devoid of multi-sensor benefits. Figure 20 depicts this approach. The end-to-end path planner is a CNN-Planner, and the speed planner is a ResNet 18 trained using the MSE loss (Eq. 5.1).

5.3.4 Intermediate Fusion for Joint Inference

To overcome this limitation, we employ a modal fusion approach, adopting a "medium fusion" strategy. Features are extracted from the BEV data and concatenate with features extracted from the RGB front image. This fused feature vector then serves as input to a unified head, which jointly computes both the path (w) and the instantaneous velocity. Figure 21 illustrates this fusion. For joint inference we train this architecture using a weighted composed loss as Eq. 5.2.

5.4 Final Considerations

In this section, we presented the methods we implemented to tackle the autonomous driving problem using only raw sensor data as input, to generate navigation waypoints and

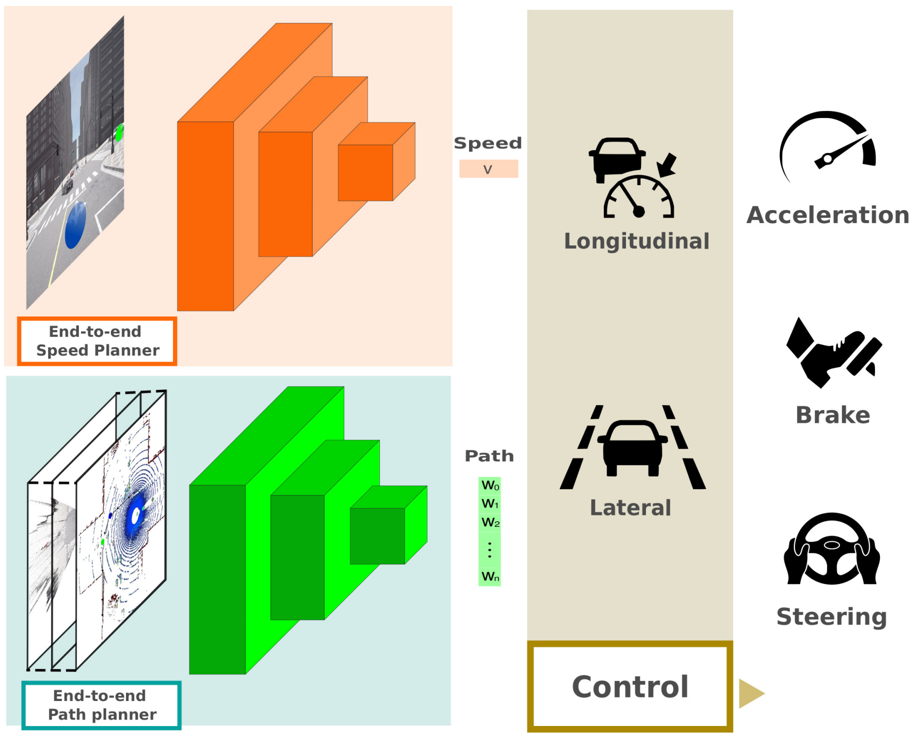


Figure 20 – End-to-end multi modal driving without fusion.

Source: Elaborated by the author.

vehicle cruising speeds per frame.

Our first proposal introduces a baseline using a single camera. Despite its limitations, such as the lack of depth perception and limited field of view of the surroundings, it offers advantages like fast, real-time execution using a lightweight ResNet-18 backbone.

Our second proposal employs a stereo rig and multi-task learning. This approach estimates disparity, path, and velocity simultaneously, improving path/velocity with depth perception estimations. Additionally, it extracts obstacles from the disparity map for emergency stops. However, the inference frequency reduces to 3 Hz due to the need for two backbones, including the more complex ResNet-50.

Our multimodal proposals significantly improve upon single or dual-camera systems. Incorporating LiDAR sensors offers a 360-degree depth perception, around the vehicle, enhancing navigation capabilities.The first multimodal proposal, which does not fuse monocular camera and LiDAR data, has inferior performance compared to the fused approach. This discrepancy is attributed to the limited influence of monocular camera data on speed inference. The second multimodal proposal leverages information from both the front camera and the BEVSFusion structure for path and speed estimation. This approach significantly outperforms our camera-only methods, demonstrating the effectiveness of our fusion scheme, BEVSFusion, and the CNN-Planner path planner.

However, despite accurate path estimation, we observed a high collision rate in all cases,

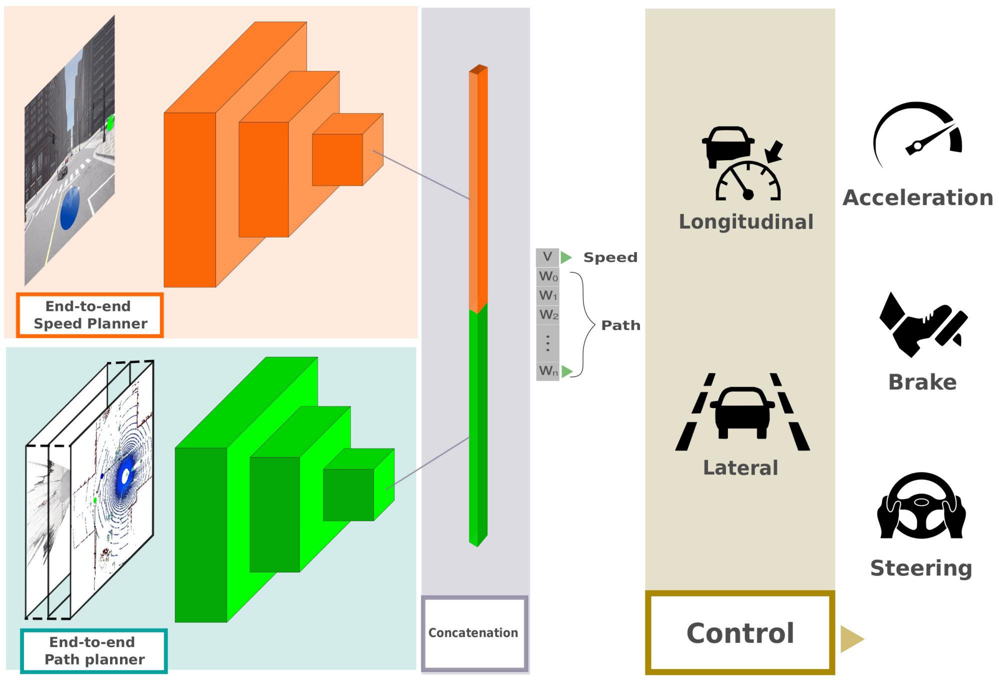


Figure 21 – End-to-end multi modal driving with intermediate fusion.

Source: Elaborated by the author.

particularly in interactions with other agents. To address this, we propose a hybrid method that combines the path estimation from sensor fusion as performed in by CNN-Planner with the reliable obstacle detection stack developed in the previous section, aiming to enhance navigation in mapless environments.

A list containing videos of our end-to-end agents going through different scenarios in simulated environments for the track SENSORS in the CARLA leaderboards in validation routes is available online[2].

[2] Available at: <https://www.youtube.com/playlist?list=PLcT95Tv_ZrmJPRu-NhNeIoL56oAjojHOO>

CHAPTER

6

HYBRID ARCHITECTURE FOR AUTONOMOUS DRIVING

This section introduces our hybrid architecture for mapless autonomous driving, designed to navigate challenging scenarios like the CARLA Leaderboard's SENSORS track. Building upon our modular pipeline described in Chapter 4, we leverage robust obstacle detection, risk assessment, and decision-making modules while replacing traditional map-based planning with our novel end-to-end path planner named CNN-Planner developed in the Chapter 5.

Conventional map-based approaches often struggle in dynamic environments lacking accurate maps. We overcome this limitation by using the CNN-Planner for mapless situations. This planner generates a set of waypoints and utilizes sensor fusion in the BEV space as its primary input.

Figure 22 displays a summary of our hybrid architecture, leveraging all available sensors as input. The bottom of the figure depicts the perception, prediction, and decision-making layers, operating as described in Chapter 4. Here, various sensors feed information to interconnected modules, collaborating to determine the desired speed for each frame.

6.1 Taking advantage of the modular pipeline

Our hybrid architecture leverages the development from the modular pipeline, starting with the perception module. Similar to the modular architecture, the perception section utilizes a module that calculates a height map for emergency situation detection. This offers an advantage over the purely end-to-end implementations discussed in the previous chapter.

For traffic light detection, the behavior mirrors the modular pipeline. A 3D detection algorithm applied to point clouds identifies three distinct object classes: vehicles, pedestrians, and bicycles. This is followed by a 3D-adapted SORT tracking algorithm, detailed in Section 4.2.

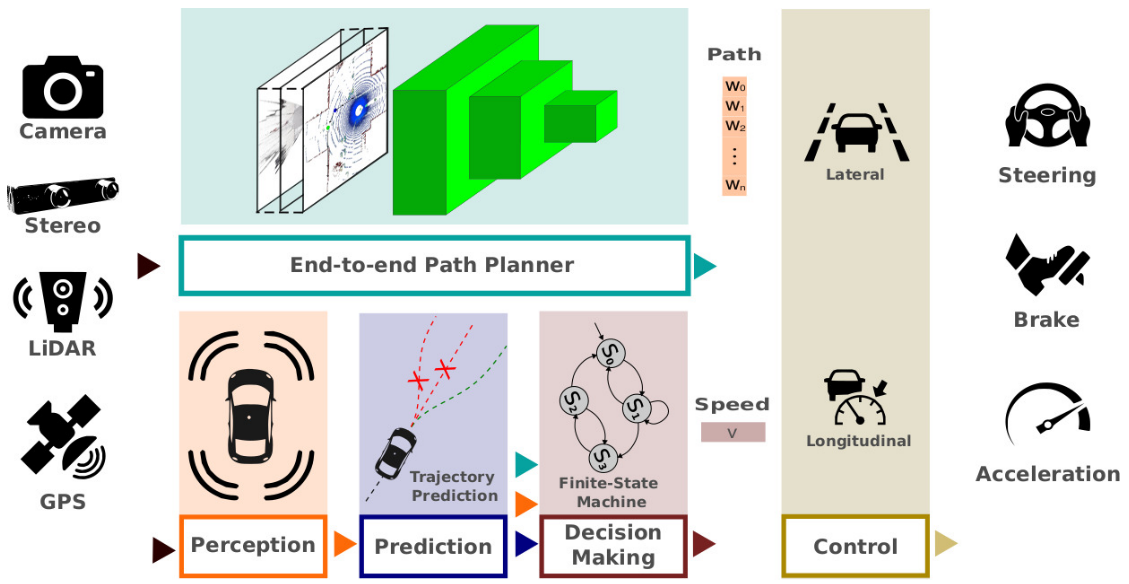


Figure 22 – Hybrid architecture for autonomous driving in unmaped scenarios.

Source: Elaborated by the author.

Prediction relies on a simple linear prediction algorithm based on vehicle speed.

The hybrid architecture additionally employs a collision risk assessment (CRA) module, akin to the one presented in Section 4.3. This module plays a crucial role in determining the significance of surrounding obstacles and is crucial to the success of the decision-making module. In section 4.1, a 40-meter map path length in front of the vehicle was utilized. Here, the CNN-Planner provides this 40-meter length by performing a regression on 200 points spaced 20 cm apart, as previously described in Section 5.3.1. Consequently, the decision-making process and the core of this architecture are also impacted by the effectiveness of the data-driven path planner.

For decision-making, the same Moore Finite State Machine, powered by obstacle detection and the CRA, is employed as in the modular pipeline. This finite state machine makes a decision on cruising speed, as detailed in Section 4.4.

Furthermore, as in the modular architecture, speed reference value and path waypoints are transmitted each frame to low-level controllers, such as those described in section 4.6 to be followed.

6.2 CNN-Planner for a Hybrid architecture

As mentioned earlier, the planning layer utilizes our CNN-Planner from Section 4.2, taking the BEVSFusion structure as input.

Using the 3D object detection information provided by the perception module, hybrid architecture incorporates in the BEVSFusion structure an additional channel compared to the BEVSFusion presented in the previous section. The new channel construction proceeds as

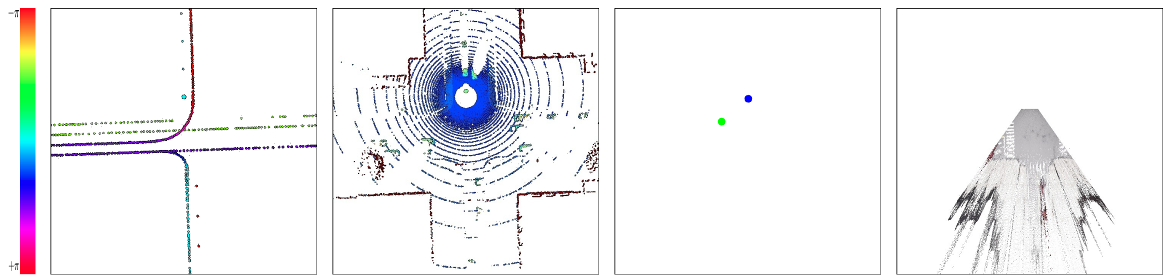


Figure 23 – Rasterized inputs for the BEVSFusion structure: footprints (orientation angle maped to hsv color), LiDAR point cloud, high level commands, and stereo point cloud

Source: Elaborated by the author.

Table 8 – Time execution for main modules in hybrid and modular architectures

Module	Module/Algorthm	Inputs	Outputs	Architecture	Average Execution Time (ms)	Device
Perception/3D detection	PointPillars	Point cloud	obstacles 3D	Modular/Hybrid	95	GPU
Perception/2D detection	Mask-RCNN	RGB image	detections 2D	Modular/Hybrid	198	GPU
Perception/tracking	SORT	Obstacles	Tracked Obstacles	Modular/Hybrid	0.23	CPU
Perception/Depth	ELAS	RGB Images	obstacles 3D	Modular/Hybrid	210	CPU
Planning/Path planning	CNN-Planner	BEVSFusion	local waypoints	Hybrid	5.3	GPU
Planning/local planner	map	global waypoints	local waypoints	Modular	0.61	CPU
Control/Lateral	MPC	local waypoints, pose	steering angle	Modular/Hybrid	0.5	CPU
Control/Longitudinal	PID	speed reference	break, throttle	Modular/Hybrid	0.1	CPU
Decision-Making	FSM	Obstacles in the path	speed reference	Modular/Hybrid	0.4	CPU

Source: Research data.

follows: we utilize the detections of 3D objects conducted by our detection module (details in Section 4.2) to obtain the position and orientation of all detected objects surrounding the ego vehicle. These poses are then transformed into bird's-eye view space. We maintain a historical record of these detections and their orientations for 200 frames, enabling the rasterization of the route traces (footprints) of surrounding vehicles. This new input gives additional information to our trajectory planner to learn from other agents in its vicinity. Footprints' points are encoded using a HSV color map to represent the rotation angle of each footprint.

Figure 23 showcases an example of the new footprint raster alongside the rasters of LiDAR, high-level commands, and stereo camera, elements already employed in the previous chapter for the BEVSFusion structure.

6.2.1 Time Execution

One of the objectives of this Thesis is to achieve soft real-time performance for our stack. Therefore, this section provides a comprehensive analysis of the execution time for each module in our framework.

Our tests were conducted locally on a high-performance computer equipped with a 16-core Intel Core i9-9900KS processor, 64 GB of RAM, and two Nvidia RTX GeForce 2080Ti graphics cards.

Table 8 presents the time taken by each module to perform its respective tasks, offering insights into the efficiency of our approach. Tasks requiring only CPU show very short execution times, all below 1 ms, except for the disparity/depth module (210 ms). While not ideal, this module is not our primary option for obstacle detection, as it is only used for traffic lights and long-range pedestrian detection.

Among the GPU modules, instance detection is the most time-consuming, taking 198 ms. Its output is merged with the disparity/depth module's output (as described in Section 8) for 3D detection of traffic lights and long-range people. Our core obstacle detection module, the 3D PointPillars detector, runs at an average of 95 ms, that is a processing rate close to 10 Hz, similar to the frequency of current LiDAR sensors.

Finally, the CNN-Planner module, leveraging a lightweight ResNet-18 for path generation in each frame, is the fastest module executing in GPU, requiring only 5 ms. Moreover, these results fulfill the soft real time execution demands of our agents, while also highlighting areas for potential optimization to further improve overall system performance.

6.2.2 Soft Real-Time System

In a soft real-time system, tasks are expected to be completed within a certain timeframe. However, unlike hard real-time systems, where missing a deadline could lead to failures, soft real-time systems can tolerate occasional delays and misses without causing system failures.

Our system is designed to process sensor data and make driving decisions quickly. While it is important to respond in a timely manner, our system can tolerate occasional delays. The goal is to maintain a balance between performance and the ability to handle these occasional delays.

We use Ubuntu 22.04.4 LTS as the operating system to deploy and run our software. Since we do not use a real-time operating system (RTOS) and due to hardware constraints, it is possible that complex and hardware-demanding algorithms may not execute quickly. For this reason, we adopt a soft real-time approach. In this work, we consider soft real-time for all our tasks, aiming to ensure that all processes can execute data processing at a minimum of 10 Hz. Despite setting this operating frequency, we are tolerant of processes that do not consistently reach that frequency. For instance, our stereo camera, together with the disparity algorithm, operates at an average of 5 Hz.

It is important to note that our CNN-Planner, used in our hybrid architecture, has an average execution time of 5.3 ms. However, it is limited by the slowest module (ELAS, with an average execution time of 198 ms). This means that although the trajectory planner is capable of running at a high frequency, it is reduced to approximately 5 Hz. Although this is a lower frequency than expected, it does not affect the navigation of our autonomous vehicle because the generated path length is 40 meters, and the maximum speed allowed for our agent is 30 m/s.

The build time of the BeVFusion structure depends on the speed of the ELAS algorithm, 3D point cloud detection (PointPillars), and tracking (SORT). The choice of the ELAS algorithm is based on the trade-off between disparity estimation quality and execution time. Although faster algorithms like BlockMatching are available, we prefer to sacrifice speed in exchange for better disparity estimation quality. Despite the availability of other algorithms, such as Semi-Global Matching (SGM), which estimate disparity with high quality, we chose ELAS because it was developed with a focus on external environmental scenarios, such as those used in autonomous driving. Additionally, we selected the PointPillars algorithm because it is designed to work in real-time and offers the best trade-off between speed and accuracy compared to other approaches that achieve top values in major datasets. Similarly, the SORT algorithm was developed for real-time applications and provides the best trade-off between accuracy and speed, as the speed of the most accurate trackers is often too slow for real-time applications.

6.3 Final Considerations

This section proposes a hybrid architecture that merges the strengths of modular pipelines and end-to-end learning for autonomous driving in map-less environments. This architecture utilizes the BeVSFusion structure and the CNN-Planner, a data-driven module trained to generate waypoints for navigation without map dependence.

Building upon the previous chapter's BeVSFusion structure, which incorporates stereo, LiDAR, and high-level command data, this chapter introduces a new channel leveraging 3D object detection from the perception module. Despite not featuring this new channel in the CADCH 2023 competition, our hybrid architecture still achieved first place. Nonetheless, the additional channel significantly enhances navigation in map-less environments by incorporating valuable information from surrounding agents about potential travel paths. This enhancement is expected to yield improved results on leaderboard 2 upon its reopening.

After CADCH, we developed the aforementioned additional channel incorporating surrounding agent footprints. This new information improves our navigation results to state-of-the-art performance on the RC (route completion) metric of leaderboard 1, as detailed in the next Chapter.

A list of videos of our hybrid agent going through different scenarios in simulated environments for the track SENSORS in the CARLA leaderboards in validation routes is available online[1].

[1] Available at: <https://www.youtube.com/playlist?list=PLcT95Tv_ZrmJPRu-NhNeIoL56oAjojHOO>

CHAPTER

7 EVALUATION AND VALIDATION

7.1 Experimental setup

Our research adopts the Robot Operating System (ROS) as the unifying framework for our modular, end-to-end, and hybrid driving architectures. ROS's publisher-subscriber communication paradigm (ZHU, 2005) facilitates efficient data exchange between components, enabling a flexible and scalable system design. The ROS master node indexes and coordinates components, while peer-to-peer messaging enables direct communication between nodes (QUIGLEY *et al.*, 2009). This structure streamlines the development and integration of multi-component systems, particularly in applications like autonomous driving and robotics.

Our autonomous driving agents implements all modules described, for the perception layer we have:

- Two monocular cameras with 71° field of view (FOV) each are combined to form a stereo camera for 3D perception, producing a pair of rectified images with dimensions of 1200×1200 pixels. The baseline of our stereo camera is 0.24 m. We utilize the ELAS algorithm (GEIGER; ROSER; URTASUN, 2010) to generate 3D point clouds from the stereo images.
- LiDAR sensor: 64 channels, 45° vertical field of view, 180° horizontal field of view, 100m range. Our system utilizes a simulated LiDAR collecting around one million data points per scan across 64 vertical layers.

  LiDAR and stereo camera are centered in the x-y plane of the ego car and mounted at 1.8m height.
- GPS and IMU: For localization and ego-motion estimation.
- CANBus: Provides vehicle internal state information such as speed and steering angle.

Our modular architecture additionally utilizes an OpenDrive map pseudo-sensor for route planning and a ObjectFinder pseudo-sensor, used exclusively for dataset creation, provides ground-truth information about dynamic and static objects within the CARLA simulator.

7.2 Metrics

Autonomous vehicles are heterogeneous and complex systems, orchestrating sensing, perception, decision-making, planning, control, and health management. Evaluating the performance of these complex systems requires a holistic approach, going beyond individual evaluation to assess the harmony of the entire system.

Traditionally, unit tests analyze individual components, seeking malfunctions and quantifying their performance with metrics like accuracy, recall, and precision (e.g., for classification algorithms (HOSSIN; SULAIMAN, 2015; THARWAT, 2018)). Integration tests take a broader perspective, examining the interplay between two or more components (e.g., obstacle detection and avoidance). Finally, system tests encompass the entire system, evaluating the harmonious collaboration of all its components (JORGENSEN, 2018; LEWIS, 2017). However, a standardized methodology for comprehensively assessing and comparing the complete performance of autonomous driving systems remains a challenge. The CARLA Leaderboards offer a standardized benchmark for evaluating autonomous driving systems, providing diverse sensor configurations and software architectures.

These leaderboards immerse the autonomous system, or "agent," in simulated urban environments. Each scenario throws diverse challenges, varying in cityscapes, traffic areas (highways, urban roads, residential areas, roundabouts, unmarked intersections), route lengths, traffic density, and weather conditions. Moreover, each route incorporates traffic situations inspired by the pre-crash typology (STATES, 2007) provided by the made haNational Highway Traffic Safety Administration of the United States (NHTSA), encompassing diverse scenarios like:

- Control loss without prior action.
- Obstacle avoidance for unexpected obstacles.
- Negotiation at roundabouts and unmarked intersections.
- Following the lead vehicle's sudden braking.
- Crossing intersections with a traffic-light-disobeying vehicle.

Leaderboard 2 expands this scenarios, adding:

- Lane changes to avoid obstacles blocking lanes.

- Yielding to emergency vehicles.
- Door obstacles (e.g. opened car door).
- Avoiding vehicles invading lanes on bends.
- Maneuvering parking cut-ins and exits.

To evaluate agent performance in each simulated scenario, CARLA Leaderboards employ a set of quantitative metrics that captures not only route completion but also adherence to traffic rules and safe driving practices. This metrics assesses the entire system's performance, transcending mere point-to-destination navigation. It factors in traffic rules, passenger and pedestrian safety, and the ability to handle both common and unexpected situations (e.g., occluded obstacles and vehicle control loss).

Key Metrics:

Driving Score (DS): The main metric of the leaderboards, calculated as the product of route completion percentage (Ri) and the infraction penalty (Pi) of the *i-th* route, $(RiPi)$. This metric rewards both efficient navigation and adherence to safety regulations.

Route Completion (RC): Percentage of the route distance successfully completed by the agent of the *i-th* route, (Ri).

Infraction Penalty (IP): $(\prod_j^{ped,veh,...,stop}(p_j^i)^{\#infractions_j})$. Aggregates all types of infractions triggered by the agent as a geometric series. Each infraction reduces the agent's score, starting from an ideal base of 1.0. Specific infraction types their penalty coefficients include:

- Collisions with pedestrians (CP) - 0.50.
- Collisions with other vehicles (CV) - 0.60.
- Collisions layout (CL) - 0.65.
- Running a red light (RLI) - 0.70.
- Stop sign infraction (SSI) - 0.80.
- Off-road infraction (ORI) - percentage of the route will not be considered.

Additional Leaderboard 2 Metrics:

- Scenario timeout (ST) - 0.70.
- Failure to maintain minimum speed (MinSI) - 0.70.
- Failure to yield to emergency vehicle (YEI) - 0.70.

Under certain circumstances, the simulation will be automatically terminated, preventing the agent from further progress on the current route. These events include:

- Route deviations (RD)
- Route timeouts (RT)
- Agent blocked (AB)

After all routes are completed, global metrics are calculated as the average of individual route metrics. The global driving score remains the primary metric for ranking agents against competitors. By employing comprehensive evaluation frameworks like CARLA Leaderboards, researchers and developers can gain valuable insights into the strengths and weaknesses of their autonomous driving systems, ultimately paving the way for safer and more robust vehicles that perform harmoniously as a whole, not just as a collection of individual components. For further details on the evaluation and metrics, visit the leaderboard website[1].

The CARLA Team offers a series of routes for training and validation for both leaderboards 1 and 2 that are available online[2] [3].

To evaluate an agent's performance, it must be submitted to the online evaluator[4]. The specific routes and the cities used are confidential. For Leaderboard 1, 10 routes are chosen and each is evaluated 10 times under varying lighting and weather conditions. Each route is roughly 1 km long, meaning an agent completing all routes at 100% would cover approximately 100 kilometers in total. Leaderboard 2 features increased difficulty compared to Leaderboard 1, with routes 10 times longer and presenting more complex scenarios. Agents must navigate these scenarios, including overtaking obstacles or yielding to emergency vehicles.

7.3 Datasets

To train the diverse components of our autonomous driving agents, we generated three comprehensive datasets running a privileged version of the our modular pipeline[5].

These datasets were created using CARLA simulator version 0.9.13 under a range of lighting and weather conditions (day, night, rain, fog) and across distinct urban environments in the CARLA towns: Town01, Town3, Town4, Town06, and Town12. These environments encompass downtown areas, residential neighborhoods, rural landscapes, and diverse vegetation.

[1] Available at: <https://leaderboard.carla.org/#evaluation-and-metrics>.
[2] Available at: <https://github.com/carla-simulator/leaderboard/tree/leaderboard-1.0/data>.
[3] Available at: <https://github.com/carla-simulator/leaderboard/tree/leaderboard-2.0/data>.
[4] Available at: <https://eval.ai/web/challenges/challenge-page/2098/overview>.
[5] The privileged version of our agent refers to an agent that has the same architecture as our modular pipeline but has access to privileged information about the simulation state, including detailed routes, maps, and precise positions (without noise) of the ego vehicle, other vehicles, and pedestrians. This privileged agent is also called the expert.

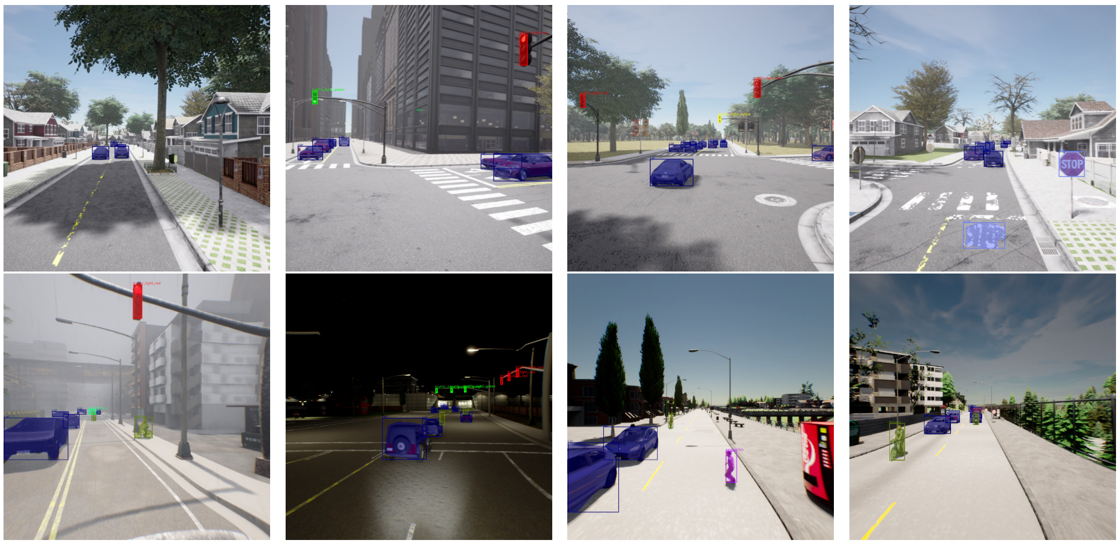


Figure 24 – Our instance detection dataset includes annotations of eight classes: Car, pedestrian, bicycle, stop sign, red traffic light, yellow traffic light, green traffic light and emergency vehicle in different urban environments and weather conditions

Source: Elaborated by the author.

- **Instance Segmentation Dataset**: We constructed a dataset of 20,000 RGB images with variable resolutions ranging from 800×800 to 1400×1400 pixels. These images encompass seven object classes: car, bicycle, pedestrian, red traffic light, yellow traffic light, green traffic light, and stop sign.

  For labeling, we employed a semi-automatic approach for cars, bicycles, pedestrians, and stop signs, leveraging sensor instances provided by the CARLA simulator. Traffic lights and stencil stop signs, however, required manual annotation for greater accuracy. All annotations were stored in the COCO format. Finally, we trained a Mask-RCNN model implemented in mmdetection[6]. for object detection and segmentation. Figure 24 showcases examples of detections achieved with our trained model. Our Instance Segmentation Dataset is available online[7].

- **3D Object Detection Dataset:** This dataset comprises 5,000 point clouds annotated with pose (relative to the ego car), height, length, width, and orientation for all cars, bicycles, and pedestrians. We leveraged the privileged sensor objects within the simulator to perform this automatic annotation. The data was subsequently saved in the KITTI format for compatibility with popular object detection algorithms. Using this dataset, we trained a PointPillars model adapted for our specific needs, implemented in the mmdetection3d framework[8].

- **Path Planner Training Dataset:** To train the path planner, we leveraged a privileged

[6] Available at: <https://github.com/open-mmlab/mmdetection>
[7] Available at: <https://github.com/luis2r/Instance-segmentatio-CARLA>
[8] Available at: <https://github.com/open-mmlab/mmdetection3d>

agent and the previously described sensors to collect approximately 300,000 frames. This agent granted access to ground-truth path information and provided error-free GPS and IMU data facilitating precise navigation. The point clouds from the LiDAR and stereo cameras were then projected and rasterized into 700x700 RGB images in the bird's-eye view space. High-level commands like "left," "right," "straight," and "lane follow" were transformed to the ego coordinate system using the command pose, then rasterized within the bird's-eye image in the same way than pointclouds but with color-coded points for commands (red for left, blue for right, white for straight, and green for lane follow). The ground-truth road path consisted of 200 waypoints spaced 20 cm apart, originating at the center of the ego car.

To simulate potential navigation errors and enhance error recovery learning, we introduced Gaussian noise to the steering wheel inputs in 50% of the routes used for dataset collection.

7.4 Results

In this section, we present the results of evaluating our agents implementing the modular, end-to-end, and hybrid architectures described earlier. All evaluations were conducted using the CARLA leaderboards.

The CARLA team offers an online benchmark with two leaderboards, each running 100 routes under various lighting and weather conditions. The specific routes and cities are secret.

For the end-to-end and hybrid architectures designed for mapless environments, we first conducted an ablation study to determine the best performer. This evaluation is carried out on a local server using the validation routes from leaderboard 1, which comprises 26 routes, each approximately 1 km long.

We present an evaluation to select the most suitable method for autonomous driving without maps among our proposed approaches. Our hybrid architecture proposal got best performance and is evaluated online in the SENSORS category. For map-based navigation, we only have one architecture, described in Chapter 4. Consequently, this agent is evaluated in the MAP category.

The complete evaluation process is presented below.

7.4.1 Evaluating End-to-End and hybrid architectures for mapless AD

To assess the performance of our end-to-end implementations across various sensor configurations, as detailed in Chapter 5, we conducted a comprehensive series of tests on a local server using leaderboard 1 validation routes[9]. Our primary objective was to understand

[9] Available at: <https://github.com/carla-simulator/leaderboard/blob/leaderboard-1.0/data/routes_testing.xml>.

Table 9 – Results: Ablation studies on both end-to-end methods and the hybrid architecture for mapless autonomous driving using the local validation routes of Leaderboard 1.

Method	DS	RC	IP	CP	CV	CL	RLI	SSI	ORI	RD	RT	AB
Monocular camera	8.24	25.38	0.40	0.22	0.68	0.75	0.18	0.00	3.40	2.86	1.09	7.56
Stereo camera	12.45	32.15	0.57	0.16	0.36	0.31	0.32	0.00	1.52	1.47	1.23	5.46
Multimodal no fusion	33.80	64.22	0.43	0.22	0.30	0.25	1.01	0.00	1.05	0.99	0.60	1.20
Multimodal fusion	35.87	67.1	0.51	0.19	0.29	0.22	0.91	0.00	0.18	0.60	0.57	0.94
Hybrid	**52.55**	**96.70**	0.61	0.28	0.12	0.03	0.86	0.00	0.10	0.02	0.00	0.12

Source: Research data.

the impact of using specific sensors and methods. We initiated our evaluation by comparing the performance scores of different implementations across diverse sensor setups.

Running locally compare all of our end-to-end implementations with our hybrid implementation, all designed for navigation in unmapped environments. A similar assessment was conducted prior to CADCH 2023 to determine the optimal architecture for participating in the SENSORS track of the challenge, as the current leaderboard only permits one entry per team. Table 9 shows the results obtained from this evaluation.

For local evaluation, we utilized routes provided by the Carla team for validation of their leaderboard 1 on GitHub. These routes differ from those used for official leaderboard 1 evaluation, which remain confidential and are exclusively assessed online.

The hybrid architecture exhibited significantly superior performance compared to the end-to-end implementations in both main metrics, with RC at 96.70% and DS at 52.55%. Notably, the route completion approached 100%.

Conversely, the end-to-end architectures yielded lower performance, primarily due to a high frequency of obstructions or blockages (high AB metric). This limited their ability to navigate a substantial portion of the route, consequently impacting their DS score. As anticipated, the limited field of view offered by their single camera setup was a major factor in their underperformance.

Future implementations could explore the use of multiple cameras or a 360-degree camera to enhance their performance. Based on this evaluation, our best agent for the SENSORS track is the hybrid architecture, which has been submitted to the leaderboards as will be shown later.

Additionally, both the hybrid and end-to-end architectures used the same CNN-Planner. However, the end-to-end versions lacked access to the perception module, preventing them from utilizing surrounding agent footprints as used in the hybrid version. This motivates us to analyze the individual impact of each sensor within the BEVSFusion structure and its influence on agent performance.

Table 10 – Results: Ablation study about influence of sensor fusion on data driven planner on hybrid implementation (local evaluation)

Sensor in BEV	DS	RC	IP	CP	CV	CL	RLI	SSI	ORI	RD	RT	AB
All rasters fusion (+footprints)	**52.55**	**96.70**	0.61	0.18	0.12	0.03	0.86	0.00	0.10	0.02	0.00	0.12
Without footprints	41.17	81.81	0.56	0.28	0.19	0.00	0.08	0.00	0.12	0.04	0.10	0.25
Without stereo BEV	47.80	90.22	0.43	0.30	0.67	0.05	1.07	0.00	0.45	0.03	0.55	0.16
Without high-level-commands	10.85	22.81	0.51	1.80	0.59	1.80	0.51	0.00	4.10	5.10	6.62	8.06
Only LiDAR BEV	5.75	20.11	0.28	1.52	2.37	1.27	0.52	0.00	0.59	3.17	5.01	4.54

Source: Research data.

7.4.2 Influence of sensor fusion in the data driven planner

We delved into the structure of our BEVSFusion module, responsible for fusing diverse sensors. Our tests were conducted using CNN-planner as path regressor in our hybrid implementation. Our aim was to evaluate the influence of different types of information within this fusion structure on path planning generation.

We conducted a ablation studie by systematically omitting specific data elements from the BEVSFusion module. These omissions included high-level commands, stereo information within the BEV space, and footprints of other actors. Throughout all tests, we consistently maintained the LiDAR point cloud in the BEV space. Table 10 show the results.

From the results we can see that removing high-level commands at intersections demonstrably hinders performance. These commands play a crucial role in resolving navigational ambiguities. Without them, the agent is likely to become disoriented and struggle even at the first intersection.

On the other hand, we can note that all the rasters contributed to the BEVSFusion structure have an influence on the final result, although some have a greater impact than others.

It can also be seen that the addition of the footprint of the other surrounding agents significantly contributed to the fulfillment of the route. This is because this information covers 360 degrees, and the position and orientation of the other objects provide an idea of where viable trajectories can be generated, thus avoiding trajectories opposite to the direction of the road or away from the routes commonly traveled by other vehicles.

7.4.3 Results on CARLA Leaderboards (online)

This section presents the performance of our modular and hybrid and modular agent architectures on the CARLA Leaderboards (Track SENSORS and Track MAP respectively), demonstrating their effectiveness in both map-based and mapless navigation tasks. To validate our models, we utilized the leaderboards provided online by the CARLA team (Leaderboard 1 and Leaderboard 2). We employed the Track MAP from the two benchmarks to assess our modular architecture, and the Track SENSORS were used to evaluate our hybrid architecture, which was chosen after comparison with end-to-end implementations

Table 11 – Results: CARLA Leaderboard 1, Track MAP

Team	**Method**	**DS**	**RC**	**IP**	**CP**	**CV**	**CL**	**RLI**	**SSI**	**ORI**	**RD**	**RT**	**AB**
Anonymous	Map TF++	61.17	81.81	0.70	0.01	0.99	0.00	0.08	0.00	0.00	0.00	0.00	0.55
mmfn	MMFN+ (TPlanner)[10] (no paper)	59.85	82.81	0.71	0.01	0.59	0.00	0.51	0.00	0.00	0.00	0.62	0.06
LRM 2023	**CaRINA agent**	41.56	**86.03**	0.52	0.075	0.38	0.13	1.6	0.03	0.01	0.04	0.05	1.29
RaphaeL	GRI-based DRL (CHEKROUN *et al.*, 2023)	33.78	57.44	0.57	0.00	3.36	0.50	0.52	0.00	1.52	1.47	0.23	0.80
mmfn	MMFN (ZHANG *et al.*, 2022)	22.80	47.22	0.63	0.09	0.67	0.05	1.07	0.00	0.45	0.00	0.00	1003.88
RobeSafe research group	Techs4AgeCar+ (GÓMEZ-HUÉLAMO *et al.*, 2022)	18.75	75.11	0.28	1.52	2.37	1.27	1.22	0.00	0.59	0.17	0.01	1.28
ERDOS	Pylot (GOG *et al.*, 2021)	16.70	48.63	0.50	1.18	0.79	0.01	0.95	0.00	0.01	0.44	0.10	3.30
LRM 2019	CaRINA	15.55	40.63	0.47	1.06	3.35	1.79	0.28	0.00	3.28	0.34	0.00	7.26

Source: CARLA (2024).

Table 12 – Results: CARLA challenge 2023. CARLA leaderboard 2, Track MAP

Team	**Method**	**DS**	**RC**	**IP**	**CP**	**CV**	**CL**	**RLI**	**SSI**	**ORI**	**RD**	**RT**	**AB**	**YEI**	**ST**	**MinSI**
Kyber-E2E	”	3.11	5.28	0.67	0.36	0.63	0.27	0.09	0.09	0.01	0.00	0.09	0.09	0.00	0.54	0.00
LRM 2023	**CaRINA agent**	1.14	3.65	0.46	**0.00**	2.89	1.31	**0.00**	0.53	**0.00**	0.13	1.31	1.18	**0.00**	2.10	**0.00**

Source: CARLA (2024).

Navigation using Map (Leaderboard 1 and 2: Track MAP)

We employed our modular CaRINA stack for map-based navigation using OpenDRIVE format as mentioned in previous sections.

Table 11 illustrates the results for Leaderboard 1 on the track MAP. On the Track MAP, we secured third place in driving score metric (DS=41.56) and the highest score in route completion (RC=86.03) among all competitors using our modular pipeline. These achievements highlight the combined strength of our CaRINA modules.

Table 12 shows the evaluation in the track MAP of Leaderboard 2. We achieved second place in driving score (DS=1.14) and route completion (RC=3.65%), only narrowly surpassed by another modular architecture. Importantly, our off-road infraction penalty in this track (ORI=0.0) emphasizes the seamless navigation facilitated by the map. This compares favorably to all methods on both Leaderboard 1 and 2 Track MAP, where some map-based approaches based on TF++ (JAEGER; CHITTA; GEIGER, 2023) and MMFN (ZHANG *et al.*, 2022) also achieve an ORI=0.0.

Mapless Navigation (Leaderboard 1 and 2: Track SENSORS)

We evaluated our hybrid CaRINA stack for mapless navigation.

Table 13 illustrates the results for leaderboard 1 in the Track SENSORS, our hybrid CaRINA agent achieved the second place on route completion score RC=93.13%, surpassing other autonomous driving methods primarily based on end-to-end learning. We also obtained a high driving score (DS=42.09) compared to similar approaches.

Table 14 presents the results for Leaderboard 2 on Track SENSORS, where our performance dominated the new leaderboard with a driving score (DS) of 1.232 and a route completion (RC) of 9.55%, showcasing a significant difference, more than twice that of the second place's performance. This highlights the effectiveness and competitive of our hybrid architecture for

Table 13 – Results: CARLA Leaderboard 1, Track SENSORS

Method	**DS**	**RC**	**IP**	**CP**	**CV**	**CL**	**RLI**	**SSI**	**ORI**	**RD**	**RT**	**AB**
ReasonNet (SHAO *et al.*, 2023)	**79.95**	89.89	0.89	0.02	0.13	0.01	0.08	0.00	0.04	0.00	0.01	0.33
InterFuser (SHAO *et al.*, 2022)	76.18	88.23	0.84	0.04	0.37	0.14	0.22	0.00	0.13	0.00	0.01	0.43
TCP (WU *et al.*, 2022)	75.14	85.63	0.87	0.00	0.32	0.00	0.09	0.00	0.04	0.00	0.00	0.54
TF++ WP Ensemble (JAEGER; CHITTA; GEIGER, 2023)	66.32	78.57	0.84	0.00	0.50	0.00	0.01	0.00	0.12	0.00	0.00	0.71
LAV (CHEN; KRÄHENBÜHL, 2022)	61.85	**94.46**	0.64	0.04	0.70	0.02	0.17	0.00	0.25	0.09	0.04	0.10
TF++ WP (JAEGER; CHITTA; GEIGER, 2023)	61.57	77.66	0.81	0.02	0.41	0.00	0.03	0.00	0.08	0.00	0.00	0.71
TransFuser (CHITTA *et al.*, 2023)(PRAKASH; CHITTA; GEIGER, 2021a)	61.18	86.69	0.71	0.04	0.81	0.01	0.05	0.00	0.23	0.00	0.01	0.43
Latent TransFuser (CHITTA *et al.*, 2023)	45.20	66.31	0.72	0.02	1.11	0.02	0.05	0.00	0.16	0.00	0.04	1.82
CaRINA hybrid	42.09	**93.13**	0.45	0.71	0.53	0.15	1.80	0.11	0.28	0.06	0.00	0.24
GRIAD (CHEKROUN *et al.*, 2021)	36.79	61.85	0.60	0.00	2.77	0.41	0.48	0.00	1.39	1.11	0.34	0.84
World on Rails (CHEN; KOLTUN; KRÄHENBÜHL, 2021)	31.37	57.65	0.56	0.61	1.35	1.02	0.79	0.00	0.96	1.69	0.00	0.47
MaRLn (TOROMANOFF; WIRBEL; MOUTARDE, 2020)	24.98	46.97	0.52	0.00	2.33	2.47	0.55	0.00	1.82	1.44	0.79	0.94
NEAT (CHITTA; PRAKASH; GEIGER, 2021)	21.83	41.71	0.65	0.04	0.74	0.62	0.70	0.00	2.68	0.00	0.00	5.22
AIM-MT (CHITTA; PRAKASH; GEIGER, 2021)	19.38	67.02	0.39	0.18	1.53	0.12	1.55	0.00	0.35	0.00	0.01	2.11
TransFuser (CVPR 2021) (PRAKASH; CHITTA; GEIGER, 2021b)	16.93	51.82	0.42	0.91	1.09	0.19	1.26	0.00	0.57	0.00	0.01	1.96
CNN-Planner 2019 (ROSERO *et al.*, 2022)	15.40	50.05	0.41	0.08	4.67	0.42	0.35	0.00	2.78	0.12	0.00	4.63
Learning by Cheating(CHEN *et al.*, 2020)	8.94	17.54	0.73	0.00	0.40	1.16	0.71	0.00	1.52	0.03	0.00	4.69
MaRLn (TOROMANOFF; WIRBEL; MOUTARDE, 2020)	5.56	24.72	0.36	0.77	3.25	13.23	0.85	0.00	10.73	2.97	0.06	11.41
CILRS (CODEVILLA *et al.*, 2019)	5.37	14.40	0.55	2.69	1.48	2.35	1.62	0.00	4.55	4.14	0.00	4.28
CaRINA 2019 (ROSERO *et al.*, 2020)	4.56	23.80	0.41	0.01	7.56	51.52	20.64	0.00	14.32	0.00	0.00	10055.99

Source: CARLA (2024).

Table 14 – Results: CARLA challenge 2023. CARLA leaderboard 2, Track SENSORS

Team	**Method**	**DS**	**RC**	**IP**	**CP**	**CV**	**CL**	**RLI**	**SSI**	**ORI**	**RD**	**RT**	**AB**	**YEI**	**ST**	**MinSI**
LRM 2023	**CaRINA hybrid**	**1.23**	9.55	0.31	0.25	**1.64**	**0.25**	0.25	**0.40**	0.43	0.10	0.30	0.60	0.10	1.20	**0.15**
Tuebingen_AI	Zero-shot TF++ (JAEGER; CHITTA; GEIGER, 2023)	0.58	8.53	0.38	0.17	1.80	0.51	0.00	3.76	0.35	0.06	0.56	0.51	0.00	2.19	0.17
CARLA	baseline	0.25	15.20	0.10	1.23	2.49	0.79	0.03	0.94	0.47	0.50	0.00	0.13	0.13	0.69	0.19

Source: CARLA (2024).

mapless navigation, surpassing the state-of-the-art end-to-end learning method Zero-shot TF++ (variation of the TF++ method (JAEGER; CHITTA; GEIGER, 2023)).

Leaderboard 2 was used for the 2023 CARLA Challenge. We achieved first place in the Track SENSORS and second place in the Track MAP categories of the 2023 CARLA Challenge with our modular and hybrid CaRINA agent versions, respectively.

7.5 Analysis and Discussion

The results in the previous section were obtained from the official CARLA Leaderboards. However, it is important to note that we only have access to the final scores for each metric, lacking additional details regarding the vehicle's performance in individual traffic scenarios and their respective impacts on the overall score. In this section, we analyze the results based on on both leaderboard scores and offline experiments conducted on a local machine. Through the insights gained from these offline experiments, we can draw conclusions about the performance of the modular, end-to-end and hybrid autonomous driving architectures.

7.5.0.1 Modular Architecture

We assessed the modular navigation architecture on the MAP track in Leaderboards 1 and 2. In both cases, our RC (Route Completion) scores surpassed those of any other technique, indicating that our vehicles completed more trajectory segments than competing agents. Nevertheless, our agent incurred a lower IP (Infraction Penalty) than the top two agents with the highest DS (Driving Score). This outcome is primarily influenced by two types of infractions, namely RLI (Running Red Light) and AB (Agent Blocked), as the remaining infraction metrics show no significant difference from the top two agents in Leaderboard 1.

In the first case (RLI), the complexity of the road network layout (especially the intersections) poses a significant challenge in correctly associating the traffic light with the vehicle's current trajectory. Consequently, as the vehicle approaches the intersection, it may either pass the location where it should stop and wait for the traffic light or fail to detect it through its cameras. In the second scenario, the vehicle struggles to pass an obstacle (e.g., object or another vehicle) that is stationary in its lane. Both situations concerns the scene understanding within the perceptual system and decision-making. Specifically, the second scenario presents an additional challenge when there is oncoming traffic in the opposite lane. In this case, in addition to recognizing the need for a lane change, the vehicle must also identify a gap in the traffic and promptly react to enter the gap. In offline experiments, we observed that due to the conservative driving style adopted, the vehicle is not always swift enough to enter a gap before the next approaching vehicle arrives.

Finally, it is worth noting that, despite having the highest RC among the agents on the leaderboard, the CV (collision with vehicles) is significantly lower than that of the other agents. This result emphasizes the effectiveness of the perception and decision-making systems in detecting and avoiding collisions with other vehicles.

7.5.0.2 Hybrid Architecture

We assessed the hybrid architecture on the SENSORS track of Leaderboards 1 and 2. The "CaRINA hybrid" agent achieved significant Route Completion (RC), securing the 6th and 1st positions in Leaderboards 1 and 2, respectively. These outcomes are similar to those on the MAP track. However, in this track, the vehicle operates without access to map information, relying entirely on data-driven path planning. However, the considerable number of collisions with other vehicles (CV) significantly impacted the performance of the navigation architecture.

Based on offline experiments, we listed two scenarios that potentially affected the perception and navigation system, increasing the number of collisions with other vehicles. In the first scenario, lane changes were initiated due to potential obstructions in the current driving lane, such as other vehicles or objects. We observed that the data-driven path planning demonstrated superior adaptability, estimating lane-change trajectories in a broader range of scenarios compared to the modular navigation pipeline. However, the execution of these maneuvers resulted in more collisions with oncoming traffic in the opposite lane. This behavior manifested in two additional metrics, apart from CV: AB (agent blocked), which is lower than the modular architecture due to the vehicle executing more lane-change and overtake maneuvers; and RD (route deviation), which occurs because the vehicle struggles to return to its trajectory after some collisions.

In the second scenario, the focus is on intersections, particularly when the vehicle fails to adhere to a red light signal. The vehicle approaches the intersection and attempts to cross it, but in most instances, it fails to avoid collisions with oncoming traffic. In certain situations, the vehicle stops midway through the intersection while trying to evade collisions and complete the maneuver. However, this behavior also results in the blockage of the vehicle (AB), contributing to intersection deadlock, or the vehicle running over road layouts (CL) and incurring off-road infractions (ORI).

7.5.0.3 Comparison and Final Remarks

The results in the previous sections provided a comprehensive assessment of the performance of modular, end-to-end and hybrid architectures for autonomous navigation. The use of the CARLA simulator and Leaderboards 1 and 2 enabled a quantitative and qualitative evaluation of this approaches, providing valuable insights into their strengths and weaknesses. Accordingly, this section presents a brief overview of the results and observations related to both navigation strategies proposed in this work.

The primary distinction between both approaches lies in their methodology. While the modular architecture relies on parsing the OpenDrive map to estimate trajectories and navigate, the end-to-end and hybrid approaches employ a mapless data-driven path planning technique to guide the vehicle to its destination. Furthermore, the Route Completion (RC) of both approaches showed similarities across both leaderboards. This suggests the efficacy of the data-driven method in estimating trajectories in diverse urban scenarios, a notable challenge given the unfamiliarity of testing cities within the CARLA simulator. These cities feature different road network layouts and city landscapes. Additionally, the evaluation involved navigating under varying weather and light conditions, significantly impacting the performance of vision-based algorithms. The sensor fusion adopted in the data-driven approach, using images and point cloud, contributes to making the method more robust to adverse conditions, improving its adaptability and generalization.

Another important observation concerning the three approaches and the CARLA challenge is the complexity of the diverse driving scenarios. Apart from requiring effective perception, decision-making, and planning systems, the challenge demands swift responses from the vehicle. For instance, when the vehicle needs to change lanes with traffic in the adjacent lane, it must identify a gap and react promptly. The offline experiments demonstrated various scenarios where the system's components correctly identified these situations. However, the vehicle was not quick enough to execute maneuvers safely, resulting in collisions and other traffic infractions. We adopted a conservative driving style, which demands more time to react to scenarios involving interactions with other traffic participants. The smooth acceleration change curve led to dangerous situations, given that the behavior of other vehicles was designed to present complex and challenging scenarios for the autonomous agent. For example, in certain situations, due to lane changes or sudden brakes of the ego-vehicle (CaRINA agent), the surrounding vehicles collide with the rear of the ego-vehicle, as they were not designed to stop in such scenarios.

CHAPTER

8

CONCLUSION

With the development of this thesis, we were able to verify that we can take advantage of the advances made so far in the areas of perception, decision-making, and planning for modular pipelines, in addition to the progress in end-to-end learning for autonomous driving. We proposed and built a new hybrid approach, leveraging the best of previous methods. With this, we answered the scientific question initially raised. We obtained comparable results with the state-of-the-art (SOTA) methods in autonomous driving for navigation with and without a map in the recognized benchmark provided by the CARLA team. Additionally, the proposed methods meet the requirements to run in soft real-time, which is of great importance for real applications, as demonstrated in this work.

Our research has not only proposes a versatile autonomous driving architecture, but also implements a robust approach to navigation. By blending the strengths of map-based and mapless paradigms within a unified framework. Integrating modularity with end-to-end path planning resulted in a holistic system that excels in both navigation styles. Modular simplicity facilitates transparent debugging and efficient issue identification, fostering continuous performance improvement. Our trajectory planning, despite using a minimalistic module compared to complex competitor models, competes impressively, exemplified by our route completion score in all tracks.

Through comprehensive experimentation and evaluation, it becomes evident that hybrid architectures, integrating both modular and end-to-end approaches, are an option to end-to-end approaches. The fusion of diverse sensor inputs, coupled with robust perception and control modules, results in reliable and adaptable autonomous driving systems.

The incorporation of information from multiple sensors, including LiDAR and cameras and detections, significantly enhances the perception capabilities of autonomous vehicles. Multimodal fusion techniques, such as BEVSFusion, offer a holistic view of the surrounding environment, enabling more accurate path planning and obstacle detection.

Standardized evaluation frameworks, such as the CARLA Leaderboards, play a vital role in assessing the performance of autonomous driving algorithms. Continual benchmarking and comparison against state-of-the-art methods are essential for driving innovation and progress in the field.

The research conducted in this thesis contributes to the broader autonomous driving community by presenting novel architectures, methodologies, and insights. The developed algorithms and frameworks have the potential to drive advancements in autonomous vehicle technology, paving the way for safer, more efficient, and more accessible transportation systems.

Finally, the success of the CaRINA agent, evident in its first-place in CADCH 2023 track SENSORS and second place in track MAP , testifies to the effectiveness and adaptability of our proposed architectures.

8.1 Challenges and Future Directions

To define the scope of our system and suggest future work based on this research, it is important to outline its limitations. First, our system operates under a soft real-time approach rather than hard real-time. Moreover, it has only been tested in simulated environments. Therefore, it is crucial to implement the system on real platforms and conduct further testing in real-world scenarios to validate its performance and effectiveness. Despite the success of our approaches, we identified areas for improvement:

Our current approach to traffic light detection relies on the position of traffic lights to determine stopping points. However, there may be unfamiliar traffic light configurations in hidden cities on the test server that we haven't accounted for, which could potentially cause issues with our detection system. This dependence on traffic light position may not be applicable in all scenarios. To address this limitation, we should consider exploring end-to-end or hybrid architectures to handle the traffic light problem.

Despite the success of our methods, we observe infractions, related to collision avoidance especially in the area of collisions with vehicles. This is primarily due to lane changes requested by high-level commands (lane change left, lane change right), becoming hazardous when other high-speed vehicles are using the targeted lanes during lane changes. A potential solution could involve a new model and controller considering the other surrounding vehicle velocities or training algorithms to adapt the speed during lane changes, particularly when other cars are traveling at high speeds.

Recognizing the limitations of a simple linear motion model is important. This model may not accurately reflect complex maneuvers or sudden changes in object's direction. Therefore, it is essential to explore more advanced prediction models in future research, potentially integrating acceleration data or historical movement patterns.

While significant progress has been made in mapped environments, navigating in unmapped scenarios remains challenging. Future research directions should focus on developing robust mapless navigation techniques, leveraging advancements in sensor technology and machine learning algorithms.

The field of autonomous driving presents numerous opportunities for further exploration and advancement. Future research endeavors could focus on refining existing architectures, exploring novel sensor modalities, enhancing real-time decision-making capabilities, and addressing safety and ethical considerations.

In this thesis we use relatively simple MPC and PID controllers. Future work can be focused on designing and implementing Adaptive Cruise Control (ACC) systems. These ACC systems will enhance both lateral and longitudinal control aspects of autonomous driving by considering critical variables such as acceleration, confort, stability and tracking. The development of this ACC system will involve extensive simulation and real-world testing to validate its performance. By integrating these considerations into the design, the ACC system aims to improve the overall safety, comfort, and reliability of autonomous vehicles.

Future work can also include implementing more sophisticated models than the currently used bicycle model. These models can approximate the real dynamics of the vehicle by incorporating additional factors such as the mass of the car and aerodynamic effects. By adopting models that closely mimic real vehicle dynamics, the control strategies can be optimized for better safety and performance. This includes smoother transitions and more reliable operation under diverse driving conditions. The development and integration of these more advanced models will involve extensive simulation and real-world testing.

In future work, exploring and implementing advanced control techniques to enhance the performance and reliability of autonomous driving systems can be included. The following control strategies will be considered: Nonlinear Model Predictive Controller (NMPC), Robust Linear Quadratic Regulator (RLQR), Model-Free Control, and Quadratic Programming (QP).

BIBLIOGRAPHY

ABDULLA, W. **Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow**. [S.l.]: Github, 2017. <https://github.com/matterport/Mask_RCNN>. Citation on page 33.

AUTOWARE. **Architecture overview**. 2024. Accessed: 2023-01-23. Available: <https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-architecture/>. Citations on pages 40 and 41.

BEWLEY, A.; GE, Z.; OTT, L.; RAMOS, F.; UPCROFT, B. Simple online and realtime tracking. **CoRR**, abs/1602.00763, 2016. Available: <http://arxiv.org/abs/1602.00763>. Citation on page 52.

BOJARSKI, M.; TESTA, D. D.; DWORAKOWSKI, D.; FIRNER, B.; FLEPP, B.; GOYAL, P.; JACKEL, L. D.; MONFORT, M.; MULLER, U.; ZHANG, J.; ZHANG, X.; ZHAO, J.; ZIEBA, K. End to end learning for self-driving cars. **CoRR**, abs/1604.07316, 2016. Available: <http://arxiv.org/abs/1604.07316>. Citation on page 34.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000. Citation on page 36.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV: Computer Vision in C++ with the OpenCV Library**. 2nd. ed. [S.l.]: O'Reilly Media, Inc., 2013. ISBN 1449314651, 9781449314651. Citation on page 34.

BROWN, A.; XIE, W.; KALOGEITON, V.; ZISSERMAN, A. Smooth-ap: Smoothing the path towards large-scale image retrieval. In: **European Conference on Computer Vision (ECCV), 2020.** [S.l.: s.n.], 2020. Citation on page 55.

BRUMMELEN, J. V.; O'BRIEN, M.; GRUYER, D.; NAJJARAN, H. Autonomous vehicle perception: The technology of today and tomorrow. **Transportation research part C: emerging technologies**, Elsevier, 2018. Citation on page 47.

CAI, P.; WANG, S.; SUN, Y.; LIU, M. Probabilistic end-to-end vehicle navigation in complex dynamic environments with multimodal sensor fusion. **IEEE Robotics and Automation Letters**, IEEE, v. 5, n. 3, p. 4218–4224, 2020. Citations on pages 40, 41, and 42.

CALDAS, K. A.; BARBOSA, F. M.; SILVA, J. A.; SANTOS, T. C.; GOMES, I. P.; ROSERO, L. A.; WOLF, D. F.; JR, V. G. Autonomous driving of trucks in off-road environment. **Journal of Control, Automation and Electrical Systems**, Springer, v. 34, n. 6, p. 1179–1193, 2023. Citations on pages 26 and 44.

CARLA, T. **Leaderboard CARLA**. 2024. Accessed: 2024-01-30. Available: <https://leaderboard.carla.org/leaderboard>. Citations on pages 93 and 94.

CASAS, S.; SADAT, A.; URTASUN, R. Mp3: A unified model to map, perceive, predict and plan. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2021. p. 14403–14412. Citations on pages 40, 41, and 42.

CHANG, M.-F.; LAMBERT, J.; SANGKLOY, P.; SINGH, J.; BAK, S.; HARTNETT, A.; WANG, D.; CARR, P.; LUCEY, S.; RAMANAN, D.; HAYS, J. Argoverse: 3d tracking and forecasting with rich maps. In: **2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2019. p. 8740–8749. Citation on page 38.

CHEKROUN, R.; TOROMANOFF, M.; HORNAUER, S.; MOUTARDE, F. GRI: general reinforced imitation and its application to vision-based autonomous driving. **CoRR**, abs/2111.08575, 2021. Available: <https://arxiv.org/abs/2111.08575>. Citation on page 94.

____. Gri: General reinforced imitation and its application to vision-based autonomous driving. **Robotics**, v. 12, n. 5, 2023. ISSN 2218-6581. Available: <https://www.mdpi.com/2218-6581/12/5/127>. Citation on page 93.

CHEN, D.; KOLTUN, V.; KRÄHENBÜHL, P. Learning to drive from a world on rails. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [S.l.: s.n.], 2021. p. 15590–15599. Citation on page 94.

CHEN, D.; KRÄHENBÜHL, P. Learning from all vehicles. In: **CVPR**. [S.l.: s.n.], 2022. Citation on page 94.

CHEN, D.; ZHOU, B.; KOLTUN, V.; KRÄHENBÜHL, P. Learning by cheating. In: KAELBLING, L. P.; KRAGIC, D.; SUGIURA, K. (Ed.). **Proceedings of the Conference on Robot Learning**. PMLR, 2020. (Proceedings of Machine Learning Research, v. 100), p. 66–75. Available: <https://proceedings.mlr.press/v100/chen20a.html>. Citation on page 94.

CHEN, L.; WU, P.; CHITTA, K.; JAEGER, B.; GEIGER, A.; LI, H. End-to-end autonomous driving: Challenges and frontiers. **arXiv preprint arXiv:2306.16927**, 2023. Citation on page 25.

CHITTA, K.; PRAKASH, A.; GEIGER, A. Neat: Neural attention fields for end-to-end autonomous driving. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2021. p. 15793–15803. Citation on page 94.

CHITTA, K.; PRAKASH, A.; JAEGER, B.; YU, Z.; RENZ, K.; GEIGER, A. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. **Pattern Analysis and Machine Intelligence (PAMI)**, 2023. Citation on page 94.

CODEVILLA, F.; MüLLER, M.; LóPEZ, A.; KOLTUN, V.; DOSOVITSKIY, A. End-to-end driving via conditional imitation learning. In: **2018 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.: s.n.], 2018. p. 4693–4700. Citation on page 34.

CODEVILLA, F.; SANTANA, E.; LOPEZ, A. M.; GAIDON, A. Exploring the limitations of behavior cloning for autonomous driving. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2019. Citation on page 94.

CONTRIBUTORS, M. **MMFlow: OpenMMLab Optical Flow Toolbox and Benchmark**. 2021. <https://github.com/open-mmlab/mmflow>. Citation on page 69.

DIAZ-DIAZ, A.; OCAÑA, M.; LLAMAZARES, Á.; GÓMEZ-HUÉLAMO, C.; REVENGA, P.; BERGASA, L. M. Hd maps: Exploiting opendrive potential for path planning and map monitoring. In: IEEE. **2022 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.], 2022. p. 1211–1217. Citations on pages 48 and 49.

DOSOVITSKIY, A.; FISCHER, P.; ILG, E.; HAUSSER, P.; HAZIRBAS, C.; GOLKOV, V.; SMAGT, P. van der; CREMERS, D.; BROX, T. Flownet: Learning optical flow with convolutional networks. In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. Citation on page 70.

DOSOVITSKIY, A.; ROS, G.; CODEVILLA, F.; LOPEZ, A.; KOLTUN, V. CARLA: An open urban driving simulator. In: **Proceedings of the 1st Annual Conference on Robot Learning**. [S.l.: s.n.], 2017. p. 1–16. Citation on page 69.

DUDEK, G.; JENKIN, M. **Computational Principles of Mobile Robotics**. 2nd. ed. New York, NY, USA: Cambridge University Press, 2010. ISBN 0521871573, 9780521871570. Citation on page 61.

DUPUIS, M.; STROBL, M.; GREZLIKOWSKI, H. Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks. In: **Proc. of the Driving Simulation Conference Europe**. [S.l.: s.n.], 2010. p. 231–242. Citation on page 49.

FAN, H.; ZHU, F.; LIU, C.; ZHANG, L.; ZHUANG, L.; LI, D.; ZHU, W.; HU, J.; LI, H.; KONG, Q. Baidu apollo em motion planner. **arXiv preprint arXiv:1807.08048**, 2018. Citation on page 40.

FERNANDES, L. C.; SOUZA, J. R.; PESSIN, G.; SHINZATO, P. Y.; SALES, D.; MENDES, C.; PRADO, M.; KLASER, R.; MAGALHãES, A. C.; HATA, A.; PIGATTO, D.; Castelo Branco, K.; GRASSI, V.; OSORIO, F. S.; WOLF, D. F. Carina intelligent robotic car: Architectural design and applications. **Journal of Systems Architecture**, v. 60, n. 4, p. 372–392, 2014. ISSN 1383-7621. Available: <https://www.sciencedirect.com/science/article/pii/S1383762113002841>. Citations on pages 26 and 43.

FERNANDES, L. C.; SOUZA, J. R.; SHINZATO, P. Y.; PESSIN, G.; MENDES, C. C. T.; OSORIO, F. S.; WOLF, D. F. Intelligent robotic car for autonomous navigation: Platform and system architecture. In: **2012 Second Brazilian Conference on Critical Embedded Systems**. [S.l.: s.n.], 2012. p. 12–17. Citation on page 26.

FRAICHARD, T.; SCHEUER, A. From reeds and shepp's to continuous-curvature paths. **IEEE Transactions on Robotics**, IEEE, v. 20, n. 6, p. 1025–1035, 2004. Citation on page 62.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: **Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2012. Citation on page 38.

GEIGER, A.; ROSER, M.; URTASUN, R. Efficient large-scale stereo matching. In: **Asian Conference on Computer Vision (ACCV)**. [S.l.: s.n.], 2010. Citation on page 85.

GIRSHICK, R. Fast r-cnn. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 1440–1448. ISSN 2380-7504. Citation on page 32.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Region-based convolutional networks for accurate object detection and segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 38, n. 1, p. 142–158, Jan 2016. ISSN 0162-8828. Citation on page 32.

GOG, I.; KALRA, S.; SCHAFHALTER, P.; WRIGHT, M. A.; GONZALEZ, J. E.; STOICA, I. Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous vehicles. In: IEEE. **2021 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.], 2021. p. 8806–8813. Citation on page 93.

GÓMEZ-HUÉLAMO, C.; DIAZ-DIAZ, A.; ARALUCE, J.; ORTIZ, M. E.; GUTIÉRREZ, R.; ARANGO, F.; LLAMAZARES, Á.; BERGASA, L. M. How to build and validate a safe and reliable autonomous driving stack? a ros based software modular architecture baseline. In: IEEE. **2022 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.], 2022. p. 1282–1289. Citation on page 93.

HARTLEY, R. I. Theory and practice of projective rectification. **International Journal of Computer Vision**, v. 35, n. 2, p. 115–127, Nov 1999. ISSN 1573-1405. Available: <https://doi.org/10.1023/A:1008115206617>. Citation on page 36.

HE, K.; GKIOXARI, G.; DOLLAR, P.; GIRSHICK, R. Mask r-cnn. In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017. Citation on page 51.

HE, K.; GKIOXARI, G.; DOLLáR, P.; GIRSHICK, R. Mask r-cnn. In: **2017 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017. p. 2980–2988. ISSN 2380-7504. Citation on page 33.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778. ISSN 1063-6919. Citation on page 32.

____. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778. Citation on page 54.

HIRSCHMULLER, H. Stereo processing by semiglobal matching and mutual information. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 30, n. 2, p. 328–341, Feb 2008. ISSN 0162-8828. Citation on page 37.

HOSSIN, M.; SULAIMAN, M. A review on evaluation metrics for data classification evaluations. **International Journal of Data Mining & Knowledge Management Process**, Academy & Industry Research Collaboration Center (AIRCC), v. 5, n. 2, p. 1, 2015. Citation on page 86.

HOUSTON, J.; ZUIDHOF, G.; BERGAMINI, L.; YE, Y.; CHEN, L.; JAIN, A.; OMARI, S.; IGLOVIKOV, V.; ONDRUSKA, P. One thousand and one hours: Self-driving motion prediction dataset. **arXiv preprint arXiv:2006.14480**, 2020. Citation on page 54.

HU, S.; CHEN, L.; WU, P.; LI, H.; YAN, J.; TAO, D. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In: SPRINGER. **European Conference on Computer Vision**. [S.l.], 2022. p. 533–549. Citations on pages 40, 42, and 43.

HUANG, W.; WANG, K.; LV, Y.; ZHU, F. Autonomous vehicles testing methods review. In: IEEE. **2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)**. [S.l.], 2016. p. 163–168. Citation on page 27.

ILG, E.; MAYER, N.; SAIKIA, T.; KEUPER, M.; DOSOVITSKIY, A.; BROX, T. Flownet 2.0: Evolution of optical flow estimation with deep networks. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 1647–1655. Citations on pages 70 and 71.

JAEGER, B.; CHITTA, K.; GEIGER, A. Hidden biases of end-to-end driving models. In: **Proc. of the IEEE International Conf. on Computer Vision (ICCV)**. [S.l.: s.n.], 2023. Citations on pages 93 and 94.

JO, K.; KIM, J.; KIM, D.; JANG, C.; SUNWOO, M. Development of autonomous car—part ii: A case study on the implementation of an autonomous driving system based on distributed architecture. **IEEE Transactions on Industrial Electronics**, IEEE, v. 62, n. 8, p. 5119–5132, 2015. Citations on pages 25, 40, and 41.

JORGENSEN, P. C. **Software testing: a craftsman's approach**. [S.l.]: CRC press, 2018. Citation on page 86.

KALRA, N.; PADDOCK, S. M. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? **Transportation Research Part A: Policy and Practice**, Elsevier, v. 94, p. 182–193, 2016. Citation on page 27.

KATRAKAZAS, C.; QUDDUS, M.; CHEN, W.-H.; DEKA, L. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 60, p. 416–442, 2015. Citations on pages 39 and 61.

KLASER, R. L.; OSóRIO, F. S.; WOLF, D. Vision-based autonomous navigation with a probabilistic occupancy map on unstructured scenarios. In: **2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol**. [S.l.: s.n.], 2014. p. 146–150. Citations on pages 26 and 43.

KOOPMAN, P.; WAGNER, M. Challenges in autonomous vehicle testing and validation. **SAE International Journal of Transportation Safety**, JSTOR, v. 4, n. 1, p. 15–24, 2016. Citation on page 27.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Available: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>. Citation on page 31.

LANG, A. H.; VORA, S.; CAESAR, H.; ZHOU, L.; YANG, J.; BEIJBOM, O. Pointpillars: Fast encoders for object detection from point clouds. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2019. Citation on page 52.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Handwritten digit recognition with a back-propagation network. In: TOURETZKY, D. (Ed.). **Advances in Neural Information Processing Systems (NIPS 1989)**. Denver, CO: Morgan Kaufman, 1990. v. 2. Citation on page 31.

LEWIS, W. E. **Software testing and continuous quality improvement**. [S.l.]: CRC press, 2017. Citation on page 86.

LIMA, P. F.; TRINCAVELLI, M.; MÅRTENSSON, J.; WAHLBERG, B. Clothoid-based model predictive control for autonomous driving. In: IEEE. **Control Conference (ECC), 2015 European**. [S.l.], 2015. p. 2983–2990. Citation on page 62.

LIU, S.; LI, L.; TANG, J.; WU, S.; GAUDIOT, J.-L. **Creating Autonomous Vehicle Systems**. [S.l.]: Morgan & Claypool Publishers, 2017. i–186 p. Citation on page 25.

MENZE, M.; HEIPKE, C.; GEIGER, A. Joint 3d estimation of vehicles and scene flow. In: **ISPRS Workshop on Image Sequence Analysis (ISA)**. [S.l.: s.n.], 2015. Citation on page 71.

____. Object scene flow. **ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)**, 2018. Citation on page 71.

MORAES, G.; MOZART, A.; AZEVEDO, P.; PIUMBINI, M.; CARDOSO, V. B.; OLIVEIRA-SANTOS, T.; SOUZA, A. F. D.; BADUE, C. Image-based real-time path generation using deep neural networks. In: IEEE. **2020 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2020. p. 1–8. Citations on pages 40 and 42.

OBAYASHI, M.; UTO, K.; TAKANO, G. Appropriate overtaking motion generating method using predictive control with suitable car dynamics. In: IEEE. **2016 IEEE 55th Conference on Decision and Control (CDC)**. [S.l.], 2016. p. 4992–4997. Citation on page 62.

OPENDRIVE. **ASAM OpenDRIVE 1.8.0**. 2023. Accessed: 2023-01-27. Available: <https://www.asam.net/standards/detail/opendrive/>. Citation on page 48.

PADEN, B.; ČÁP, M.; YONG, S. Z.; YERSHOV, D.; FRAZZOLI, E. A survey of motion planning and control techniques for self-driving urban vehicles. **IEEE Transactions on intelligent vehicles**, IEEE, v. 1, n. 1, p. 33–55, 2016. Citation on page 39.

POMERLEAU, D. A. Alvinn: An autonomous land vehicle in a neural network. In: TOURETZKY, D. (Ed.). **Advances in Neural Information Processing Systems**. Morgan-Kaufmann, 1988. v. 1. Available: <https://proceedings.neurips.cc/paper/1988/file/812b4ba287f5ee0bc9d43bbf5bbe87fb-Paper.pdf>. Citation on page 34.

PRAKASH, A.; CHITTA, K.; GEIGER, A. Multi-modal fusion transformer for end-to-end autonomous driving. In: **Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2021. Citation on page 94.

____. Multi-modal fusion transformer for end-to-end autonomous driving. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2021. p. 7077–7087. Citation on page 94.

QUIGLEY, M.; CONLEY, K.; GERKEY, B.; FAUST, J.; FOOTE, T.; LEIBS, J.; WHEELER, R.; NG, A. Y. Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA workshop on open source software**. [S.l.], 2009. v. 3, n. 3.2, p. 5. Citations on pages 47 and 85.

REDA, M.; ONSY, A.; GHANBARI, A.; HAIKAL, A. Y. Path planning algorithms in the autonomous driving system: A comprehensive review. **Robotics and Autonomous Systems**, Elsevier, p. 104630, 2024. Citation on page 42.

REDMON, J.; DIVVALA, S. K.; GIRSHICK, R. B.; FARHADI, A. You only look once: Unified, real-time object detection. **CoRR**, abs/1506.02640, 2015. Available: <http://arxiv.org/abs/1506.02640>. Citation on page 32.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: **Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1**. Cambridge, MA, USA: MIT Press,

2015. (NIPS'15), p. 91–99. Available: <http://dl.acm.org/citation.cfm?id=2969239.2969250>. Citation on page 33.

ROSERO, L. **Detecção de obstáculos usando fusão de dados de percepção 3D e radar em veículos automotivos**. Master's Thesis (mathesis) — University of São Paulo, Instituto de Ciências Matemáticas e de Computação, 2017. Citation on page 44.

ROSERO, L.; SILVA, J.; WOLF, D.; OSóRIO, F. Cnn-planner: A neural path planner based on sensor fusion in the bird's eye view representation space for mapless autonomous driving. In: **2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)**. [S.l.: s.n.], 2022. p. 181–186. Citations on pages 74 and 94.

ROSERO, L. A.; GOMES, I. P.; SILVA, J. A. R. da; SANTOS, T. C. d.; NAKAMURA, A. T. M.; AMARO, J.; WOLF, D. F.; OSÓRIO, F. S. A software architecture for autonomous vehicles: Team lrm-b entry in the first carla autonomous driving challenge. **arXiv preprint arXiv:2010.12598**, 2020. Citations on pages 29, 44, and 94.

ROSERO, L. A.; GOMES, I. P.; SILVA, J. A. R. da; PRZEWODOWSKI, C. A.; WOLF, D. F.; OSóRIO, F. S. Integrating modular pipelines with end-to-end learning: A hybrid approach for robust and reliable autonomous driving systems. **Sensors**, v. 24, n. 7, 2024. ISSN 1424-8220. Available: <https://www.mdpi.com/1424-8220/24/7/2097>. Citations on pages 30 and 47.

ROSERO, L. A.; OSóRIO, F. S. Calibration and multi-sensor fusion for on-road obstacle detection. In: **2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR)**. [S.l.: s.n.], 2017. p. 1–6. Citation on page 44.

SCHARSTEIN, D.; HIRSCHMÜLLER, H.; KITAJIMA, Y.; KRATHWOHL, G.; NEŠIĆ, N.; WANG, X.; WESTLING, P. High-resolution stereo datasets with subpixel-accurate ground truth. In: JIANG, X.; HORNEGGER, J.; KOCH, R. (Ed.). **Pattern Recognition**. Cham: Springer International Publishing, 2014. p. 31–42. ISBN 978-3-319-11752-2. Citation on page 38.

SCHöPS, T.; SCHöNBERGER, J. L.; GALLIANI, S.; SATTLER, T.; SCHINDLER, K.; POLLEFEYS, M.; GEIGER, A. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2017. p. 2538–2547. Citation on page 38.

SHAO, H.; WANG, L.; CHEN, R.; LI, H.; LIU, Y. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. **arXiv preprint arXiv:2207.14024**, 2022. Citations on pages 40, 41, 42, and 94.

SHAO, H.; WANG, L.; CHEN, R.; WASLANDER, S. L.; LI, H.; LIU, Y. Reasonnet: End-to-end driving with temporal and global reasoning. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2023. p. 13723–13733. Citations on pages 40, 41, 42, and 94.

SONG, S.; HU, X.; YU, J.; BAI, L.; CHEN, L. Learning a deep motion planning model for autonomous driving. In: IEEE. **2018 IEEE Intelligent Vehicles Symposium (IV)**. [S.l.], 2018. p. 1137–1142. Citations on pages 40, 42, and 43.

STATES, N. H. T. S. A. of the U. **Pre-Crash Scenario Typology for Crash Avoidance Research**. 2007. Accessed: 2023-01-30. Available: <https://www.nhtsa.gov/sites/nhtsa.gov/files/pre-crash_scenario_typology-final_pdf_version_5-2-07.pdf>. Citation on page 86.

SUN, D.; YANG, X.; LIU, M.-Y.; KAUTZ, J. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In: **2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 8934–8943. Citations on pages 68, 69, and 70.

SZELISKI, R. **Computer vision: algorithms and applications**. [S.l.]: Springer Science & Business Media, 2010. Citation on page 37.

TAMPUU, A.; MATIISEN, T.; SEMIKIN, M.; FISHMAN, D.; MUHAMMAD, N. A survey of end-to-end driving: Architectures and training methods. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 33, n. 4, p. 1364–1384, 2020. Citation on page 25.

TAŞ, Ö. Ş.; SALSCHEIDER, N. O.; POGGENHANS, F.; WIRGES, S.; BANDERA, C.; ZOFKA, M. R.; STRAUSS, T.; ZÖLLNER, J. M.; STILLER, C. Making bertha cooperate–team annieway's entry to the 2016 grand cooperative driving challenge. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 19, n. 4, p. 1262–1276, 2018. Citations on pages 40 and 41.

TENG, S.; HU, X.; DENG, P.; LI, B.; LI, Y.; AI, Y.; YANG, D.; LI, L.; XUANYUAN, Z.; ZHU, F. *et al.* Motion planning for autonomous driving: The state of the art and future perspectives. **IEEE Transactions on Intelligent Vehicles**, IEEE, 2023. Citations on pages 25 and 39.

THARWAT, A. Classification assessment methods. **Applied Computing and Informatics**, Elsevier, 2018. Citation on page 86.

THRUN, S.; MONTEMERLO, M.; DAHLKAMP, H.; STAVENS, D.; ARON, A.; DIEBEL, J.; FONG, P.; GALE, J.; HALPENNY, M.; HOFFMANN, G.; LAU, K.; OAKLEY, C.; PALATUCCI, M.; PRATT, V.; STANG, P.; STROHBAND, S.; DUPONT, C.; JENDROSSEK, L.-E.; KOELEN, C.; MARKEY, C.; RUMMEL, C.; NIEKERK, J. van; JENSEN, E.; ALESSANDRINI, P.; BRADSKI, G.; DAVIES, B.; ETTINGER, S.; KAEHLER, A.; NEFIAN, A.; MAHONEY, P. Stanley: The robot that won the darpa grand challenge. In: ____. **The 2005 DARPA Grand Challenge: The Great Robot Race**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1–43. ISBN 978-3-540-73429-1. Available: <https://doi.org/10.1007/978-3-540-73429-1_1>. Citation on page 51.

TOROMANOFF, M.; WIRBEL, E.; MOUTARDE, F. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020. Citation on page 94.

VITELLI, M.; CHANG, Y.; YE, Y.; FERREIRA, A.; WOŁCZYK, M.; OSIŃSKI, B.; NIENDORF, M.; GRIMMETT, H.; HUANG, Q.; JAIN, A. *et al.* Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. In: IEEE. **2022 International Conference on Robotics and Automation (ICRA)**. [S.l.], 2022. p. 897–904. Citations on pages 40 and 43.

WANG, D.; WANG, C.; WANG, Y.; WANG, H.; PEI, F. An autonomous driving approach based on trajectory learning using deep neural networks. **International journal of automotive technology**, Springer, v. 22, p. 1517–1528, 2021. Citations on pages 40, 42, and 43.

WEI, J.; SNIDER, J. M.; KIM, J.; DOLAN, J. M.; RAJKUMAR, R.; LITKOUHI, B. Towards a viable autonomous driving research platform. In: IEEE. **Intelligent Vehicles Symposium (IV), 2013 IEEE**. [S.l.], 2013. p. 763–770. Citation on page 40.

WU, P.; JIA, X.; CHEN, L.; YAN, J.; LI, H.; QIAO, Y. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. In: **NeurIPS**. [S.l.: s.n.], 2022. Citations on pages 40, 41, 42, and 94.

XIAO, Y.; CODEVILLA, F.; GURRAM, A.; URFALIOGLU, O.; LóPEZ, A. M. Multimodal end-to-end autonomous driving. **IEEE Transactions on Intelligent Transportation Systems**, v. 23, n. 1, p. 537–547, 2022. Citations on pages 40, 41, and 42.

XU, G.; CHENG, J.; GUO, P.; YANG, X. Attention concatenation volume for accurate and efficient stereo matching. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2022. p. 12981–12990. Citation on page 72.

XU, Z.; XIAO, X.; WARNELL, G.; NAIR, A.; STONE, P. Machine learning methods for local motion planning: A study of end-to-end vs. parameter learning. In: IEEE. **2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)**. [S.l.], 2021. p. 217–222. Citation on page 42.

ZABIH, R.; WOODFILL, J. Non-parametric local transforms for computing visual correspondence. In: EKLUNDH, J.-O. (Ed.). **Computer Vision — ECCV '94**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994. p. 151–158. ISBN 978-3-540-48400-4. Citation on page 37.

ZHANG, Q.; TANG, M.; GENG, R.; CHEN, F.; XIN, R.; WANG, L. Mmfn: Multi-modal-fusion-net for end-to-end driving. In: IEEE. **2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. [S.l.], 2022. p. 8638–8643. Citations on pages 40, 41, 42, and 93.

ZHOU; CHELLAPPA. Computation of optical flow using a neural network. In: **IEEE 1988 International Conference on Neural Networks**. [S.l.: s.n.], 1988. p. 71–78 vol.2. Citation on page 32.

ZHU, H. **Software design methodology: From principles to architectural styles**. [S.l.]: Elsevier, 2005. Citation on page 85.

APPENDIX

A

PUBLICATIONS

Published journal articles:

Rosero, L.A.; Gomes, I.P.; da Silva, J.A.R.; Przewodowski, C.A.; Wolf, D.F.; & Osório, F.S. Integrating Modular Pipelines with End-to-End Learning: A Hybrid Approach for Robust and Reliable Autonomous Driving Systems. Sensors 2024, 24, 2097. <https://doi.org/10.3390/s24072097>

Caldas, K. A., Barbosa, F. M., Silva, J. A., Santos, T. C., Gomes, I. P., Rosero, L. A., ... & Grassi Jr, V. (2023). Autonomous driving of trucks in off-road environment. Journal of Control, Automation and Electrical Systems, 34(6), 1179-1193.

Published conference papers:

Rosero, L., Silva, J., Wolf, D., & Osório, F. (2022, October). CNN-Planner: A neural path planner based on sensor fusion in the bird's eye view representation space for mapless autonomous driving. In 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE) (pp. 181-186). IEEE.

Rosero, L. A., & Osório, F. S. (2017, November). Calibration and multi-sensor fusion for on-road obstacle detection. In 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR) (pp. 1-6). IEEE.

Shinzato, P. Y., dos Santos, T. C., Rosero, L. A., Ridel, D. A., Massera, C. M., Alencar, F., ... & Wolf, D. F. (2016, November). CaRINA dataset: An emerging-country urban scenario benchmark for road detection systems. In 2016 IEEE 19th international conference on intelligent transportation systems (ITSC) (pp. 41-46). IEEE.

dos Santos, T. C., Gómez, A. E., Massera Filho, C., Gomes, D., Perafan, J. C., Wolf, D. F.,

... & Rosero, L. A. (2015, October). A simulation framework for multi-vehicle communication. In 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR) (pp. 301-308). IEEE.

Alencar, F. A., Rosero, L. A., Massera Filho, C., Osório, F. S., & Wolf, D. F. (2015, October). Fast metric tracking by detection system: Radar blob and camera fusion. In 2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR) (pp. 120-125). IEEE.

Preprint articles:

Rosero, L. A., Gomes, I. P., da Silva, J. A. R., Santos, T. C. D., Nakamura, A. T. M., Amaro, J., ... & Osório, F. S. (2020). A software architecture for autonomous vehicles: Team lrm-b entry in the first carla autonomous driving challenge. arXiv preprint arXiv:2010.12598.

ICMC USP
SÃO CARLOS