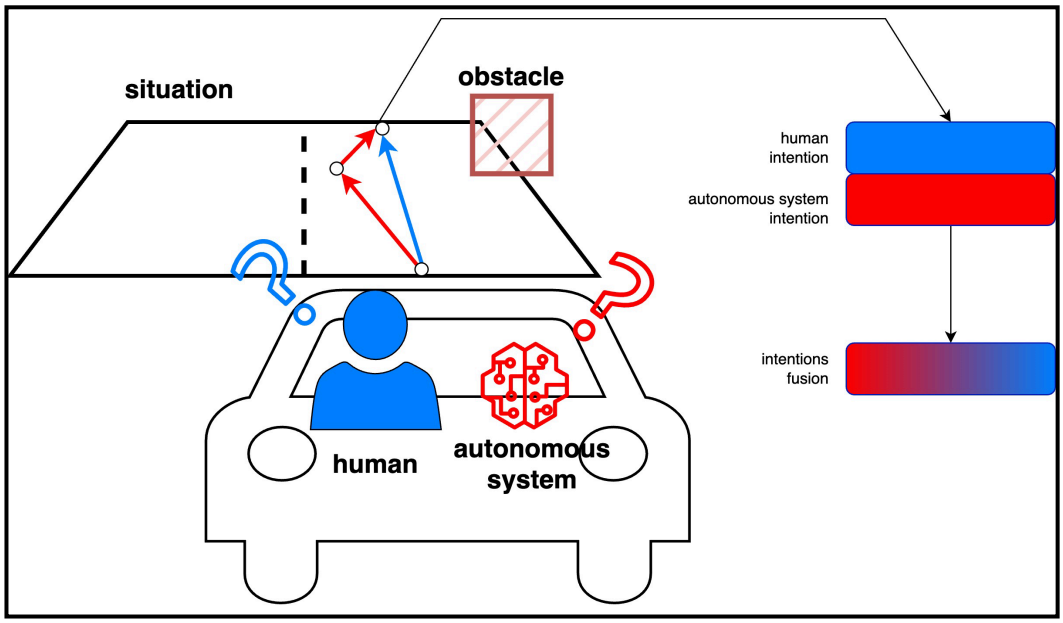
Shared navigation in a cybernetic multi-agent autonomous system

Hugo Pousseur

Par **Hugo POUSSEUR**

Shared navigation in a cybernetic multi-agent autonomous system

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 28 mars 2024
Spécialité : Automatique et Robotique : Unité de recherche Heudyasic (UMR-7253)
D2802





Université de Technologie de Compiègne

École doctorale n°71 : Sciences pour l'ingénieur

Laboratoire Heudiasyc - UMR CNRS 7253

Shared navigation in a cybernetic multi-agent autonomous system

Thèse soutenue le 28 mars 2024 par

Hugo POUSSEUR

Pour obtenir le grade de

Docteur de l'Université de Technologie de Compiègne

Spécialité : Automatique et Robotique

Sous la direction de :

Alessandro Victorino & Hiroshi Fujimoto

JURY:

Reviewers

Name: BASSET Michel
Position: Researcher Director
Field: Automatic Robotic Informatics
University: Université de Haute Alsace

Name: MARTINET Philippe
Position: Professor
Field: Automatic Robotic
University: Inria

Examiners

Name: IVANOV Valentin
Position: Professor
Field: Automatic Robotic
University: Technical University of Ilmenau

Name: SHYROKAU Barys
Position: Professor
Field: Automatic Robotic
University: Delft University of Technology

Name: TALJ Reine
Position: CNRS Researcher - HDR
Field: Automatic Robotic
University: Université de technologie de Compiègne

Name: MAMMAR Saïd
Position: Professor
Field: Automatic Robotic
University: Université d'Evry Paris-Saclay

Directors

Name: VICTORINO Alessandro
Position: Associate Professor HDR
Field: Automatic Robotic
University: Université de technologie de Compiègne

Name: FUJIMOTO Hiroshi
Position: Professor
Field: Automatic Robotic
University: University of Tokyo

Abstract

My thesis is based on shared navigation between the autonomous system and the human. In our research, we focus on command fusion. In our approach, both entities, the human and the autonomous system, simultaneously control the vehicle, and a module acquires their commands and performs the command fusion. This approach involves studying the intentions of both the human and the autonomous system to ensure the most appropriate fusion of their choices and to evaluate the decision-making of each entity. The intention of the autonomous system is calculated using a visual servoing controller. The implementation of visual servoing relies on a deep learning network detecting lanes. For the human driver, who actively drives and cannot express their intention simultaneously, we use a deep learning-based model to predict their intention. The construction of this model required the creation of a driving dataset using our vehicles and the development of a recurrent model that integrates data of various types. Each of these intentions is then evaluated according to specific criteria, including safety, comfort, and context, to guide the fusion process towards the selection of the highest quality intention. This quantification is based on a state analysis derived from the realization of these intentions. We then use game theory to facilitate the fusion process, where each entity, human and autonomous system, aims to steer the final command towards their choice.

Contents

List of Figures

List of Tables

Acronyms

ACC Adaptive Cruise Control. 15

AI artificial intelligence. 14

CAN Controller Area Network. 18

CNN Convolutional Neural network. 37, 45, 51, 76

CTRA Constant Turn Rate And Acceleration. 40

CTRV Constant Turn Rate And Velocity. 33, 35, 40

DDT Dynamic Driving Task. 26

DMV Department of Motor Vehicles. 14

DVR Driver Vehicle Road. 29, 31, 38

DWA Dynamic Window Approach. 7, 16, 44, 45, 53, 54, 55, 56, 57, 64

GNSS Global Navigation Satellite System. 18, 36

GPS Global Positioning System. 18, 32, 36

GPU Graphics Processing Unit. 19

IBVS Image-Based Visual Servoing. 46

IDWA Image-Based Dynamic Window Approach. 53

LSTM Long Short-Term Memory. 39, 40, 66, 76, 80

MAE Mean Error Absolute. 80

MPC Model Predictive Control. 32, 33

PBVS Position-Based Visual Servoing. 46

RAM Random-Access Memory. 19

RNN Recurent Neural Network. 66

ROS Robot Operating System. 18, 19

SAE Society of Automotive Engineers. 14, 15

TTC Time To Collision. 6, 25, 32, 33, 41

UTC Université de Technologie de Compiègne. 13

VR Vehicle Road. 28, 37

1 Introduction

***Abstract**: Introduction to the thesis, explaining the contributions made in this thesis, as well as the publications and detailing the thesis plan. The work environnement, exploited during this thesis, is presented.*

Contents

1.1 Context

This thesis is government-funded through a scholarship from the Université de Technologie de Compiègne (UTC).

In the context of the doctoral co-supervision program between UTC and University of Tokyo. The thesis works were realized at the Heudiasyc Laboratory UMR CNRS 7253 at UTC and set in an international environment, as part of the European project OWheel[1]. This project is coordinated by the German university of Ilmenau. It brings together a set of universities participating from the United Kingdom, the Netherlands, Italy, Lithuania, and France. In addition to these participants, non-European partners are also part of the project, including Japan and South Africa. In this context, the thesis works were done in collaboration to the Fujimoto laboratory, affiliated with the University of Tokyo.

1.2 Problem Statement And Global Overview

1.2.1 Objective

The aim of this thesis is to integrate intentions between entities sharing the car: the human and the automated system. To achieve this integration at the scale of the car, it is essential to understand the intentions of each entity over a short time horizon. Our focus on a short time horizon is due to our emphasis on fusion at the local navigation level. For the autonomous system, we have chosen to reuse visual servoing approaches previously developed in our laboratory [10]. Driving the vehicle according to the information provided by the camera. For the human, it is necessary to define a model capable of predicting their intentions by analyzing the commands they have previously executed on the vehicle, as well as sensor data and intrinsic vehicle data. Thus, for each entity, we have an intention that is defined over a short time horizon. This intention is rich in information defining the desired maneuver (lane keeping and obstacle avoidance for examples) and the way to achieve it. These intentions are evaluated to guide the fusion towards the intention of the highest quality Once the intentions are predicted and evaluated, the fusion of intentions should be applied. The result of this fusion is not a selection of one intention over the other but a merge of both intentions, based on evaluations. Figure 1.1 illustrates the relationship between these different steps. Furthermore, this fusion process should be capable, in future perspectives, of incorporating constraints from surrounding vehicles to refine the intention fusion by considering the behavior of vehicles around the ego-vehicle.

[1]Owheel project: https://cordis.europa.eu/project/id/872907, https://o-wheel.eu/partners/

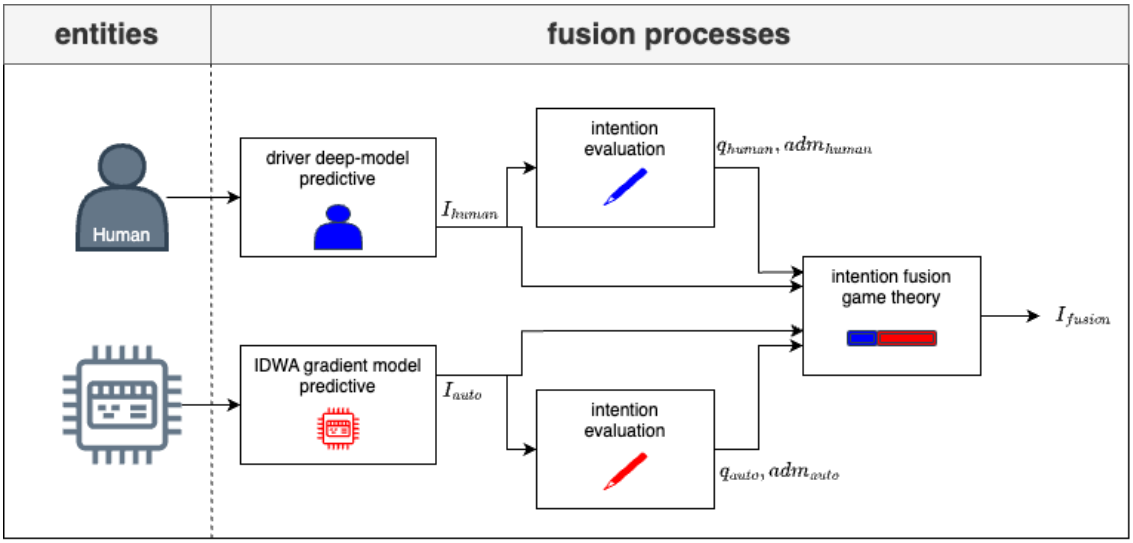


Figure 1.1: Thesis workflow: Intention fusion processes.

In this section, we introduce the base research problem of this thesis. Particularly the limitations of autonomous vehicle and the importance of the human-machine interaction in solving this problem. A detailed bibliography review will be presented in chapter 2.

1.2.2 Autonomous Vehicle

Statistical studies conducted by the National Highway Traffic Safety Administration indicate that 94% of road accidents are attributed to human error, such as poor decision-making and driver distraction[2]. Autonomous cars offer a promising solution to significantly reduce these accidents. However, the current technological limitations and public skepticism hinder the full replacement of human drivers with autonomous vehicle driven by artificial intelligence (AI). AI has not yet achieved the capability to adapt to all driving scenarios, and legal questions regarding liability in accidents remain unresolved.

As noted in Zhao et al. [11], despite significant advancements, autonomous cars are not yet fully equipped to avoid all problems. This is evidenced by the California Department of Motor Vehicles (DMV)'s analysis of autonomous vehicle disengagements (Figure 1.2), which shows that many issues are not hardware failures but stem from the autonomous system's perception and decision-making capabilities[3]. This highlights that the primary challenges in implementing autonomous systems lie within the software responsible for environmental analysis and decision-making processes.

The level of vehicle autonomy, as defined by the Society of Automotive Engineers (SAE), varies significantly, impacting both the role of the automated system in

[2] https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115

[3] Disengaged data from California gov website: https://www.dmv.ca.gov/portal/file/2022-autonomous-vehicle-disengagement-reports-csv/

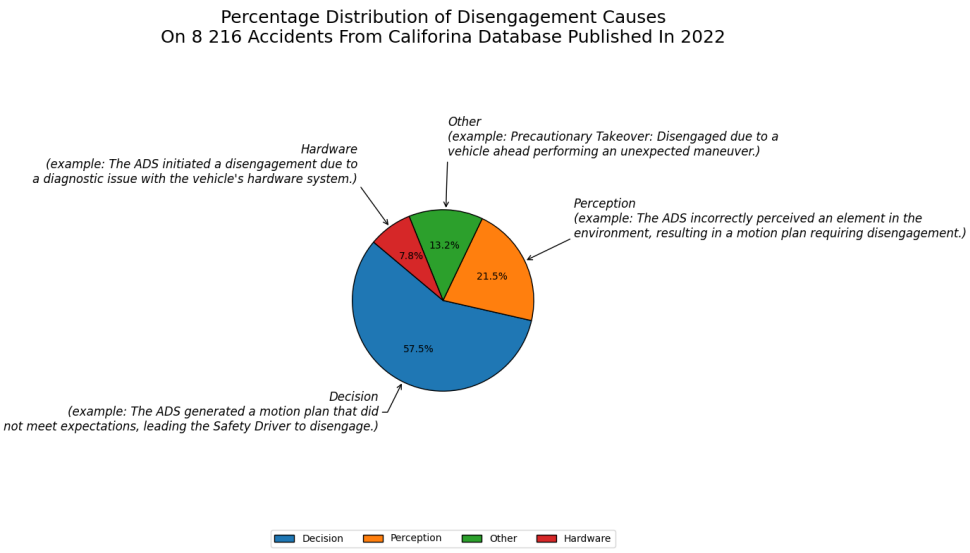


Figure 1.2: Causes of autonomous disengagement in 8,216 accidents. The other category includes accidents not related to perception and decision-making, such as those caused by erratic driving from other individuals.

driving and the associated responsibilities. The SAE levels range from level 0 (no automation) to level 5 (full automation), each representing varying degrees of human intervention and system autonomy. For example, level 1 provides driving assistance, level 3 offers conditional automation, and level 5 achieves complete automation, operating autonomously under all conditions without human intervention. At levels 1 to 3, the autonomous part contributes to driving, but under the supervision of a human to avoid any bad decisions.

1.2.3 Human-Machine Interaction

The advancement of robotics and human-machine interaction has led to the development of diverse forms of interaction. These interactions blend human intelligence and adaptability with machine precision and efficiency. The nature of these interactions varies based on their implementation and the specific tasks they are designed to accomplish. In supervised control, humans oversee and guide the automated system, intervening primarily to correct or modify actions. This form of cooperation involves the driver monitoring the vehicle's control systems and intervening when the vehicle cannot handle a situation or exhibits unwanted behavior. In a cooperative control scenario, a vehicle equipped only with adaptive cruise control (ACC) is a good illustration. The ACC automatically adjusts the vehicle's speed to maintain a safe distance from the vehicle ahead, while the human driver manages the steering. This setup clearly demonstrates cooperative control, where the automated system handles speed regulation, and the human is in charge of steering. This example aligns more accurately with the concept of cooperative control, with distinct yet complementary roles for the human and the automation system.

In assisted control, the automated system aids the driver, who retains ultimate authority and responsibility for decisions. For example, many modern vehicles feature emergency braking systems that activate in critical situations if the driver fails to respond in time. Finally, shared control represents an active and simultaneous interaction between human and automation in task performance, characterized by continuous and mutual influence. This approach is integral to the concept of shared navigation.

1.2.4 Shared Navigation Motivation

Shared navigation is a collaborative driving approach where both the human driver and the automated system are actively involved in controlling the vehicle. This method focuses on improving driving safety and efficiency by allowing both parties to contribute their decision-making skills constantly. Unlike supervisory cooperation, in shared navigation, the driver stays actively involved, capable of making decisions and overriding the automated system's actions if needed. This collaboration requires both parties to work together on the same driving task, demanding a high degree of integration and coordination.

1.2.5 Thesis Contributions

- Autonomous control:
  - Combination of visual servoing with lane detection based on a neural network;
  - Optimization of computation on obstacle avoidance, Dynamic Window Approach;
- Human driving prediction:
  - Creation of a dataset based on real vehicle;
  - Development of deep learning model, where each type of input is processed by a specific input model to reduce complexity;
  - Assessment of the importance of each variable on the quality of the prediction, across various tested scenarios.
- Intention Evaluation:
  - Create a generic multi-criteria formulation for evaluating intentions;
- Intention Fusion:
  - Definition of the non-cooperative game adapted to the intention fusion;
  - Incorporating authority variable to the game;

1.3 Work Environment

1.3.1 Simulator and Real Vehicle Setup

In order to test the various solutions proposed in this thesis, we used a SCANeR studio simulator, as well as real vehicles available in the laboratory. This section provides details on these resources.

Simulator

In the context of this thesis, work has been carried out on the professional simulator SCANeR studio (Fig. 1.3). This simulator is developed by the company AVSimulation. This simulator allows the simulation of a car's physics and its interaction with the environment. It enables the editing of maps and scenarios, including dynamic or static obstacles. This simulator has an API for Matlab, C++, and Python. It is therefore possible to interact with the car's controls using code external to the simulator. It is possible to retrieve information from the car and the sensors that the car is equipped with. We used the simulator to validate the autonomous navigation defined in chapter 3. The validation of the evaluation part (Chap. 5) and intention fusion (Chap. 6) were also tested on this simulator. The use of a simulator allows us to ensure that our system works without taking the risk of deploying our solution directly on the laboratory's cars.

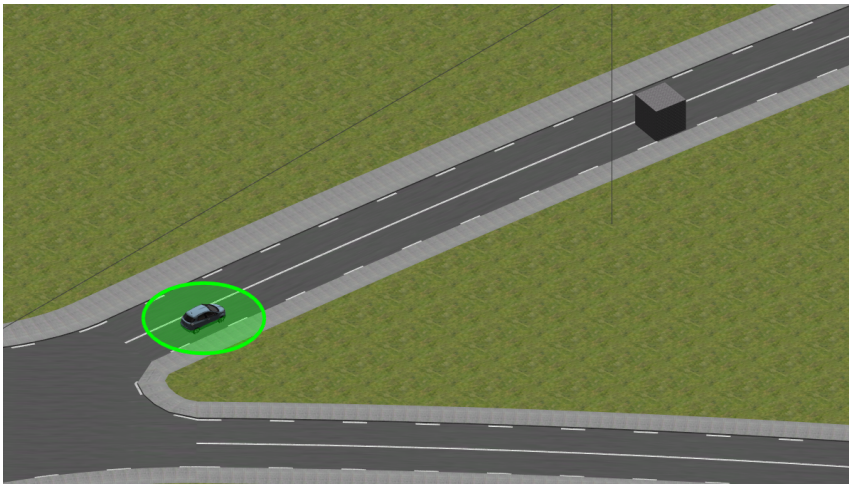


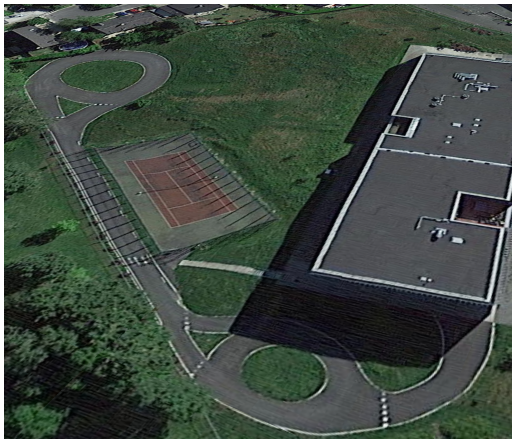
Figure 1.3: SCANeR studio illustration.

Real Vehicle

The Heudiasyc laboratory has several vehicles used for research conducted in this laboratory. These cars are Renault ZOE models. One is primarily used for data acquisition, and the other car (Fig. 1.4a), referred to as motorized, can be driven autonomously. The architecture of the 2 cars is almost identical, with the exception of the motorized part of the 2nd car. These cars are equipped with a set of

sensors for environmental perception, a 32-layers LiDAR for acquiring a point cloud of elements around the car, 4 cameras ensuring a complete view around the car. The position of the car is determined by a GNSS sensor, with an accuracy of about a cm. In addition to the GPS, we have an HD map of a part of the city of Compiègne. This map describes the road infrastructure, such as the lanes. By combining GPS position data with this map, it is possible to extract local information. In addition, data from the Controller Area Network (CAN) bus provide intrinsic car data, such as speed and access commands. An onboard computer, using the Robot Operating System (ROS) middleware[4], allows for the acquisition of data from various sensors and communication with the autonomous part.

(a) Heudiasyc car.



(b) Seville track

Figure 1.4: Overall caption for both figures.

Seville Track The laboratory has a test track, on which it is possible to conduct autonomous car tests. As shown in Figure 1.4b[5], this track consists of 2 roundabouts connected by a straight line, the distance between roundabouts is approximately 100 meters.

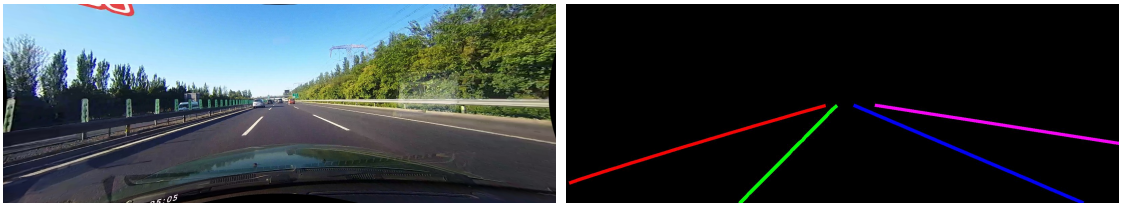
1.3.2 Data

Public Datasets

In the context of this thesis, we have used datasets that are publicly accessible. Especially for the lane detection part, where there are accessible datasets. the CULane dataset [12], which is a public dataset offering a large number of labeled lane images. The CULane dataset contains a total of 133 235 images, which were extracted from over 55 hours of videos. These videos were recorded using cameras mounted on six different vehicles driven by various drivers in Beijing. Figure 1.5

[4]ROS website: https://www.ros.org

[5]Satellite image from Google Earth, captured on December 23, 2023.



(a) (input) raw image (b) (output) mask image

Figure 1.5: Example data from CuLane dataset, image and label.

represents data from this dataset, showing the image from the camera at the front of the vehicle and the label information about this image. This dataset was used for training the neural network capable of lane detection, as discussed in chapter 3.

Own Datasets

In this thesis, to adapt our research to our car, we created a dataset specific to our vehicle. These datasets were created using a set of tools developed internally in our laboratory. Additionally, we developed other tools based on ROS for the dataset generation process, to pre-process the data, and to extract the datasets, making them suitable for training our learning model. We created a dataset specific to our cars for the model on human driving prediction (Chap. 4). Section 4.2 details the dataset created in the context of human driving. We have decided to use real data instead of simulated data, as it reflects better quality and more accurately represents driving conditions. This decision was influenced by the fact that simulator interfaces (such as joysticks or steering wheel controllers) can introduce biases, as they do not sufficiently replicate real-world scenarios.

Cluster

All the deep learning models (Chap. 3, 4) discussed in this thesis were developed and trained on datasets by myself. All of these models were created using the TensorFlow framework[6], which allows for the creation, training, and deployment of neural networks. To facilitate deployment on our clusters, we used the Docker[7] solution, ensuring the necessary environment for the code to function. To train these deep learning models, we utilized a cluster available in our laboratory. This cluster significantly enhanced our capability to handle large training datasets, especially those comprising images and point clouds. Given our computational needs, we opted for a Graphics Processing Unit (GPU) cluster, equipped with 8 graphics cards, each with 32 GB of memory, and supported by 512 GB of Random-Access Memory (RAM).

[6]TensorFlow website: https://www.tensorflow.org/
[7]Docker website: https://www.docker.com

Development and Implementation

The methods presented in this thesis were developed and implemented by ourselves. Throughout my development work, whether in simulation or with real vehicles, I was careful to create the development as generic as possible and to make the end result modular.

1.4 Publications

The methods described in this thesis have been validated through simulations and real-world experiments. These results have been published in several international conferences, as mentioned below:

- Publications 2021:
  - Shared Decision-Making Forward an Autonomous Navigation for Intelligent Vehicles [13] - IEEE SMC 2021: Shared decision-making between human and automated system, where fusion is expressed by polynomial coefficient fusion;
  - Context Modelling Applied To The Intelligent Vehicle Navigation [14] - IEEE IECON 2021: Influence of an ontology describing the driving context on the control of an autonomous system;
- Publications 2022:
  - Motion Control For Aerial And Ground Vehicle Autonomous Platooning [15] - IEEE AMC 2022: Implementation of a drone tracking solution of a ground robot using camera data based on the drone;
  - Prediction of human driving behavior using deep learning: a recurrent learning structure [16] - IEEE ITSC 2022: Prediction of human driving behavior using a recurrent neural network;
  - Dynamic Context Awareness in Autonomous Navigation [17] - IEEE SMC 2022: Driving strategy adapted by the driving context, expressed using an ontology;
  - Model-based and machine learning-based high-level controller for autonomous vehicle navigation: lane centering and obstacles avoidance [18] - IAES: Control strategy including lane keeping and obstacle avoidance based on visual servoing;
  - Gradient descent dynamic window approach to the mobile robot autonomous navigation [19] - IEEJ SAMCON 2022: Optimization of the Dynamic Window Approach strategy with gradient descent;

- Publications 2023:
  - Proposal of On-board Camera-Based Driving Force Control Method for Autonomous Electric Vehicles [20] - IEEE AIM 2023: Control of vehicle grip using a camera-based grip estimation;
  - General and Multi-Criteria Approach to Study the Admissibility and Quality of a Driving Intention [21] - IEEE IV 2023: Generalized multi-criteria formalization to evaluate a driving intention;
  - Cooperative architecture using air and ground vehicles for the search and recognition of targets [22] - IEEE ITSC 2023: Collaborative task between a drone and ground robots, aiming for the ground robots to reach positions given by the drone;

1.5 Organization

Initially, we will address in chapter 2 the works and concepts related to the contributions of the thesis. Additionally, this chapter establishes a clear position, referencing our work in the context of literature. The following chapters illustrate the contributions made during this thesis. Figure 1.6 shows a block diagram of our solution, defining the chapter of each part. Thus, we will first find in chapter 3 the definition of the intention of the automated part, based on a visual servoing controller. In chapter 4, human intention is predicted using a neural network trained on a dataset created for this part. These two intentions must then be evaluated by the method defined in chapter 5, which defines a generic multi-criteria way of evaluating an intention. From these intentions combined with their evaluation, it is thus possible to realize a fusion of this intention, thereby expressing the sharing of intention, defined by chapter 6, where this fusion of intention is applied using game theory, and whose resolution is made by game theory.

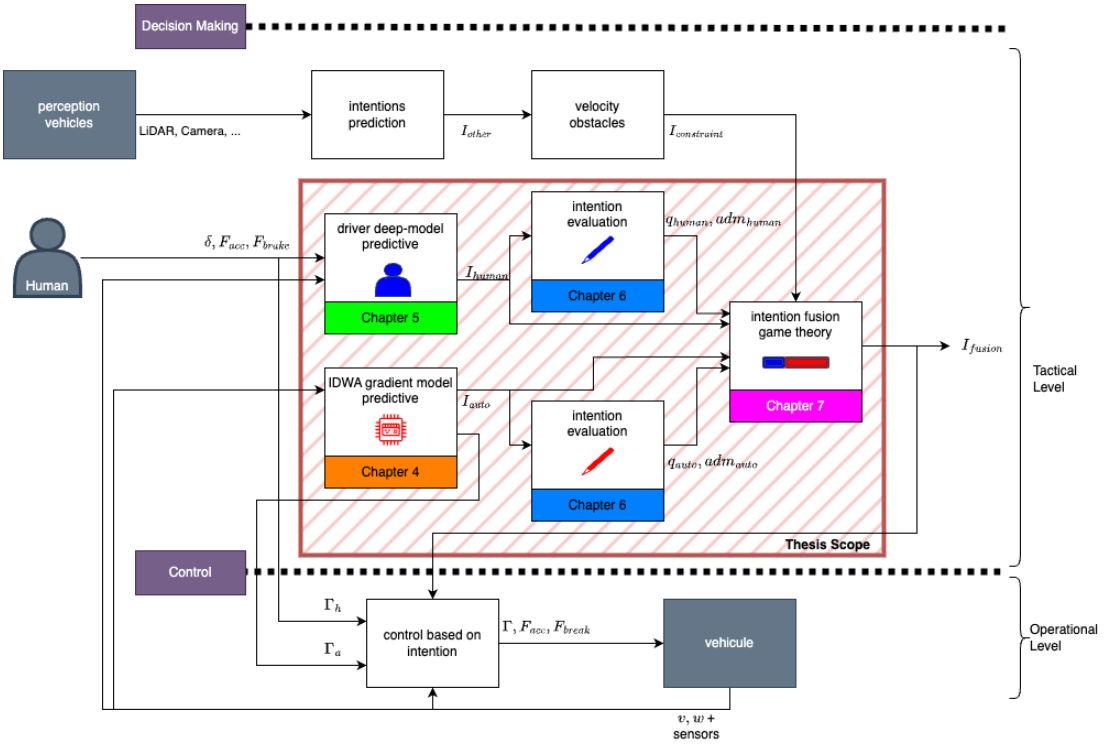


Figure 1.6: Thesis schema block combined with the thesis organization.

2 Related Work

***Abstract**: This chapter defines the works related to the thesis topic and the various concepts discussed in this thesis. For each section, the position of our work is established, in order to explain the relationship of these works with our approaches and how we distinguish ourselves.*

Contents

2.1 Shared Control

2.1.1 Concept

Shared control is an approach to facilitate control and communication between humans and intelligent machines. Although there is no consensus on the exact definition of shared control or even guidelines for its design and evaluation [23]. The concept of shared control has evolved over time. One early definition by Sheridan described it as a situation where the human acts as a supervisor for some control variables and as a direct controller for others [24]. However, this definition is somewhat vague regarding the specific variables it refers to. With the divergence of definitions of shared control, Inagaki [25] supplemented the definition by adding the concept of partitioning, meaning that the task to be performed can be divided into subtasks, with each entity exclusively working on that task. Faced with this divergence in definition, Abbink et al. [23] proposes axioms to limit the definition of shared control, as following:

1. Continuous and harmonious operation by both human and robot;
2. Active participation in a perception-action cycle by both entities;
3. The task should be capable of being accomplished individually under ideal conditions;

Based on these considerations, it is possible to distinguish shared control from human-machine cooperation and supervisory control. In human-machine cooperation, humans and machines share the same tasks and control a situation cooperatively. This involves active collaboration where both work together to achieve a common goal. In supervisory control, either a human or a machine performs the task while the other supervises. In this case, the supervisor (human or machine) does not intervene directly in the operational aspects of the task, unless necessary. When the supervisor decides to intervene and take over the task, this act of intervention and transfer of control authority can be considered as shared control. As highlighted by Abbink et al. [26], shared control can be divided into two general methods: input-mixing shared control and haptic shared control. In the input-mixing method, the final control is a result of blending commands from both the human and an optimal controller. In haptic shared control, the interaction occurs continuously with awareness of each other's choices, particularly at the force level through haptic feedback.

2.1.2 Authority Variable

The concept of authority allows adjusting the influence of each entity on the final task. In a generic sense, this notion of authority is defined by the following relationship [27]:

$$u = \lambda \cdot u_h + (1 - \lambda) \cdot u_a \tag{2.1}$$

Here, $\lambda \in [0, 1]$ represents the authority variable, where u_h denotes human control, and u_a denotes autonomous system control. The value of λ can vary: if λ equals 1, the system is in manual mode; if λ equals 0,the vehicle is in fully automated mode. Intermediate values of λ represent shared control modes. The design of λ should be influenced by various factors, including the driving context, maneuver risk, driver's state, and vehicle status. Additionally, feedback torque provided by the automated system is crucial for informing the driver of the system's intentions. This variable can be deduced from the error generated by the human driver's choices. In the case of lane-keeping assistance, if the human's control results in a significant error, as a consequence, the authority executed by the autonomous component will be greater. This variable can also be inferred by analyzing the control, using metrics such as Time To Collision (TTC).

2.2 Shared Navigation

Navigation sharing involves sharing control between the autonomous system and the human for the task of navigation. This navigation task relies on a hierarchy of different levels, as described by various sub-tasks, from the higher-level task of planning to the lowest-level task of execution. These different tasks interact with each other to achieve the overall task, which is the navigation task. This hierarchy is based on the framework proposed by Michon in 1985 [28]. Figure 2.1 displays its various levels and relation between them, and application of shared navigation is indicated.

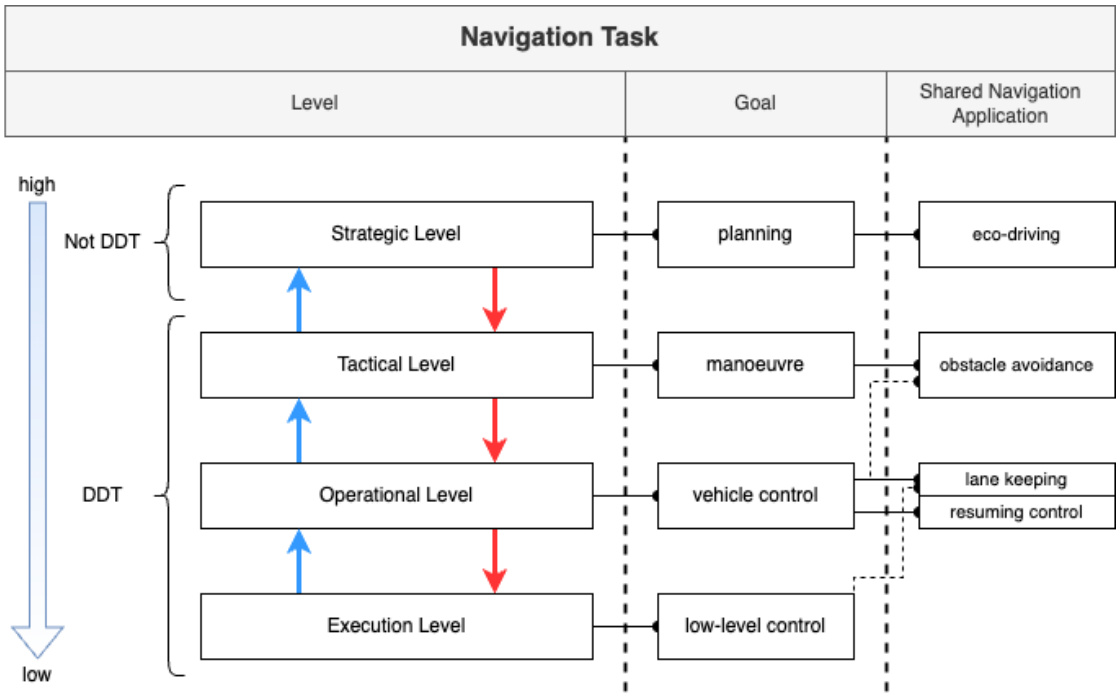


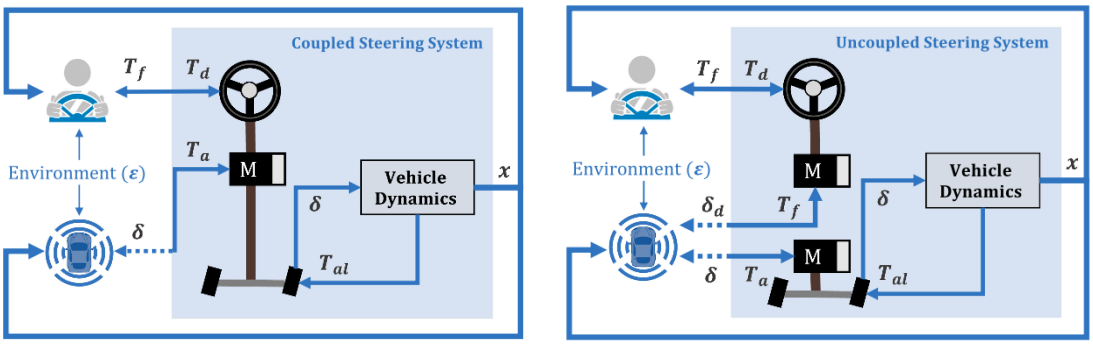
Figure 2.1: Navigation task levels relations and shared navigation applications examples.

Strategic Level The strategic level in vehicle control systems refers to the highest level of decision-making, involving long-term planning and overall task set adaptation. This level is concerned with the larger objectives and goals of the driving journey, such as selecting the destination, planning the route, and considering factors like traffic conditions, weather, and time constraints. As highlighted in the article [27], this level is typically not considered a part of the Dynamic Driving Task (DDT) in automated vehicles. This makes it difficult to find control-sharing applications at this level. However, there are scenarios where a strategic component plays a significant role, and one such example is eco-driving. In such cases, the studies lean more towards assisting the driver rather than sharing the driving task. The articles [29, 30] define haptic systems to alert the driver to fuel-consuming, which can impact the route planning.

Tactical Level This level focuses on specific actions such as initiating a lane change or adjusting the following distance with the preceding vehicle. It represents an intermediate level between the strategic level, which involves planning and task set adaptation (for example, recognizing that an intersection is approaching and that actions such as stopping and turning must be integrated into the tactical level), and the operational level, which involves continuous control and discrete maneuvers such as lane keeping and car following. In this context, the sharing of control operates at the decision-making level, as [6] demonstrates in the case of highway driving, where the sharing is done to determine whether it is preferable to stay in the lane or to initiate a lane change. At this level, most shared applications aim to avoid obstacles, as shown by [6]. Some shared control systems [5, 31] use information from this level to achieve shared control at the operational level.

Operation Level The operational level involves continuous control and discrete maneuvers such as lane keeping and car following. This level requires direct and immediate interaction with the vehicle's controls, ensuring real-time responsiveness to the driving environment. It is characterized by actions that are more reflexive and automated, requiring minimal cognitive load compared to the strategic and tactical levels. At the operational level, the focus is on executing the decisions made at the tactical level, translating them into physical actions like steering, accelerating, or braking. This level is critical for maintaining safety and smooth operation of the vehicle, especially in dynamic and unpredictable road conditions. This relates to vehicle control, specifically concerning the commands sent to the actuators, such as the output of the lateral error controller. It encompasses both continuous control and discrete maneuvers, including tasks like lane keeping [32–35] and car following. The article [36] proposes a solution for sharing driving responsibilities between humans and autonomous systems to navigate curves optimally.

Execution Level The execution level in vehicle control systems represents the most immediate and direct layer of interaction with the vehicle's controls. It involves the actual physical actions carried out by either the driver or the automated system,



(a) Coupled steering wheel. (b) Uncoupled steering wheel.

Figure 2.2: Example of mechanically coupled and uncoupled systems, from the article [1].

such as pressing the accelerator, applying the brakes, or turning the steering wheel. This level is fundamentally about the real-time implementation of the decisions taken at the operational level. At the execution level, the focus is on precise and accurate control to ensure the desired outcome of the operational level's decisions is achieved. There aren't direct applications that primarily exploit this level, but drive-sharing at the operational level also utilizes this level during sharing. In the case of shared navigation, with the objective of following the lane, control is shared at the operational level to determine the optimum steering angle. Based on this angle, a command is defined and sent to the steering motor, which has an impact on the execution level [27].

2.2.1 System Coupled And Uncoupled

In addition to the diversity of levels at which control sharing operates, a distinction is made between two different systems on which this sharing is applied: the coupled and uncoupled systems (Fig. 2.2). These two systems are the most utilized in the field of shared navigation. In each of these systems, the type of sharing is different. Indeed, in the case of a coupled system, the control sharing is referred to as haptic shared control, because both human and automated agents directly share the control commands. In the case of an uncoupled system, the type of sharing is input-mixing, as this type of system allows each party to express their command choices without the intervention of the other.

Haptic Shared Control & System Coupled

In this approach, shared control happens at the force level. The human uses the haptic sensory modality to share control of the machine interface with an automatic controller. This method is significant as it allows a direct and tangible interaction between the human and the automated system, thereby facilitating better understanding and response to the system's intentions and actions.

In the case of a coupled system, the entities are mechanically interconnected, meaning they share the same commands, and their interactions between each entity are facilitated through force feedback [37]. The most prevalent system is the coupled steering wheel. In this scenario, the steering wheel of the human driver and the motor of the autonomous system share the same steering column, which is directly connected to the vehicle. When one of the two entities applies force to the steering wheel, the other entity directly senses the effort exerted by the other entity.

This type of system is found in other domains of control sharing applications, notably in cobots [38], where the operator directly manipulates the robotic arm.

Input-Mixing Shared Control & System Uncoupled

In the input-mixing approach described in article [39] on neuromuscular analysis, command fusion is performed in a weighted manner. Initially, the system favors autonomous control. Unlike haptic shared control, each entity can propose a command that is not noisy due to the other's command. In this case, an additional haptic feedback is provided to inform the driver of the autonomous part's choice. In the article [31], the haptic feedback for the driver in this system is made through the reaction torque generated on the steering wheel, allowing the driver to physically feel the suggestions or corrections of the assistance system.

2.2.2 Shared Navigation Applications

Navigation sharing in autonomous driving systems is primarily applicable in three distinct scenarios:

1. **Lane Keeping:** The human driver and the autonomous system collaboratively work to maintain the vehicle within its lane. This joint effort ensures smooth and consistent lane adherence, enhancing road safety and driving precision [4, 32, 33, 35, 40, 41].

2. **Obstacle Avoidance:** Shared control plays a crucial role in ensuring the vehicle's safety, especially in avoiding obstacles. In such cases, the system may involve changing lanes if required to navigate effectively around obstacles. This collaborative approach ensures that both immediate and safe responses are made to unexpected road conditions [5, 6, 31].

3. **Control Resumption:** This scenario involves the transfer of authority between the human driver and the autonomous system. The shared control system facilitates a smooth transition, ensuring that the switch between autonomous and manual driving is seamless and secure, thereby maintaining a consistent and safe driving experience [7].

Lane Keeping

The main technique of lane keeping [4, 32, 33, 35, 40, 41] shared control consists of defining a driver model and a vehicle-road model (VR model) to establish the

state space, the global model is called Driver Vehicle Road model (DVR model). In this way, with a DVR model, it is possible to model the interactions between the driver, the vehicle, and the road. Then a controller is then applied with the goal of controlling the vehicle within the constraints defined by this state space.

$$\dot{x} = Ax + B_1u + B_2w$$
$$z = Cx + D_1u$$

This relation formalizes the state space. The first equation defines the state equation. It describes how the system's state x changes over time, with A being the matrix that defines the internal relations between the different state variables. B_1 is the control input matrix u, showing how the controls impact the system's state and B_2 is the matrix of external disturbances w affecting the system's state. The second equation describes the output equation; it depicts how the observable outputs of the system z depend on the state variables x and control inputs u. With C being the output matrix linking the state variables to the output and D_1 the direct transmission matrix that directly connects the control inputs to the output.

The objective of the controller applied to the state space would be to optimize lane-following accuracy, minimize the error with respect to the center of the lane, and improve comfort by reducing the lateral skid angle, lateral acceleration and steering speed. In addition, the controller aims to minimize conflicts between the autonomous system and driver actions, seeking a balance between driver inputs and autonomous system corrections to keep the vehicle safely and stably in the lane [23]. This can be achieved by designing a controller that takes into account both measurements of the vehicle's current state (such as lateral position and orientation in relation to the lane) and driver actions (such as steering wheel movements). Using modern control techniques such as predictive control, state feedback control or optimal control, the controller can calculate the appropriate control inputs for the steering system and other vehicle actuators to achieve these goals, while harmonizing the interaction between the driver and the vehicle control system.

In this application case, solutions [4, 32, 33, 35, 40, 41] typically propose defining a driver-vehicle-road (DVR) model. This model is composed of two parts: a driver model aimed at approximating human behavior in a certain situation, combined with a vehicle-road model. On the overall DVR model, a controller is applied to best satisfy the two previously mentioned models.

The articles [32, 40, 41] define a driver model based on visual (cognitive) and sensorimotor information [3], as shown in Figure 2.4. This driver model is divided into two parts: a visual part, which aims to adapt control based on the center of the lane, and a sensorimotor part, which simulates human muscular behavior, essential for replicating physical reactions such as steering torque. The visual part of the model approaches human perception by modeling how the driver perceives and interprets elements near and far relative to the vehicle and the center of the lane. This modeling is crucial for understanding how drivers adjust their direction and position on the road, taking into account different visual perspectives. As shown in Figure 2.3, two points are placed: a near point and a farther point. From

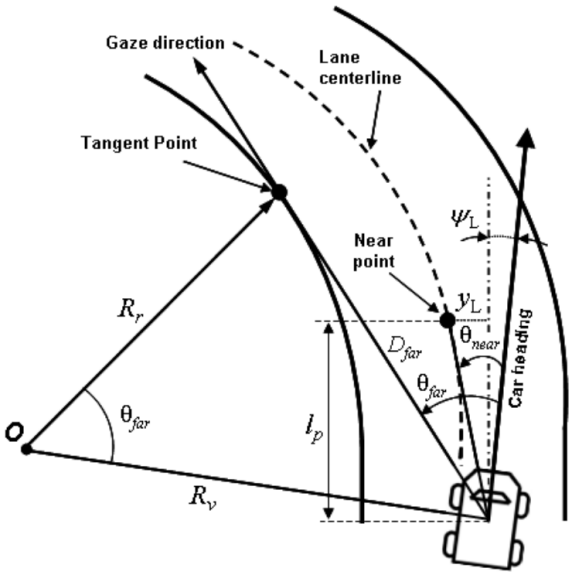


Figure 2.3: Visual information approach. From article [2], this diagram represents the near and far points, exploited by the driver model.

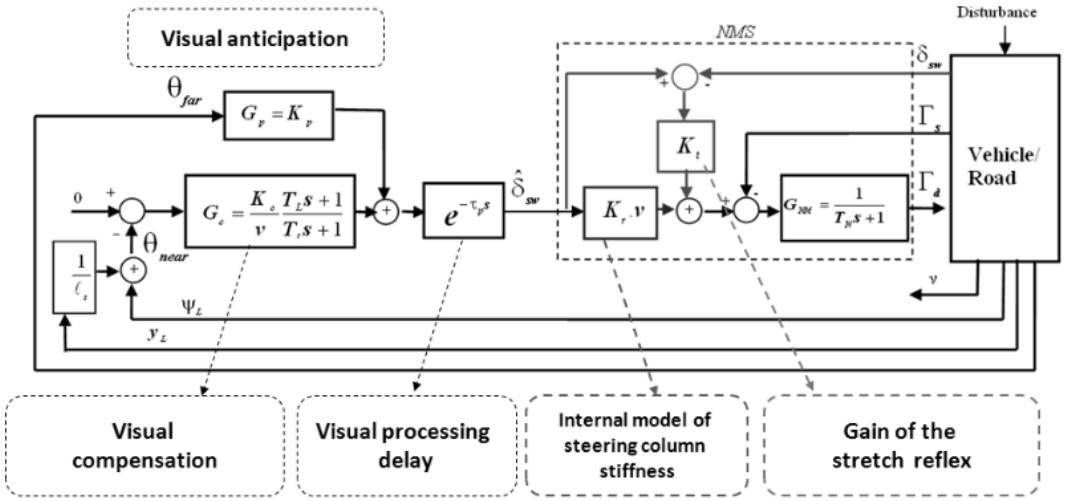


Figure 2.4: Driver model design by [3]. This diagram represents the modeled parts (visual and neuromuscular) exploited by the driver model.

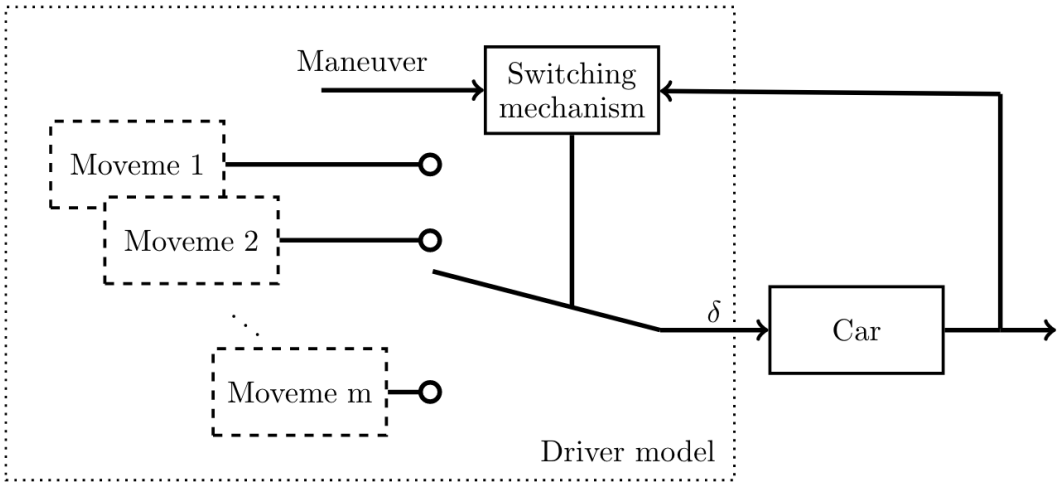


Figure 2.5: Driver model based on movemes, from [4].

these points, it is possible to calculate the angle between each of these points and the heading of the car. Taking these two points into account is important, as the near point allows for compensatory action to ensure the driver positions the car at the center of the lane, while the farther point enables the inclusion of anticipatory behavior for upcoming actions, particularly anticipating the curvature of the road. The integration of these two aspects allows for precise calculation of the steering torque that the driver is likely to apply, facilitating a more intuitive and natural interaction between the driver and the vehicle control systems.

To develop these driver models, a set of hyperparameters based on real data is used, adjusted to best represent human behavior [40].

In a similar approach, the article [4, 35] also uses a sensorimotor model to represent the human, employing a set of sub-models, or movemes. For each maneuver, a moveme is associated, to model human behavior according to the situation. Thus it is possible from the selected moveme to represent the torque used by the driver on the steering wheel. As shown in figure 2.5, a switching mechanism allows for the selection of the moveme to be applied to the car. Thus, the group of movemes combined with the switch represents the driver model.

The vehicle-road part is defined by a dynamic bicycle model combined with road information, providing a simplified yet effective representation of a vehicle's dynamics.

A controller is applied on the state space defined by the DVR model, incorporating both the driver and the road-vehicle models. For example, an H2 controller [42] is applied to the global DVR, encompassing the driver model and the vehicle-road model. The H2 controller is used to manage the complex interaction between a human driver and an automated system, the cooperation between human control and automated control, while minimizing disturbances. In the article [32], the H2 controller is applied to minimize the heading error as well as the lateral deviation. It also minimizes assistance sharing $\Gamma_a - \Gamma_d$ and cooperation quality $\Gamma_a \times \Gamma_d$. Where

Γ_a and Γ_d represent respectively the automated and driver torque.

Other articles [41] utilize an MPC (Model Predictive Control) controller. The use of this controller necessitates modifying the previously described approach to include a predictive model and the addition of constraints. In the context of this article, the authors add constraints on lateral position, slip angle, and control input.

Obstacle Avoidance

In this section, we explore approaches to shared control in the context of obstacle avoidance and lane keeping. Shared control involves a cooperative interaction between humans and autonomous systems to ensure safe and efficient driving experiences. In the case of shared control for obstacle avoidance and lane keeping, as outlined in [33], the approach delineates two distinct control levels. The first level focuses on path planning to circumvent obstacles on the road and lane-keeping, while the second level is dedicated to steering control adapting control on information from the previous level. Control sharing can then be found at the level of path planning sharing and/or at the operational level, where the focus is on the torque to be applied to the car's steering wheel.

An implementation of navigation sharing methodology at the operational level, exploiting information from the tactical part, is presented in [5]. However, it is important to note that in this case the sharing aspect is not realized at the decision-making level. The approach, as detailed in [5], focuses on assessing the human's choice at the tactical level. It involves deducing whether the human intends to maintain their lane or perform an obstacle avoidance maneuver. This decision is considered along with the intentions of the autonomous system, and the operational level sharing is determined. The path planning in this approach is done using data provided by sensors and devices such as front cameras and GPS, followed by the application of a linear-time-varying MPC controller to derive the control command for the vehicle. Additionally, a machine-learning model, is used to understand the driver's intentions based on deviations from the road. The cooperative coefficient in this approach depends on the Time to Collision (TTC) metric and the η metric. η metric calculates the error between the vehicle's path and the planned path, as well as the lateral deviation of the driver's path from the planned path. This cooperative coefficient determines the level of cooperation between the human and the autonomous system. As highlighted in Figure 2.6, the cooperative coefficient is deduced using fuzzy logic based on the metrics TTC and η. This coefficient is subsequently utilized, during the command calculation, to control the impact of each entity (human and autonomous system).

TABLE I
THE SHARED FUZZY CONTROLLER RULE BASE

ttc	Performance evaluation index η			
	S	M	MH	H
HD	M	MH	H	H
D	M	M	MH	H
S	S	M	MH	MH

Figure 2.6: Fuzzy table from [5]. This table defines the level of sharing between the autonomous system based on the values of TTC (HD = high dangerous, D = dangerous, and S = safe) and η.

In the article [40], the operational part is managed by two controllers: the first controller enables automated lane keeping, and the second controller facilitates shared control. The transition between these two controllers is ensured at the tactical level. The idea of this approach is to detect a conflict between the human and the automated system. In case of a conflict (for example, to avoid an obstacle), a transfer of control is realized.

Navigation sharing is also applied to robots operating in off-road terrains [43]. In this article, the space is defined by homotopy space, in which the robot can evolve. From this distribution of space, an optimal trajectory is defined by MPC to ensure maximum stability. This stability depends on the threat value, which here refers to the front wheel slip. In this context, the control sharing is only at the operational level, where the control sharing is only carried out if the vehicle's threat value exceeds a certain threshold, and in this case, the upstream-defined trajectory will impact the driving of the robot.

In other works, the shared navigation can be applied on both tactical and operational levels. In this approach [6], both entities propose a trajectory. The trajectory from the autonomous part is predicted based on the lane center. The trajectory for the driver is predicted based on the torque exerted on the steering wheel combined with a bicycle model and a CTRV (constant turn rate and velocity) model, allowing the determination of the trajectory desired by the driver. The final trajectory is a combination of the human and automated trajectories, with the human impact defined by an authority variable, whose value reflects whether the driver is distracted or not. Once the trajectory has been defined, a shared control is implemented at the operational level. This sharing is substantially similar to the lane-keeping approach previously discussed in the section (Sec. 2.2.2). However, unlike in prior studies, the trajectory to be followed is the one determined by the tactical layer, rather than being centered on the lane. Figure 2.7 represents the control sharing of this article, showing the relationship between the entities, as well as the levels of navigation. Once the trajectory is planned, a controller then calculates how to achieve this trajectory while considering the torques exerted by the human.

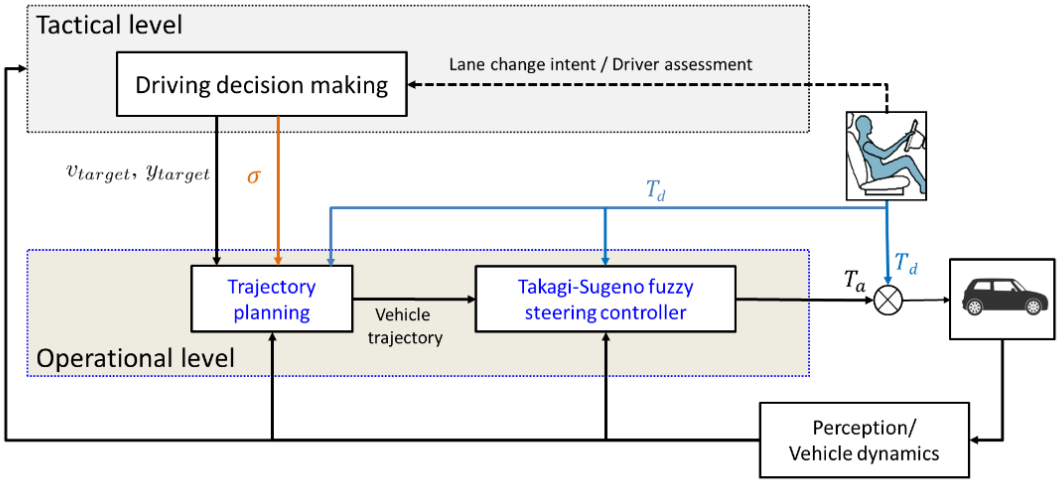


Figure 2.7: Shared control structure defined by [6]. This diagram shows the sharing carried out between the automated system and the human in terms of the trajectory and the control applied to the vehicle.

Resumption Control

During a control authority transfer, shared control occurs to smooth the transition between the two entities. In the article [7], shared control is exploited in the transfer phase between the autonomous system and the human, this sharing via Haptic Shared Control. This method is particularly important in Levels 2 and 3 of automated driving where drivers are required to assume control from the vehicle. In this application, the control sharing focuses on the sharing of the steering wheel, where this driving sharing is subject to a gain, the purpose of which is to make this gain tend towards 0 within a set time to assign authority over the car to the human, thereby improving steering stability. Figure 2.8 represents the authority transfer with shared control method.

2.2.3 Game Theory Approach

The shared control can be approached in the manner of a non-cooperative game, where the human and the autonomous system find themselves in competition. As presented in [31], the steering torque control sharing is achieved using game theory in a non-cooperative game framework. For each entity, human and automated, a path is predicted. For the automated part, this prediction is based on a Vehicle-Road model, and for the human, this prediction is made using a deep-learning model. Each entity is considered as a player, and each player's strategy is defined by torque inputs. This loss function evaluates the difference between their predicted path and the human-machine path, adding a penalty to the desired torque. In this way, it is possible to influence one player more than another to have higher torque inputs. The definition of this penalty is based on fuzzy logic which sets this value

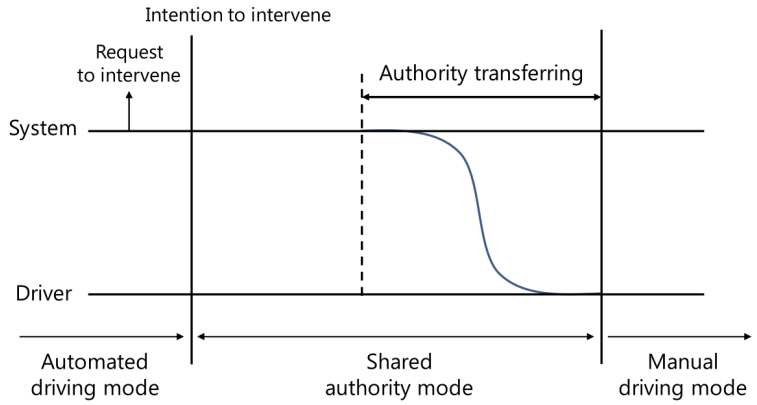


Figure 2.8: Authority transfer from autonomous system to driver defined by [7]. It is within this range that the control sharing is carried out, to ensure a smooth transfer.

according to the degree of conflict and risk.

Other approaches also utilize game theory, as seen in [44]. In these approaches, the game is also non-cooperative, and sharing occurs at the level of linear and angular velocities. In contrast to the previous approach, the strategy is shared in this context and it's applied a the tactical level.

2.2.4 Position To Related Works

In the context of our research, we focus on sharing at the tactical level. Where the goal is to share the intention of each entity, human and automated. Our work distinguishes itself from lane-keeping approaches [4, 32–35, 40, 42], as we are interested in intentions and not commands (such as the torque of each entity). Our work exploits a similar idea from the article [6], where as previously described (Sec. 2.2.2), an initial sharing is done at the tactical level, where a trajectory is planned based on the path followed by the human and the automated system. This approach is interesting, as it describes the intention of each entity and how to achieve it. However, we distinguish (Chap. 4) ourselves from this approach, particularly in the way the human's intention is calculated, which is approached by exploiting the exerted torque combined with a CTRV model. More recent work [31] orients the study of the human driver using a neural network, we have taken this direction, but unlike this article, our network allows defining the intention of speeds and not maneuvers, giving richer information. Because of this richer information, compared to a constant speed projection, we aim (Chap. 5) to provide a more precise evaluation that exploits these intentions to assess their impact. The fusion resolution is inspired by the work [44], applying game theory (Chap. 6) to the intentions of entities, unlike these approaches we define a non-cooperative game without strategy sharing, ensuring a Nash equilibrium and including authority variables, drawing inspiration from the work done by [31]. Unlike them, our game theory is applied to share intention

and not command. It seems important to us to separate the strategy of each entity, in order to ensure a mixing-input intention. Moreover, we distinguish ourselves from other approaches because our authority variable is dependent on the authority variable from the previous execution. As shown in the article [7], the transfer of authority is a complex task that needs to be done smoothly. Based on this idea, we have then conceptualized the sharing of intention based on the previous authority variable to ensure a smooth and non-unpleasant transfer for the human.

2.3 Autonomous Control

In the context of this thesis, due to our challenge of achieving control sharing, whether it occurs at the tactical or operational level, we are interested in local navigation.

In systems where there's a distinct separation between high-level route planning and low-level control, the navigation process involves two key components. Firstly, a path planner creates the overall route using a map. Secondly, a local reactive navigation system, focused on immediate decision-making, generates the vehicle's next action. The global plan serves as a guide for this local navigation. Local navigation operates within the control space, dynamically creating actions that are feasible for the vehicle's immediate circumstances [45].

At the local scale, navigation is reactive and depends solely on information from sensors. These sensors allow for knowing the vehicle's [46] (GPS, GNSS) and also provide information about the surrounding elements, such as LiDAR, which can estimate the position of obstacles and is increasingly used for object recognition [47] and tracking [48]. The camera, after analysis, provides information about the road [49] and the environment where algorithms for lane detection, pedestrian detection [50], and object detection can be applied.

With knowledge of the overall goal and local information, it is possible to control the vehicle in a way that maintains the lane while avoiding obstacles. This local navigation technique can rely on various approaches, such as trajectory-based navigation [51], where both road and obstacle information are taken into account. Visual servoing techniques [10] can also be used, where information from cameras is exploited to steer the vehicle in its lane. These data can be combined with LiDAR, allowing for the consideration of obstacle positions to make this local navigation safe and secure [10].

2.3.1 Lane Detection

Lane detection is a crucial aspect in the development of autonomous or assisted vehicles. Accurately locating the lane to follow is essential. The evolution of lane detection technology began in the 1990s [52]. This technology plays a pivotal role in driving safety, particularly in preventing unwanted lane departures [53, 54]. In clear conditions, with visible lines and favorable weather, lane detection can be rapidly and accurately performed. However, reliably detecting lanes in all scenar-

ios, especially in adverse weather conditions or in the presence of shadows, presents significant challenges due to factors like reduced visibility and contrast issues. Before the advent of deep learning, the most widely used method was the geometric approach [55–57].

In this approach, the first phase involves preprocessing the image to make the lane lines as visible as possible [57], such as transforming the color space to Hue Saturation Lightness. This aims to adjust the saturation of the image to enhance the visibility of pixels representing lane lines. From the pre-processed image, edge features are calculated using Gaussian filter, Steerable filter, or Gabor filter. These traditional methods, while effective in certain conditions, often struggled with complex scenarios like varying lighting or weather.

The exponential advancement in machine learning and deep learning has enabled the development of more versatile models capable of detecting lanes in a broader range of situations [58], including the challenging scenarios described earlier. Therefore, employing a deep learning model, particularly with the advent of Convolutional Neural network (CNN) [59], is often seen as a highly effective solution. CNN and other deep learning models have revolutionized lane detection by offering improved accuracy and adaptability, far surpassing the capabilities of traditional geometric approaches. Every convolution layer is composed of a set of filters, each filter executes a convolution product to extract a feature (for example, a vertical line, a horizontal line). Thus, each filter creates a feature map in relation to its input and its kernel. These layers are usually followed by a pooling layer, whose purpose is to perform dimensionality reduction. In this way, the input image is reduced to a smaller dimension, and it is possible to connect a neural network to this reduced dimension, which will exploit the features resulting from the convolutions. It is important to understand that thanks to this approach, the number of parameters in the neural network is reduced, because each filter is defined by its kernel parameters (often of size 3x3 or 5x5). Therefore, the execution of a large model requires fewer memory resources and computational power compare to traditional neural network.

2.3.2 Position To Related Works

In the works on control sharing, a large part of these studies [4–6,31–33,35,40,41] rely on the definition of a VR model, which includes a bicycle model combined with road information. In our approach (Chap. 3), we have based our strategy on visual servoing and obstacle avoidance [10]. We were inspired by this approach, but unlike the defined approach, we wanted to use a deep-learning model to detect lanes. Additionally, we propose an optimization solution to achieve the strategy that the car should apply, unlike the initial approach which proposes a brute-force resolution. In the works [44], the intention was then deduced from the human prediction model applied to the car. In our case, we wish to distinguish these two models. That is why we have made a prediction of the intention of the autonomous system, based on its strategy. Our motivation for dissociating human and automated models is to avoid redundancy of error in the final system. For example, in our human driving behavior

prediction system (Chap. 4), we utilize information other than lane detection, in order to avoid propagating any error throughout the entire sharing.

2.4 Human Driving Intention

In the context of shared control, a set of works utilizes a model known as DVR, which consists of a human driver component. The purpose of this component is to emulate human behavior based on a model. As explained in the shared navigation section, many of these works rely on sensorimotor models with the goal of modeling human behavior. In our approach, we model human behavior using a deep learning model. The idea is to explore an alternative way of approaching this behavior. Currently, there are studies focusing on human driving, and most of these studies primarily focus on an external perspective to the ego-vehicle. In other words, these models are based on observations and data outside the vehicle. However, some studies utilize an internal perspective.

As pointed out by [60], in comparison to pedestrians, the behavior of drivers is more predictable and less random because it is constrained by the behavior of surrounding vehicles, traffic rules, and road geometry.

In a generic context, the prediction of a vehicle is defined by the sequence X as follows:

$$X = \{x_t, \ldots, x_{t+m}\} \tag{2.2}$$

Here, x_t represents a state of the vehicle, such as a position, velocity, or maneuver.

2.4.1 Models And Data

This section describes the types of human driving behavior prediction models defined in the literature.

Models

Parametric and Non-parametric The human driver predictions models are divided into two categories parametric models and non-parametric models. The parametric model regroups physic models [61]. These models are based on theoretical concepts. These models have the particularity of being fast in execution, but are nevertheless too rigid models that are not able to adapt to a set of diverse and complex situations. The non-parametric models regroups models based on data. Due to the complexity of the problem and the very rapid development of deep learning in recent years, research on the prediction of human driving has been strongly oriented towards the use of neural networks [61]. Thus, the model, unlike parametric-model based solutions, is able to make a complex representation of the data and finding relationships, it is however important to note that in this case the model may be biased by its learning [62]. Because of its temporality, the prediction of driver intentions could be modeled by a recurrent neural network [63]. These neural networks have the particularity of being able to process data series

(for example, temporal, sequence of words). These networks have the characteristic of having a kind of memory that allows them to retain information about previous inputs. Thus, the predicted output depends on the input sequence. Thus, the model does not only depend on current data but also on previous data and is able to establish relationships between these data. Avoiding the problems associated with vanishing/exploding gradient learning [64], recurrent networks are replaced by Long Short-Term Memory (LSTM) cells [65]. This problem is particularly evident in long sequences. To remedy this, LSTM networks have been designed to filter the information that is useful to retain and that which should be forgotten. This mechanism can be implemented thanks to its structure of gates and cell states.

Classification and Regression The prediction of human driving can be expressed in several forms, the next manoeuvre that the driver will perform [66, 67] or the state of the vehicle in the near future [68, 69]. Depending on the nature of the prediction, the model must be adapted, the prediction of the maneuvers is usually based on classification, in this case the model is able to classify among a fixed number of maneuvers that the driver will perform. In the case of vehicle state prediction, the model estimates a sequence of states (position or speed); thus, the model must performs a regression and gets as close as possible to the numerical values (defining the vehicle state). Some models [70] combine the two types of prediction by giving for each manoeuvre a sequence of possible states of the car.

Data

Data exploitable for the prediction Most of the research [63, 66, 69, 71] on human driving prediction relies on forecasting driving behavior (maneuvers or trajectories) by leveraging dynamic vehicle data to predict acceleration, speed, positions, angles, as well as information about surrounding vehicles. Other models utilize more complex information; the article [68] addresses a bird's-eye view of the situation with information about local agents (vehicles). In this model, [72] predicts car trajectories by exploiting point cloud data from LiDAR, noting that this kind of network requires more substantial computational resources. In this paper [73], the model does not directly utilize raw radar data but rather information derived from radar, such as the position and speed of the vehicle in front, deduced from radar data combined with the ego-vehicle information. In the article [74], an occupancy grid is used instead of lidar-derived information. This reduces the amount of information transmitted to the model.

Datasets Interstate 80 Freeway [1]: this dataset was generated from a 45-minute video, with the camera positioned atop a building. The dataset contains dynamic vehicle data, including velocity and position.

[1]Interstate 80 Freeway's dataset website: https://www.fhwa.dot.gov/publications/research/operations/06137/index.cfm and https://data.transportation.gov/Automobiles/Next-Generation-Simulation-NGSIM-Vehicle-Trajectories/8ect-6jqj

US Highway 101 [2]: this second dataset is quite similar to the previous one. The process is analogous, with a camera placed atop a building to record data. This dataset was recorded in Los Angeles.

These datasets focus on external perceptions. In our research, we are interested in vehicle intrinsic data. Interior datasets are not available, and creating a prediction model in this context can be sensitive to the data and sensors used. Custom dataset creation is a commonly [71,75] used technique in the literature, adapting the dataset to the equipment available to the researcher publishing the work.

2.4.2 Ego Vehicle Prediction

The article [75] deals with the prediction of the ego vehicle using an LSTM network. The application of this model is to make trajectory predictions in order to optimize energy consumption at the local level. In this approach, the model infers 30 seconds of data in advance to make a 15-second trajectory prediction. In this study, the model leverages the vehicle's dynamic data, relative positions with the vehicles in front and behind, and the ego vehicle's position relative to the road. Figure 2.9 shows the model defined in the paper. The model is defined by 2 LSTM layers, and the output of the recurrent network is then connected to a classic feed-forward layer to make the prediction. The network dimension was tested using the HORD algorithm [76], whose goal is to optimize the model to ensure a higher-quality prediction accuracy. In a similar approach [71], this article focuses on predicting the driving behavior of an ego-vehicle. Unlike the previous article, the objective here is to predict, based on the vehicle's state, which exit the driver intends to take at a roundabout. In this case, we are dealing with a classification problem.

2.4.3 Position To Related Works

In our work, we decided to use a deep-learning network, predicting the driver's intention over a time horizon. This approach allows us to know both the maneuver the driver wants to make and how to achieve it. In the approaches made in control sharing, the models [31] define the maneuver and from a model, the way to achieve this maneuver is deduced and approached using a CTRA or CTRV model. We show in chapter 4 that this approach can induce a significant error. Moreover, in our work, we have defined a new type of model, capable of handling different nature of information, and part of the pre-processing is adjusted to the learning of the model in order to present the most raw data possible to the model and adjust these data, contrary to the approaches [75] where the data is pre-processed upstream. We are following an approach similar to the model [71] (even though it is applied to a maneuver classification), we do not implement an output fully connected to the LSTM layers, as this approach significantly impacts the number of network parameters and its runtime execution.

[2]US Highway 101's dataset website: https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm and https://data.transportation.gov/Automobiles/Next-Generation-Simulation-NGSIM-Vehicle-Trajectories/8ect-6jqj

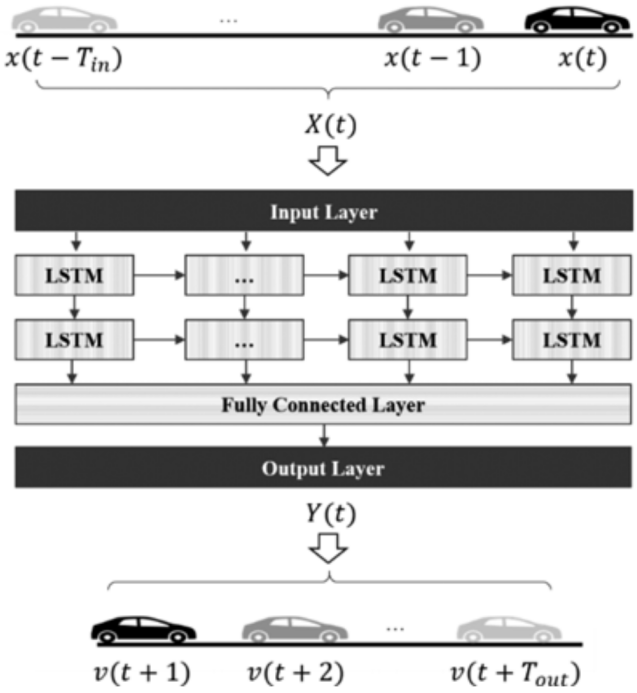


Figure 2.9: Model defined and utilized in the article [8].

2.5 Evaluation Intention

2.5.1 Evaluation Situation

The literature [77] defines a large number of metrics for quantifying a situation. Many of these metrics are related to the safety of the ego-vehicle and the surrounding vehicles. For example, one such metric is the Time To Collision (TTC). This metric calculates the time to collision between the ego-vehicle and the vehicle in front, assuming constant velocities. However, this indicator does not provide direct information about whether the situation is problematic. It is necessary to compare this value to reference values to critically assess the situation. [78] defines a reference variable, Time to Accident (TA), at 1.5 seconds, ensuring that the driver can brake in time to avoid a collision. This value includes a range of scenarios at different speeds. Other studies decide to define this reference value based on reaction time and deceleration rate data [79].

2.5.2 Evaluation Sequence Commands

In our approach to intention quantification, we drew inspiration from the principles of reinforcement learning, as detailed in [80]. Specifically, we utilized the concept of the Q function, a key element in reinforcement learning algorithms, which evaluates the value of taking a certain action in a given state. This Q func-

tion is derived using the principles of the Bellman equation, a fundamental concept in Markov Decision Processes (MDP) as described by Bellman [81]. The Q function represents the expected sum of rewards when an agent starts in state s, takes action a, and follows a specific policy (π):

$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s')Q(s',a')] \tag{2.3}$$

Where:

- $Q(s,a)$ represents the value of the state-action pair (s,a).
- $P(s'|s,a)$ is the probability of transitioning from state s to state s' when taking action a.
- $R(s,a,s')$ is the immediate reward obtained by taking action a from state s and transitioning to state s'.
- γ is the discount factor representing the importance of future rewards compared to immediate rewards.
- $\pi(a'|s')$ is the probability of selecting action a' in state s' according to policy π.
- $Q(s',a')$ represents the value of the state-action pair (s',a').

This formulation enables us to evaluate the impact of an action on a state by analyzing, using a dynamic programming approach, the long-term effects of that action. It considers all future actions, which are influenced by the policy π defined within the formula.

2.5.3 Position To Related Works

In the context of our thesis (Chap. 5), we have exploited this formulation, where the policy is defined by the intention of the drivers. In this way, we can evaluate each command defined by this intention. Figure 2.10 illustrates how formula 2.3 is applied. In the same figure, there is the adaptation to our problem, where the policy follows the intention. Thanks to this formalization, we have generically formulated how to evaluate an intention according to several criteria. In this manner, it is possible to assess the intention using different metrics.

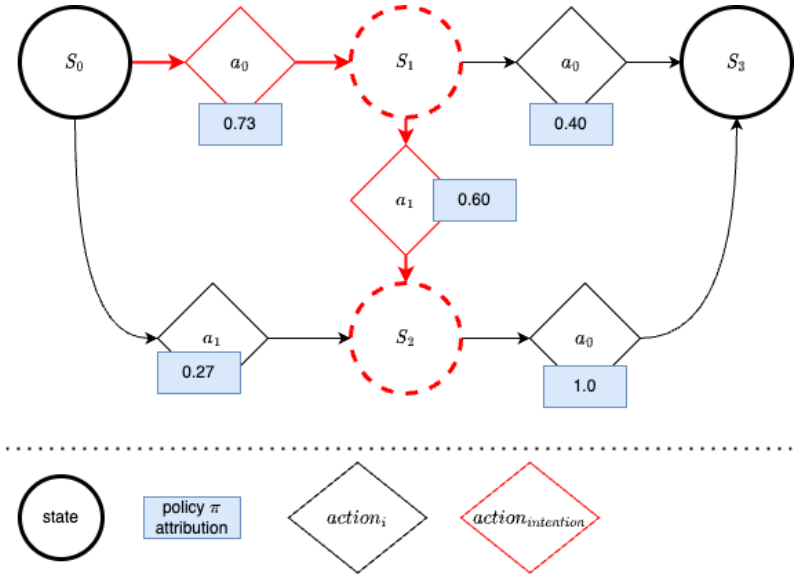


Figure 2.10: Markov Decision Process diagram showing our position with the formulation defined Eq. 2.3.

3 Autonomous Control

***Abstract**: In this chapter, we discuss the control and strategy employed by the autonomous system. As part of our shared driving approach, addressing this aspect is crucial for expressing the choices made by the autonomous system. Within this chapter, we implement vehicle control using visual servoing combined to an obstacle avoidance strategy known as the Dynamic Window Approach (DWA). Additionally, we present our contributions related to the lane detection model and strategy optimization. An evaluation in simulation was conducted to demonstrate the effectiveness of our approach.*

Contents

3.1 Motivation

In the context of shared control between humans and autonomous systems, it is crucial to establish an effective control loop for autonomous system operation. Unlike supervised driving or collaborative driving, in the case of shared control, both entities are involved in driving the vehicle. Initially, proposing a new autonomous navigation method to be integrated into the vehicle was not considered as a research objective of this thesis. In fact, we believe that there are many existing autonomous navigation methodologies that could be used. Nevertheless, we initially adapted local navigation approaches proposed in previous theses [9,44], which were developed at the Heudiasyc laboratory and to which we made contributions. In this chapter, we present these previous methods as well as the innovations we have proposed to make them more robust and efficient in real-time. In the work conducted in this thesis, we focused on developing an autonomous system capable of lane following and obstacle avoidance. Building upon the methods developed in previous theses, we relied on local navigation, utilizing data from a camera to detect lanes and lidar data to determine the position of obstacles in the local environment. It's important to note that this method does not depend on GPS data and maps, making the system self-contained in terms of external information. In our initial approach, the car's driving was ensured by visual servoing combined with a strategy known as the Dynamic Window Approach (DWA). Visual servoing was accomplished through geometric lane detection. As mentioned in the literature review (Sec. 3.2.4), the use of Convolutional Neural network (CNN) is more robust, which explains the application of this network (see Section 3.2.4). This lane-following method is combined with an obstacle avoidance strategy. In our proposal, we applied an optimization solution to address this strategy (Sec. 3.3.2).

3.2 Lane Keeping

The task of ensuring that the vehicle remains within its designated lane is accomplished through visual servoing. This involves detecting the lane's edges in which the vehicle is traveling and computing the necessary steering commands to keep the vehicle centered within its lane. Notably, this technique does not rely on external factors such as maps or GPS data.

3.2.1 Visual Servoing: Fundamental And Tools

Visual servoing [82,83], entails using computer vision data to control a robot's motion. This data can be sourced from a camera mounted on the robot or be derived from the robot's surroundings. The primary goal of any vision-based control system is to minimize an error $e(t)$, defined as:

$$e(t) = s(m(t), a) - s^* \tag{3.1}$$

Here, $m(t)$ represents image measurements (e.g., coordinates of interest points within the image), $s(m(t), a)$ represents visual features computed from these measurements and a vector a that contains parameters and knowledge, and s^* represents the desired values for these features. Visual servoing methods can be categorized into two primary approaches:

- Image-Based Visual Servoing (IBVS) Control: In this approach, s comprises features that are directly accessible from the image data. The error is defined as $e(t) = s(t) - s^*$.
- Position-Based Visual Servoing (PBVS): s is used to define, with the help of a model, additional information related to this feature, such as the position of elements extracted by this feature in the robot's coordinate system. Letting X represent the resulting element from the 3D model based on s. The error is defined as $e(t) = X(t) - X^*$.

3.2.2 Visual Servoing: Concept

Figure 3.1 illustrates an example of Image-Based Visual Servoing (IBVS). In this task, the objective is to position the robot in front of the door. The set of image features in this case consists of the coordinates of the door's corners, denoted as $s = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$. The error between the desired goal and the current state is defined as:

$$e(t) = s(t) - s^* \tag{3.2}$$

The task is considered accomplished when the error approaches zero, i.e., $e(t) \to 0$. To achieve this, we need to find a control command that drives the error towards zero. We can define a relation between the variation of image features ($\dot{s}$) and the velocity u_c (linear and angular) of the sensor (in our case, the camera) as follows:

$$\dot{s}(t) = L_s.u_c \tag{3.3}$$

The matrix L_s in Equation 3.3 is known as the interaction matrix [84], which expresses the relationship between the variation of image features and the movements of the sensor. By using Equation 3.3, we can derive the evolution of the error ($\dot{e}$) over time, assuming that the variation of the objective ($\dot{s^*}$) is zero:

$$\begin{aligned} \dot{e}(t) &= \dot{s}(t) - \dot{s^*}(t) \\ &= \dot{s}(t) - 0 \end{aligned} \tag{3.4}$$

So if we apply exponential decay, we have:

$$\dot{e}(t) = -\lambda L_s^+ e(t) \tag{3.5}$$

where L_s^+ represents the Moore-Penrose pseudo-inverse of the matrix L_s, defined as:

$$L_s^+ = (L_s^T L_s)^{-1} L_s^T \tag{3.6}$$

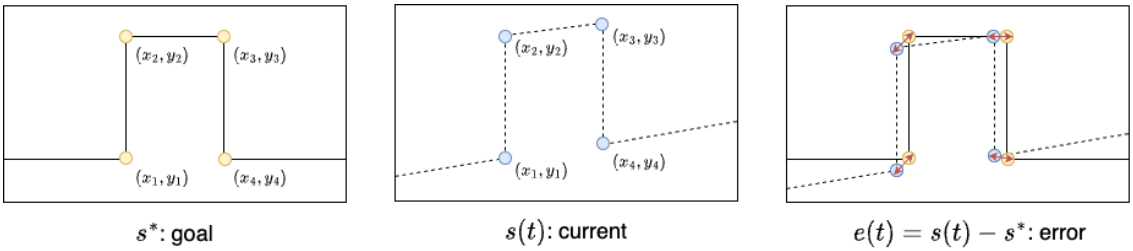
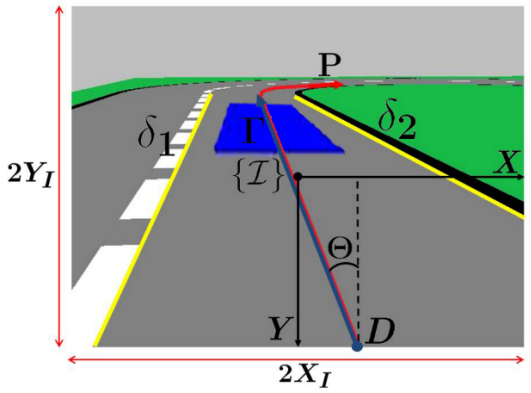


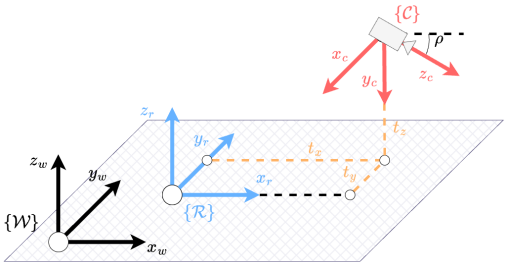
Figure 3.1: Example of a visual servoing task: in this situation, the robot has to move in front of the door.

3.2.3 Visual Servoing: Autonomous System

In the context of autonomous vehicle control, visual servoing can be utilized to guide the vehicle and keep it centered within the lane. By using a forward-facing camera, the vehicle's orientation can be adjusted by applying specific velocities (v, w), thereby ensuring the vehicle stays within its lane. In this scenario, image features are calculated in two steps. Initially, detection is performed on the image from the camera to identify the lane. Once this detection is completed, the feature controlling the vehicle can be calculated.



(a) Visual servoing features for lane following, illustration extracted from [9].



(b) Relation between robot frame $\{\mathcal{R}\}$ and camera frame $\{\mathcal{C}\}$.

(c) Representation of the elements used in the visual servoing controller.

Figure 3.2: Variables of visual servoing used to control the vehicle and keep it centered in the lane, including the definition of image features and the relations between frames.

In Figure 3.2a, lane detection is represented by the yellow lines on both sides of the lane (δ_1 and δ_2). From this detection, it is possible to define the path to follow (P), which is the path positioned in the center of the lane boundaries. Using this path, the tangent line (Γ) is then calculated. The angle between Γ and the vertical axis represents the angle Θ. Point D represents the intersection of P and the boundary of the image I, and the coordinates of this point are defined as (X, Y). Let (X_p, Y_p) be the coordinates of point D in the image frame defined by their pixel

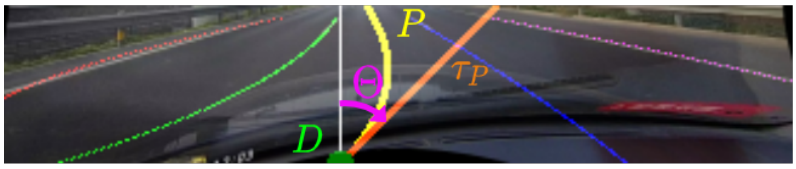


Figure 3.3: Visual servoing features computed by our development on real image.

positions. Considering the camera's intrinsic data, (f_x, f_y) as the focal lengths, and (c_x, c_y) as the principal point, the coordinates of point D can be defined in meters as follows:

$$X = \frac{(X_p - c_x)}{f_x} \tag{3.7}$$

$$Y = \frac{(Y_p - c_y)}{f_y} \tag{3.8}$$

Thus, we define the image features for visual servoing as follows:

$$s = [X, Y, \Theta] \tag{3.9}$$

The Figure 3.3 shows the extraction of features from a real image. The goal of the control is to compute an input $u_r = [v, w]^T$ to drive these features to the final configuration $s^* = [X^*, Y^*, \Theta^*]$. We introduce $u_c = [v_{c,x}, v_{c,y}, v_{c,z}, w_{c,x}, w_{c,y}, w_{c,z}]^T$, which represents the vehicle velocity in the world frame $\{\mathcal{C}\}$. The position of the camera is defined in the robot frame $\{\mathcal{R}\}$ by (t_x, t_y, t_z), ρ defines the pitch angle of the camera, defining the angle between axis z_c and x_r. These values represent the extrinsic camera parameters.

The visual servoing method defined a relationship between the velocity of image features and the camera's velocity:

$$\dot{s} = L_s(X, Y, \Theta)u_c \tag{3.10}$$

In our context, the interaction matrix is defined as follows:

$$L_s(X, Y, \Theta) = \begin{bmatrix} -\frac{T_1}{t_z} & 0 & \frac{XT_1}{t_z} & XY & -1 - X^2 & Y \\ 0 & \frac{T_1}{t_z} & \frac{YT_1}{t_z} & 1 + Y^2 & -XY & -X \\ \frac{\cos\rho\cos^2\Theta}{t_z} & \frac{\cos\rho\cos\Theta\sin\Theta}{t_z} & -\frac{T_2\cos\Theta}{t_z} & -T_2\cos\Theta & -T_2\sin\Theta & -1 \end{bmatrix} \tag{3.11}$$

$$\triangleq \begin{bmatrix} L_X & L_Y & L_\Theta \end{bmatrix}^T \tag{3.12}$$

Where $T_1 \triangleq \sin\rho + Y\cos\rho$, $T_2 \triangleq Y\sin\Theta + X\cos\Theta$. In order to control the robot, the relation has to correlate variations in features with the vehicle's velocity. Figure 3.2b

shows the relation between robot ($\{\mathcal{R}\}$) and camera ($\{\mathcal{C}\}$) frames. By applying homogeneous transformation, we have the relation between u_c and u_r:

$$u_c = {}^{C}T_R \, u_r \tag{3.13}$$

Where ${}^{C}T_R$ defines the transformation between the camera frame ($\{\mathcal{C}\}$) and the robot frame ($\{\mathcal{R}\}$):

$${}^{C}T_R = \begin{bmatrix} 0 & -t_y \\ -\sin\rho & 0 \\ \cos\rho & 0 \\ 0 & 0 \\ 0 & -\cos\rho \\ 0 & -\sin\rho \end{bmatrix} \tag{3.14}$$

$$\triangleq \begin{bmatrix} T_v & T_w \end{bmatrix} \tag{3.15}$$

$$u_c = [T_v, T_w] u_r \tag{3.16}$$

In this way, by utilizing equation 3.13 and equation 3.10, the relation between variation of the image feature and vehicle's velocity is defined by:

$$\dot{s} = L_s(X, Y, \Theta) {}^{C}T_R \, u_r \tag{3.17}$$

Controller Adaptation

The image feature goal varies depending on the situation. Indeed, as shown in Figure 3.4, in the case of lane keeping (Fig. 3.4, a), the goal of control is to maintain the lane, and in this case, the desired image feature is $s^* = [0, Y, \Theta]$. In this context, we are not interested in the variable Y. In the second case (Fig. 3.4,b), the vehicle needs to reach its lane. In this case, the feature image goal is defined as $s^* = [X, 0, \pm\frac{\pi}{4}]$. The plus or minus sign is justified depending on the position of the lane to the left ($\Theta^* = \frac{\pi}{4}$) and to the right ($\theta^* = -\frac{\pi}{4}$). Depending on the situation, a different controller is defined. In the first case mentioned, the variation is made only on the variables X, Θ, and in the second case, it's Y, Θ. The aim of this control is to follow a path instead of track a trajectory. In this context the control is limited to 2 degrees of freedom, for this reason the rank of the matrice L_s is equal to 2 [85].

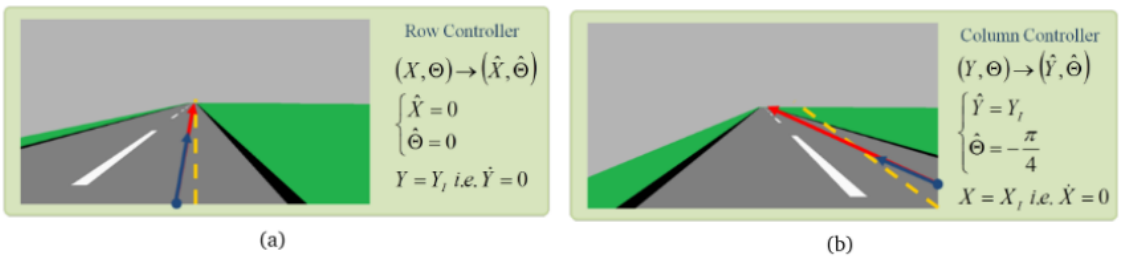


Figure 3.4: Controller adaptation depends on the situation, extracted from [9].

Row controller

The row controller is responsible for driving the features (X, Θ) to $(X^*, \Theta^*) = (0, 0)$.

$$\begin{aligned} \begin{bmatrix} \dot{X} \\ \dot{\Theta} \end{bmatrix} &= \begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_v v + \begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_w w \\ &\triangleq A_{row} v + B_{row} w \end{aligned} \tag{3.18}$$

From the previous relation, we can define the control law as:

$$w = -B_{row}^{+} \left(\begin{bmatrix} K_X e_X \\ K_\Theta e_\Theta \end{bmatrix} + A_{row} v_d \right) \tag{3.19}$$

Where B_{row}^{+} represents the pseudo inverse B_{row}, K_X and K_Θ positives gains and errors are defined as $e_X = X - X^*$ and $e_\Theta = \Theta - \Theta^*$.

Column controller

As with the row controller, the column controller is responsible for driving the features (Y, Θ) to the desired values $(Y^*, \Theta^*) = (Y_I, \pm\frac{\pi}{4})$, where Y_I represents the lower bound of the image.

$$\begin{aligned} \begin{bmatrix} \dot{Y} \\ \dot{\Theta} \end{bmatrix} &= \begin{bmatrix} L_Y \\ L_\Theta \end{bmatrix} T_v v + \begin{bmatrix} L_Y \\ L_\Theta \end{bmatrix} T_w w \\ &\triangleq A_{col} v + B_{col} w \end{aligned} \tag{3.20}$$

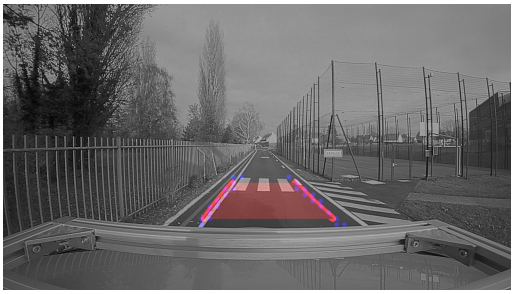
The control law is defined as:

$$w = -B_{col}^{+} \left(\begin{bmatrix} K_Y e_Y \\ K_\Theta e_\Theta \end{bmatrix} + A_{col} v_d \right) \tag{3.21}$$

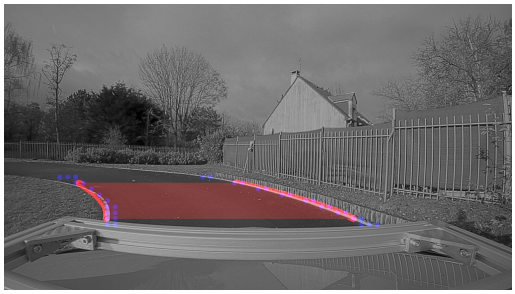
Where B_{col}^{+} represents the pseudo inverse B_{col}, K_Y, K_Θ positives gains and errors are defined as $e_Y = Y - Y^*$ and $e_\Theta = \Theta - \Theta^*$.

3.2.4 Lane Detection

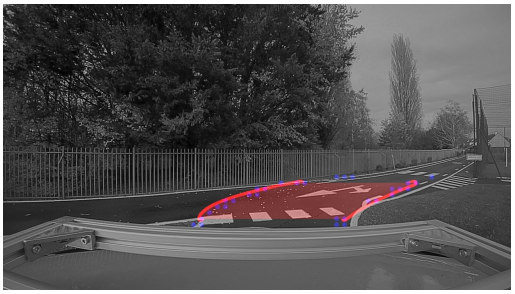
Visual servoing requires detecting the position within an image from that image. In some approaches [10], detection is performed using a so-called geometric model. However, the use of this method has certain limitations. This method relies on marker detection, but this solution may not be reliable due to complex situations where markers are absent or other markings can disrupt recognition. In our proposal, we developed a lane detection method that is based on machine learning (neural networks). This is one of our contribution to the previous work in the Heudiasyc laboratory. Figure 3.5 shows detections made with our model using real data from our test tracks. As these images demonstrate, the use of neural networks in this type of task allows for resistance to complex situations that would disrupt a traditional approach.



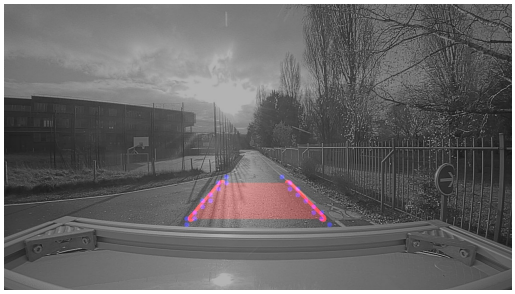
(a) Case simple.



(b) Case curved.



(c) Case complex with deceptive markings.

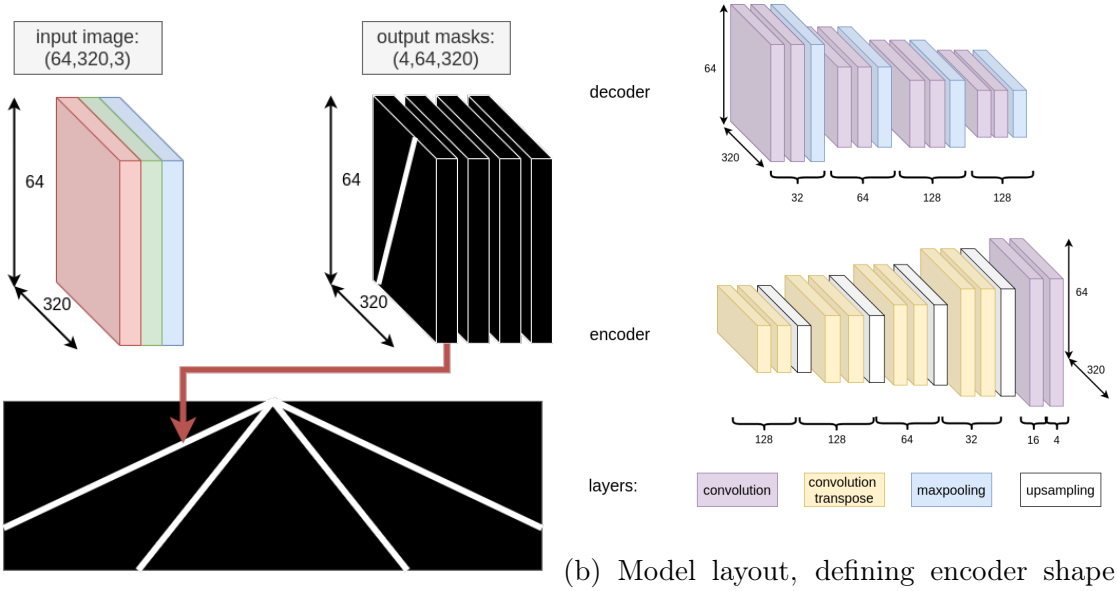


(d) Case without marker.

Figure 3.5: Lane detections perform by our deep-learning model on track test (Sec. 1.3.1).

Lane Detection Model

This model aims to be capable of detecting lanes around the ego-vehicle, including the current lane where the vehicle circulates and the lanes on either sides if any exist. Thus, it is possible to model this problem by detecting the 4 lines separating these different lanes (Fig. 3.6a). Inspired by the literature, we have created a deep-learning model known as an autoencoder, which uses Convolutional Neural network to significantly reduce the data, perform manipulations in very small dimensions, if a latent zone exists, and a decoding is carried out on small amounts of data to expose the data in a dimension of a larger size. In our case, we decided to use a simple autoencoder defined by an encoder and a decoder. We chose this model because it allows the creation of a neural network with a substantial number of parameters, thereby ensuring faster execution. Figure 3.6b shows the layout and shapes of the model used for autoencoding. The model (Fig. 3.6) returns for each image submitted to the model, 4 binary matrices giving for each line, the position in the image of the line. Figure 3.6a shows the data format used by the model and the format of the binary matrices. For each binary matrix, we apply a regression to model each line. To avoid disturbances from outlier detections, we have chosen to perform a regression of a low degree (1 or 2) and to apply RANSAC [86] algorithm that allows interpolation while avoiding the consideration of outlier. The model is trained on the CULane dataset (Sec. 1.3.2). Figure 3.7 shows a detection carried out by a detection pipeline, using an image from the CULane dataset. As shown in



(a) Input and output model.

(b) Model layout, defining encoder shape and decoder shape.

Figure 3.6: Deep-learning lane detection model configuration.

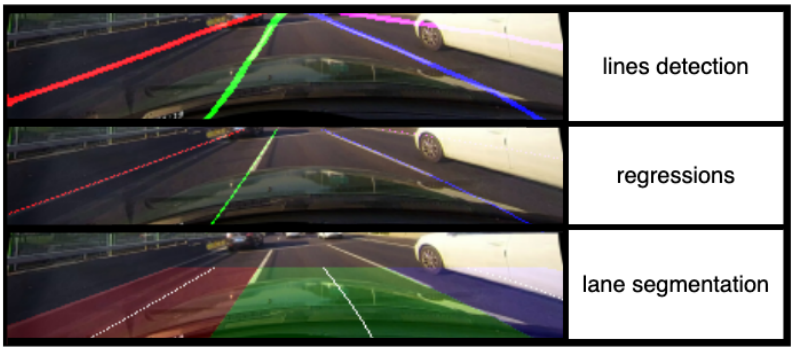


Figure 3.7: Output detection from our model applied on CULane dataset image.

the figure, we have developed a lane segmentation from lane line detection. Thus, based on the detections, it is possible to calculate the path P and then to calculate the image features required for vehicle control to help it stay in its lane.

3.3 Obstacle Avoidance

The visual servoing approach places the vehicle in the center of the lane, but this solution does not take into account obstacles in its environment. Furthermore, the visual servoing solution only enables lateral control. In a previous work developped in the Heudiasyc laboratory, presented in [10], it shown that it is possible to integrate visual servoing into the DWA strategy. The strategy selects the best candidate (v, w) reaching the goal (lane keeping) in avoiding obstacles. In next sections, we recall the initial DWA methodology and the Image-Based Dynamic Window Approach (IDWA) adaptation.

3.3.1 Implementation with DWA

This solution follows the same approach as the classical (DWA) [87]. Firstly, the linear and angular velocities accessible to the robot within a time horizon dt are determined, also referred to as the velocity space, V_t. This velocity space encompasses all achievable velocities by the robot (taking into account maximum acceleration and current speed) that do not result in collisions with obstacles.

Secondly, for each candidate $(v_i, w_i) \in V_t$, an objective value is assigned. The selected command is the candidate with the highest maximum value. This objective function is defined as follows:

$$G(v, w) = \alpha \cdot \text{heading}(v, w) + \beta \cdot \text{dist}(v, w) + \gamma \cdot \text{velocity}(v, w) \tag{3.22}$$

Where α, β, and γ are weights indicating the importance of each function. In the original definition, the *heading* evaluates whether the candidate command aligns the robot towards the target. The *dist* assesses how far the robot will be from obstacles when applying the command. The *velocity* evaluates the linear velocity and determines if it is close to or far from the target velocity, this element helps prevent the robot from getting stuck.

Heading Replacement - The IDWA method

The heading function in the objective function maintains the heading toward the target. The IDWA (Image Based Dynamic Window Approach) proposed by Alves de Lima in [10], that we used as framework, the goal is not to orient the robot towards a target but to ensure that the robot stays in the lane it is navigating. Therefore, this modification takes place at the heading function level to incorporate information related to visual servoing. Let $s = [X, Y, \Theta]$ be the feature vector

extracted from the image. We denote $e_{t+\Delta t}$ (error in the next image after Δt time) as follows:

$$e_{t+\Delta t} = \begin{bmatrix} X_{t+\Delta t} - X^* \\ Y_{t+\Delta t} - Y^* \\ \Theta_{t+\Delta t} - \Theta^* \end{bmatrix} \tag{3.23}$$

Before defining the error, we need to estimate $X_{t+\Delta t}$, $Y_{t+\Delta t}$, and $\Theta_{t+\Delta t}$:

$$\begin{aligned} X_{t+\Delta t} &= \dot{X}\Delta t + X_t \\ Y_{t+\Delta t} &= \dot{Y}\Delta t + Y_t \\ \Theta_{t+\Delta t} &= \dot{\Theta}\Delta t + \Theta_t \end{aligned}$$

From Equation 3.18, we establish a relationship between the vehicle's velocity (v, w) and the velocity of the image features $(\dot{X}, \dot{Y}, \dot{\Theta})$. In the case where we apply the row controller (Eq 3.18), we define the following relationship:

$$\begin{bmatrix} X_{t+\Delta t} \\ \Theta_{t+\Delta t} \end{bmatrix} = (A_{row}v + B_{row}w)\Delta t + \begin{bmatrix} X_t \\ \Theta_t \end{bmatrix} \tag{3.24}$$

Now, we define errors between the image features in the next image and the expected features (X^*, Y^*, Θ^*):

$$\begin{aligned} e_X(t + \Delta t) &= X_{t+\Delta t} - X^* \\ e_Y(t + \Delta t) &= Y_{t+\Delta t} - Y^* \\ e_\Theta(t + \Delta t) &= \Theta_{t+\Delta t} - \Theta^* \end{aligned}$$

$$XY_{error} = \begin{cases} 1 - \frac{|e_X|}{e_{Xmax}} & \text{for row controller} \\ 1 - \frac{|e_Y|}{e_{Ymax}} & \text{for column controller} \end{cases} \tag{3.25}$$

$$\Theta_{error} = 1 - \frac{|e_\Theta|}{\pi} \tag{3.26}$$

We introduce the new heading function:

$$heading(v, w) = \alpha_1 \cdot XY_{error}(v, w) + \alpha_2 \cdot \Theta_{error}(v, w) \tag{3.27}$$

Within the new heading function, we do not directly use the error function because the heading function should increase when the error is decreasing. The divisions by maximal values allow remapping of values between $[0, 1]$. The objective function of DWA becomes:

$$\begin{aligned} G(v, w) =& \alpha_1 \cdot XY_{error}(v, w) + \alpha_2 \cdot \Theta_{error}(v, w) + \\ & \beta \cdot \text{dist}(v, w) + \gamma \cdot \text{velocity}(v, w) \end{aligned} \tag{3.28}$$

3.3.2 Gradient Descent Resolution

In the traditional approach, solving the Dynamic Window Approach (DWA) is accomplished by searching in the velocity space for the solution (v, w) with the highest value of the objective function. In other words, solving this problem involves evaluating the value of each candidate. This resolution is then of high complexity, $O(n \times m)$, with n and m representing the sizes of the search space of admissible velocities. Furthermore, the most optimal solution will only be approximated due to the discretization of the search space. Our idea is to apply a method capable of achieving lower complexity and higher precision. Specifically, we propose converting the initial objective function into a loss function. This means that the optimal solution minimizes the function instead of maximizing it. Additionally, if this loss function is convex, we can apply gradient descent to converge to the global minimum, thus avoiding brute force. Let γ be the learning rate and L be a function defined and differentiable in a neighborhood of the point w. The gradient descent is defined as follows:

$$w_{n+1} = w_n - \gamma \cdot \nabla L(w_n) \tag{3.29}$$

The approach involves changing the objective function to a convex one. To create the loss function, we can utilize the following property of convex functions:

> If $w_1, \ldots, w_n \geq 0$, and $f_1, \ldots, f_n$ are all convex, then $w_1 f_1 + \ldots + w_n f_n$ is convex.

The idea is to leverage this property and find an equivalent convex loss function for each initial sub-function of the DWA. The loss function is defined as follows:

$$\begin{aligned} L(v, \omega) &= \alpha \cdot \text{heading}_{\text{loss}}(v, \omega) + \delta \cdot \text{dist}_{\text{loss}}(v, \omega) \\ &\quad + \gamma \cdot \text{velocity}_{\text{loss}}(v, \omega) \end{aligned} \tag{3.30}$$

Contrary to the initial definition, the optimal solution (v^*, w^*) is the velocity belonging to the search space V_t that minimizes the loss function:

$$(v^*, w^*) = \underset{(v,w) \in V_t}{\arg\min}(L(v, w)) \tag{3.31}$$

Thus, by defining a convex function for each initial function, it becomes possible to find a solution that minimizes this function.

3.3.3 Constraining the Gradient Descent

The initial DWA approach constrains the search space to velocities reachable within a time interval dt while avoiding obstacles. Therefore, applying gradient descent until the optimal solution is found is not feasible because we cannot guarantee that the optimal solution lies within this limited search space. In our approach, we incorporate the concept of a search space similar to the DWA approach, but we perform gradient descent until the solution belongs to the defined search space, V_t. This search space is determined based on the current robot's velocity, ensuring that

the current speed is encompassed within this space ($(v_t, w_t) \in V_{t+1}$). To incorporate this notion of the search space, we continue applying gradient descent until we exit this predefined space. Figure 3.8 illustrates this limitation. The hashed rectangle represents the search space at time t, the green point (first point) represents the current speed where the gradient descent starts, and the yellow points are found by the gradient descent. As shown, the 5^{th} point is outside the search space, so we stop the gradient descent on the variable w and place the point on the boundary. The gradient descent continues only on the remaining axis, which is not constrained. Once the descent is limited on an axis, means the variable is out-side of the velocities space (V_t), that variable is considered non-trainable, and the descent evolves only on the last remaining axis.

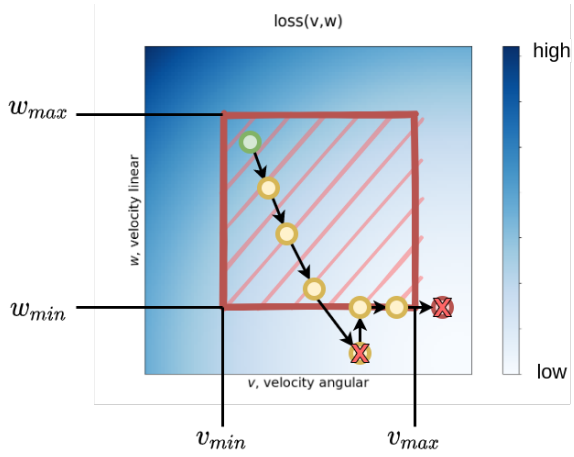


Figure 3.8: Explanation of gradient descent constraint.

First, we applied the gradient descent method in the context where the DWA (Dynamic Window Approach) was defined. In this context, the robot must reach positions while avoiding obstacles, detected through its perception. Figure 3.9 shows the results of our approach. In the remainder of this section, we focus on the problem of autonomous vehicle driving. Our published article [19] provides detailed insights into our solution in the original context. Furthermore, this solution has been employed within the scope of an academic-industrial project. Moreover, we have employed this approach in these following works [22, 88], the solution was applied within the context of collaborative navigation between a drone and a ground robot.

3.3.4 Loss Functions Definition

In order to apply our gradient descent solution, we define three convex functions that enable the robotic vehicle to navigate within its lane while avoiding obstacles.

Heading Function

By using the command generated by the visual servoing controller mentioned in (3.19), we exploit this command as the reference value. At a time instant t,

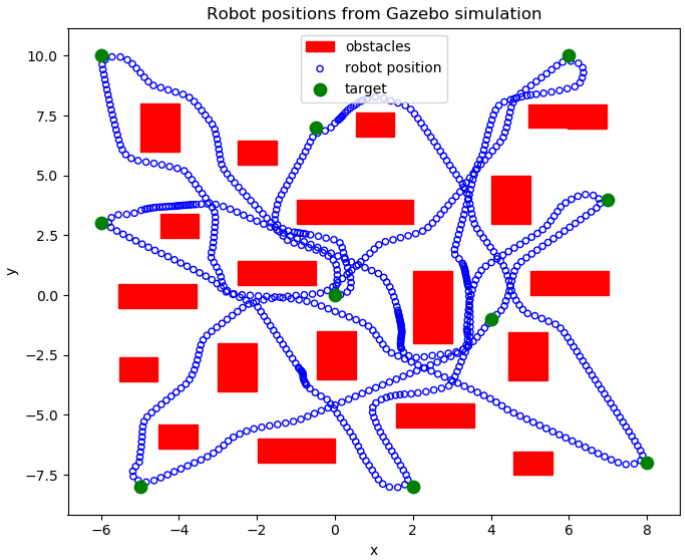


Figure 3.9: DWA combined with gradient approach, positions results. Applied in simulation with Gazebo.

we consider this command to be the most optimal command for achieving the objective and recentering in the lane. Therefore, we define this value as $w^*_{heading}$. This command depends solely on the angular velocity w. Thus, we define the loss function $heading_{loss}$ as follows:

$$heading_{loss}(v, w) = \frac{1}{2}(w - w^*_{heading})^2 \tag{3.32}$$

From this function, we calculate the partial derivatives with respect to the axes v and w to define the gradient of this function:

$$\frac{\partial heading_{loss}}{\partial v} = 0 \tag{3.33}$$

$$\frac{\partial heading_{loss}}{\partial w} = (w - w^*_{heading}) \tag{3.34}$$

Dist Function

The dist function estimates the most optimal candidate from the velocity space for the robot to avoid an obstacle while maintaining its current heading. This function is based on a concept of a safe zone, which represents an area where the robot can move safely. This approach draws inspiration from the Vector Field Histogram methods [89]. From the data obtained from the LiDAR, it is possible to detect the areas in which the vehicle can safely maneuver. Figure 3.10 illustrates the construction of these zones. Initially, based on a critical distance r_{limit}, we define out zones, which indicate areas where it is not possible to go due to the

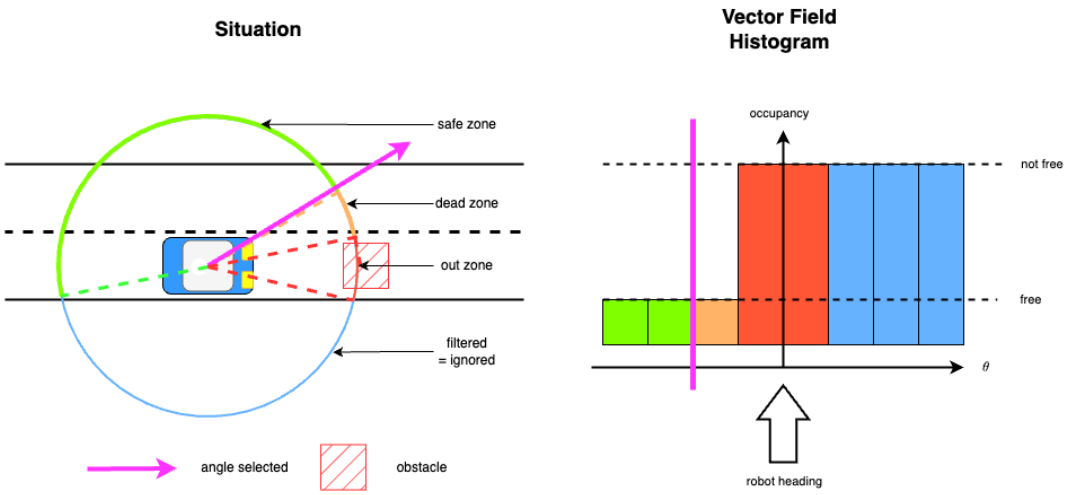


Figure 3.10: Explanatory illustrations of zone construction and optimal angle definition. LiDAR filter is explain in part 3.4.

presence of an obstacle. A dead zone is added to the out zone to account for a safety distance $d_{security}$, considering the vehicle's dimensions to prevent collisions. Once these zones are defined, it becomes possible to establish a safe zone in which the robot can operate. From these zones, we calculate the angle θ_{dist} closest to the vehicle's heading, which belongs to a safe zone. Using this value θ_{dist} and taking into account the time interval dt we define the function $dist_{loss}$ as follows:

$$\begin{aligned} dist_{loss}(v, w) &= \frac{1}{2}(w - w^*)^2 \\ &= \frac{1}{2}\left(w - \frac{\theta^*_{dist}}{dt}\right)^2 \end{aligned} \tag{3.35}$$

From the $dist_{loss}$ function, we define the gradient of this function as:

$$\frac{\partial dist_{loss}}{\partial v} = 0 \tag{3.36}$$

$$\frac{\partial dist_{loss}}{\partial w} = \left(w - \frac{\theta^*_{dist}}{dt}\right) \tag{3.37}$$

Velocity Function

The function $velocity_{loss}$ directly utilizes the desired linear velocity as the reference value v^*. Thus, we define the loss function as follows:

$$velocity_{loss}(v, w) = \frac{1}{2}(v - v^*)^2 \tag{3.38}$$

From this function, we compute the partial derivatives as follows:

$$\frac{\partial velocity_{loss}}{\partial v} = (v - v^*) \tag{3.39}$$

$$\frac{\partial velocity_{loss}}{\partial w} = 0 \tag{3.40}$$

3.4 LiDAR Road Filter

The solution without performing filtering on the lidar data, the vehicle's behavior may be dangerous. Indeed, without limiting the vehicle's possibilities, it is possible that when faced with an obstacle, the strategy decides to overtake the obstacle by passing it outside the road. For example, in Figure 3.10, when applying the strategy without a filter, the vehicle can avoid the obstacle either to the left or the right. Considering the road's topology, we aim to guide the vehicle to pass the obstacle on the left. In the literature [9], this problem can be resolved by adding virtual obstacles to restrict the range of possibilities. The idea we propose is to perform lidar data filtering based on detections made by the lane detection system. This way, the strategy will be guided towards a region covered by a lane during overtaking, ensuring that there are no critical cases where the vehicle overtakes the obstacle in a non-lane area. As discussed in the previous sections, our lane detection model can detect the lines on both sides of the current lane, if they exist. Figure 3.11 illustrates the concept of filtering. As shown in the figure, if the lane is detected, then the LiDAR data covering that area is retained; otherwise, it is filtered.

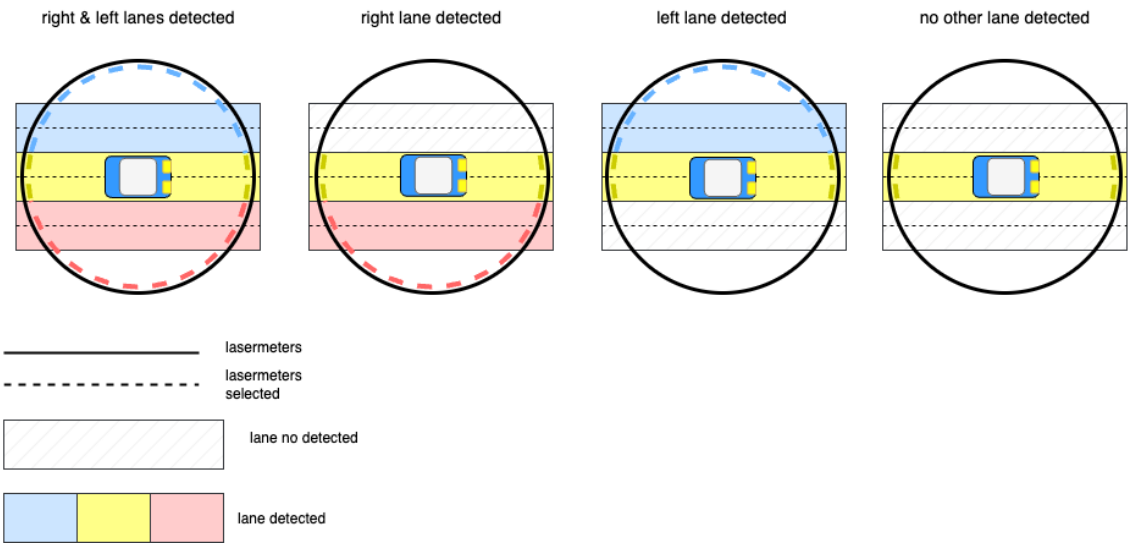


Figure 3.11: Explanation of LiDAR filtering with use cases.

The filtered areas will subsequently be interpreted as non-free zones, meaning that the strategy will not steer the vehicle into those areas. Figure 3.12 illustrates the final block diagram connecting the planning part described in this section and the controller parts defined in the previous sections.

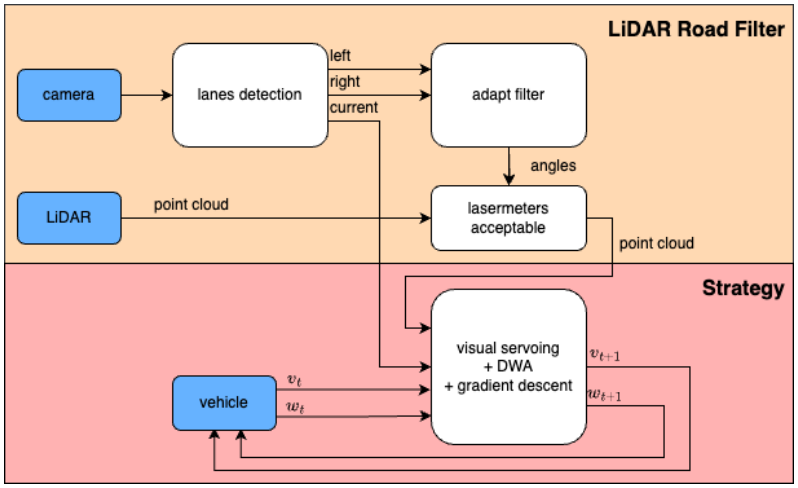


Figure 3.12: Simplified block schema of LiDAR Road Filter and Strategy parts.

(a) Straight test. (b) Curve test. (c) Obstacle test.

Figure 3.13: Simulation images from SCANeR studio for each test.

3.5 Applications

This controller was developed and tested using the SCANeR studio simulator. We chose to conduct tests in a simulation environment because this method involves obstacle avoidance with the primary goal of preventing accidents. Testing in simulation allowed us to ensure its proper functioning and offered the flexibility to define test scenarios as needed. In these tests, we defined three scenarios to evaluate its performance. In the first case, the car drives on a straight road without obstacles (Fig. 3.13a). In the second test, the car moves along a curved road without obstacles (Fig. 3.13b). These initial tests aimed to verify the functioning of the visual versioning. In the final test, the vehicle must avoid an obstacle in its path (Fig. 3.13c). This test evaluates the strategy's performance in obstacle avoidance. Figure 3.15a displays the car's speed during the tests, showing that the vehicle is capable of maintaining a speed close to the 10 km/h limit. It is also indicated in figure 3.14 that the vehicle stays within its lane. Notably, in the test on a curved road, the deviation from the center of the lane is more significant than in the straight-line test. This deviation can be attributed to the use of first-degree polygon interpolation for the road lines to avoid sensitivity to outliers. Consequently, interpolation of

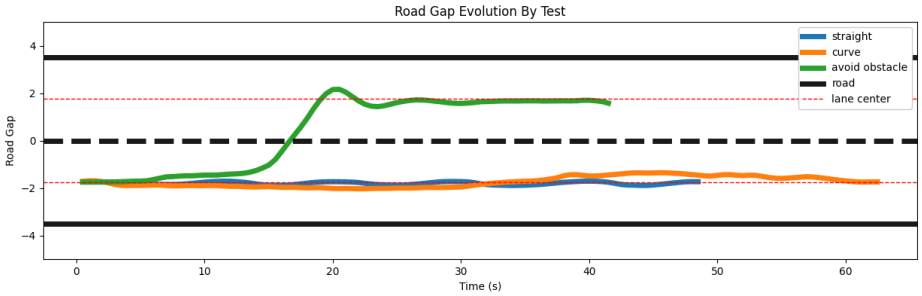


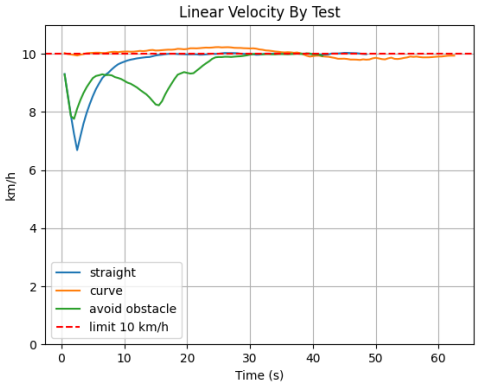
Figure 3.14: Trajectory by test.

these models in cases where the road curves is less precise. However, the vehicle still maintains its lane despite a slightly larger error. Figure 3.15c displays the number of iterations performed by the gradient descent during the tests. In the first two tests, the descent required only a few iterations (≈ 10). In the third test (Fig. 3.14), the vehicle stays in its lane until it reaches the obstacle. Then, it initiates a lane change to the left, as the vehicle is aware of the available left lane. Notably, as the vehicle approaches the obstacle, a deceleration occurs to adapt the angular velocity and facilitate the lane change. Once the overtaking is complete, the vehicle continues on the left lane. Unlike the previous tests, the number of iterations performed by the gradient descent (Fig. 3.15c) becomes more significant when avoiding the obstacle, reflecting the influence of the $dist_{loss}$ function. Furthermore, Figure 3.15b indicates the angle that the vehicle must follow to avoid obstacles. In tests without obstacles, the angle is zero. In the case of the obstacle test, the angle increases as the vehicle approaches the obstacle and abruptly returns to zero once the lane change is completed.

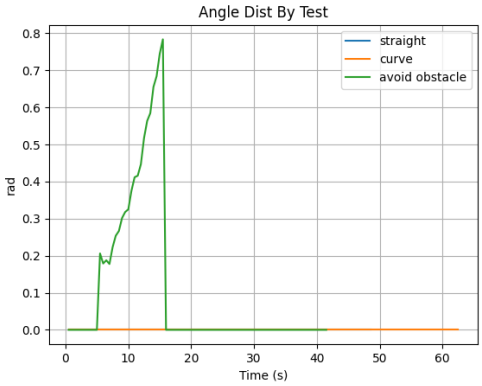
In addition to these applications, this solution has been used in other studies [14], [17], where we perform autonomous driving with context awareness to adapt driving based on both external and internal context.

3.6 Defining Intention From Command

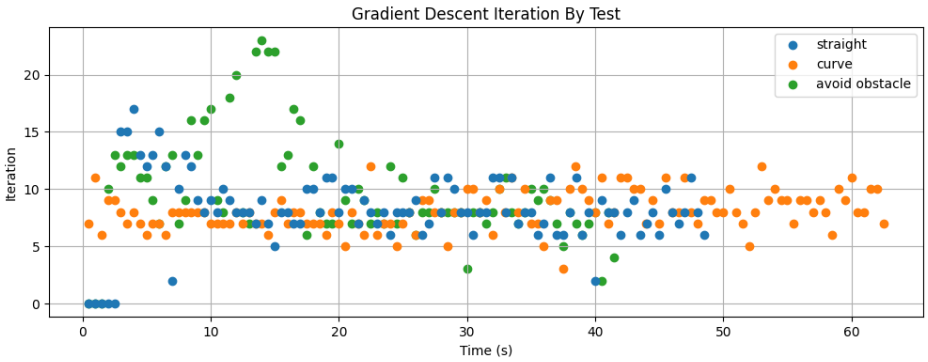
In the shared control approach proposed in this thesis, we estimate the intentions of each entity, autonomous system, and human. We define the intention of the autonomous system based on the control and strategy defined in this chapter. This strategy defines the command based on information from the image feature defined by lane detection, the surrounding obstacles, and the current speed of the vehicle. This diverse set of data can be predicted over a time horizon, with a time interval between two data points of δ_t. Thus, dynamically, it is possible to predict the future inputs of the model, which will generate the future outputs and so on. Figure 3.17 shows a diagram of this solution. In this approach, we need to express the next input for each control input based on the current output.



(a) Speed by test.



(b) Angle safe by test.



(c) Number of iterations by test.

Figure 3.15: Data resulting from the simulations conducted for each test.

3.6.1 Image Feature Prediction

Let X and Θ represent the features of the image used for visual servoing. Inspired by the equation 3.24, and taking into account the relationship in Eq 3.18, the derivative of these image features depends on the vehicle's velocity. Thus, by utilizing the velocity derived from our controller (v_c, w_c). Furthermore, we decide to use a point D that is higher on the image illustrating the future behavior of the road.

$$\begin{bmatrix} X_{t+1} \\ \Theta_{t+1} \end{bmatrix} = \begin{bmatrix} X_t \\ \Theta_t \end{bmatrix} + \left(\begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_v v + \begin{bmatrix} L_X \\ L_\Theta \end{bmatrix} T_w w \right) \cdot \Delta_t \tag{3.41}$$

Where Δ_t represents the time interval of controller execution.

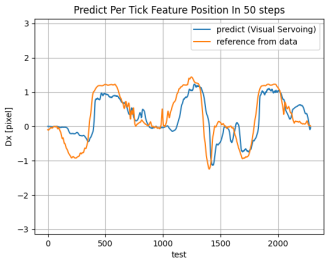
Figure 3.16 illustrates the feature prediction tests. Figures 3.16a and 3.16b represent the prediction of the feature of $D.x$ and θ performed over more than 2000 tests. For each test, the prediction is compared to the value of the feature found at 50 iterations (equivalent to 5s). Figure 3.16c shows the prediction error normalized by the maximum domain of definition (1280 pixels and 2π angle). The error in the prediction of $D.x$ is low, it is higher for the prediction of the angle θ. However, during our control tests, we give more importance to the position of the feature position than the angle. Indeed, the control gain is more significant on the translation error than the angle. Thus, even if the prediction of the angle is of lower quality, it does not impact the result of the command prediction. Figures 3.16d, 3.16e, and 3.16f represent the data of a test. In this test, the car is placed in a roundabout, the lane detection is illustrated by figure 3.16d. Figure 3.16e is the prediction of $D.x$ in the context of this test. From the predictions of the feature image, the estimates w are then calculated 3.16f. As the image shows, the control of the angular velocity is very close to the calculated command and the command executed by the car during the tests.

3.6.2 LiDAR Prediction

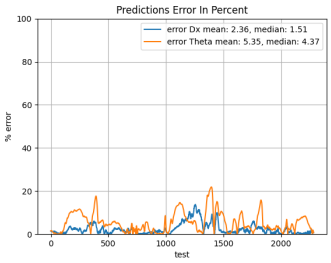
In addition to the image feature data, we need to provide future lidar data to the controller. From the lidar data obtained at a given moment, we predict lidar data for a time horizon, Δ_t. For this, we assume that the environment is static. Thus, based on a velocity command, we project it and define the new relative position after applying this velocity for a time interval, Δ_t. Then we can translate the lidar data defined by polar coordinates to redefine the center, by the new position estimated by the velocity projection. Let c_t be the current center of the lidar data, meaning $c_t = (0, 0)$, in the lidar's coordinate system. Let c_{t+1} be the new center, from the velocity projection. We denote x and y as the translation from point c_t to point c_{t+1}, and θ the orientation of the robot in the new position. Thus, we apply the following calculation to all lidar points (l_i):

$$\begin{bmatrix} x_{i,t+1} \\ t_{i,t+1} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_{i,t} \\ t_{i,t} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.42}$$

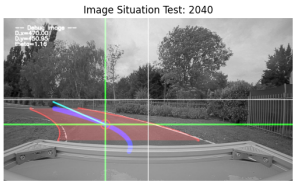
(a) Predict DX



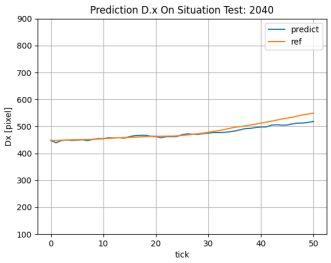
(b) Predict Theta



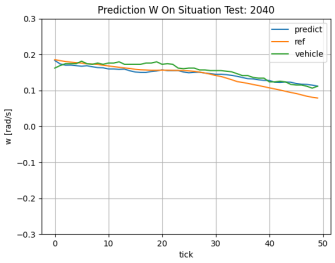
(c) Errors prediction in percentage taking account maximal value possible.



(d) Situation Image



(e) Situation Prediction DX



(f) Situation Prediction W

Figure 3.16: Prediction tests conducted on real data at the Seville track.

Where $(x_{i,t+1}, y_{i,t+1})$ denotes the Cartesian position of the l_i point. Thus, the lidar data will be centered at $(0, 0)$ in the new position.

3.7 Conclusion

Utilizing lane detection combined with lidar, the autonomous system drives the car within its lane while avoiding obstacles. In this new approach, we have proposed a more optimized way to find the best candidate. Additionally, we have used a deep learning network to perform lane detection, making our controller more robust. We have redefined the controller's behavior by introducing a planning layer that limits the possible actions of the autonomous system to ensure the car's safety. This approach has been tested in experimental works of [22, 88] using an indoor mobile robot, and in simulation using a virtual vehicle in a professional car driving simulator SCANeR Studio, demonstrating the controller's adaptability in various scenarios. It's important to note that this method relies on a set of hyperparameters to be configured, including the weights of the loss function, convergence step size, and convergence criteria. These different hyperparameters can make it challenging to fine-tune the controller for different situations. This issue inherits from the original version of the DWA, where parameter tuning depends on the environment. Therefore, we can consider using concepts from the literature to make these parameters dynamic based on the situation [90]. With the definition of this controller, it becomes possible to anticipate its intentions by predicting future states.

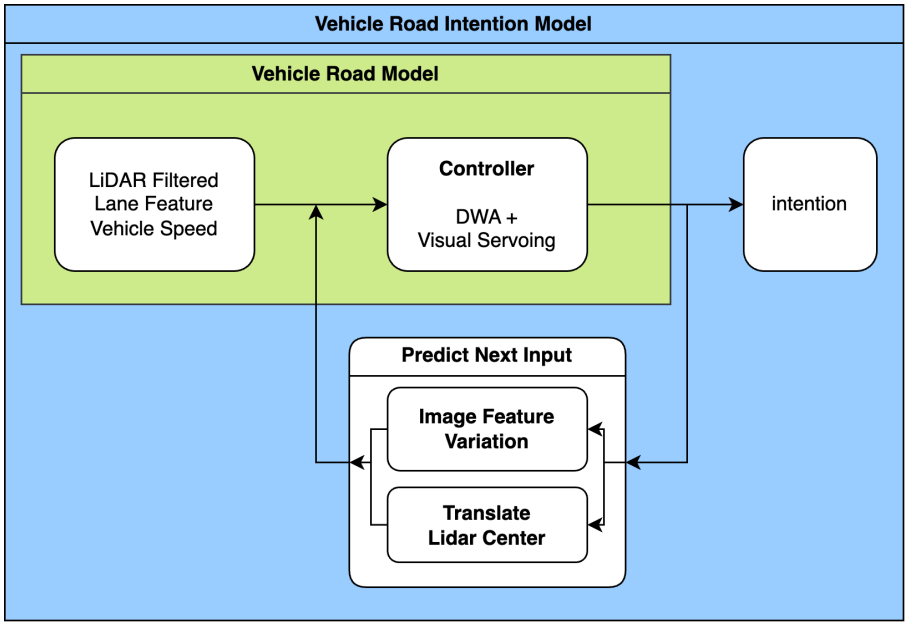


Figure 3.17: Simplified block diagram model, including intention generation.

4 Human Driving Prediction

Abstract: *Predicting the intentions of the human and the machine in the near future is required for the human-machine shared control of automated intelligent vehicles. The autonomous system can inform about its future intentions, but it is not possible for the human to provide this information; thus, prediction becomes necessary. This chapter proposes a deep learning methodology to predict human navigation intentions within a time horizon of a few seconds. It utilizes a Recurent Neural Network (RNN) architecture based on Long Short-Term Memory (LSTM). Taking various preprocessed and non-preprocessed data as input, generated by embedded sensors and the intrinsic data of the vehicle, the proposed model predicts the future linear and angular velocities of the vehicle. The model was trained and tested on a dataset created from real data collected from our cars equipped with sensors (LiDAR, camera), in various scenarios and road types. Furthermore, a data sensitivity study is presented to evaluate the effects of missing data in the learning process.*

Contents

4.1 Motivation

In this approach, shared navigation involves the fusion of human and autonomous intentions. The driving intentions are represented as a sequence of velocities:

$$I = \{(v_t, w_t), ..., (v_{t+k}, w_{t+k})\} \tag{4.1}$$

This intention is fundamental for command fusion. Thus, the fusion process cannot rely solely on the current command from both the human and autonomous systems because: It lacks detailed information, failing to provide a clear understanding of the intentions of each entity in the near future. The information becomes outdated by the time it reaches the fusion system. While the autonomous system can share its driving intentions, it is not admissible to continuously request real-time intentions from the human for the following seconds. In fact, the human is driving the vehicle, he can't in the same time inform about these intentions. Therefore, the solution is to develop a predictive model capable of anticipating human driving behavior in the coming seconds, effectively using this prediction as human intention. The problem is then formulated as:

$$H_\Theta(X) = \{\hat{y}_{t+1}, ..., \hat{y}_{t+k}\} \tag{4.2}$$

Here, H_Θ represents the prediction model, X denotes the input data (previous vehicle and environment states), and $\{\hat{y}_{t+1}, ..., \hat{y}_{t+k}\}$ constitutes the sequence to be predicted. Figure 4.1 illustres the prediction goal, showing the model exploiting states data to predict the velocities sequence. The figure also demonstrates that the model's inference is performed using data from a dataset.

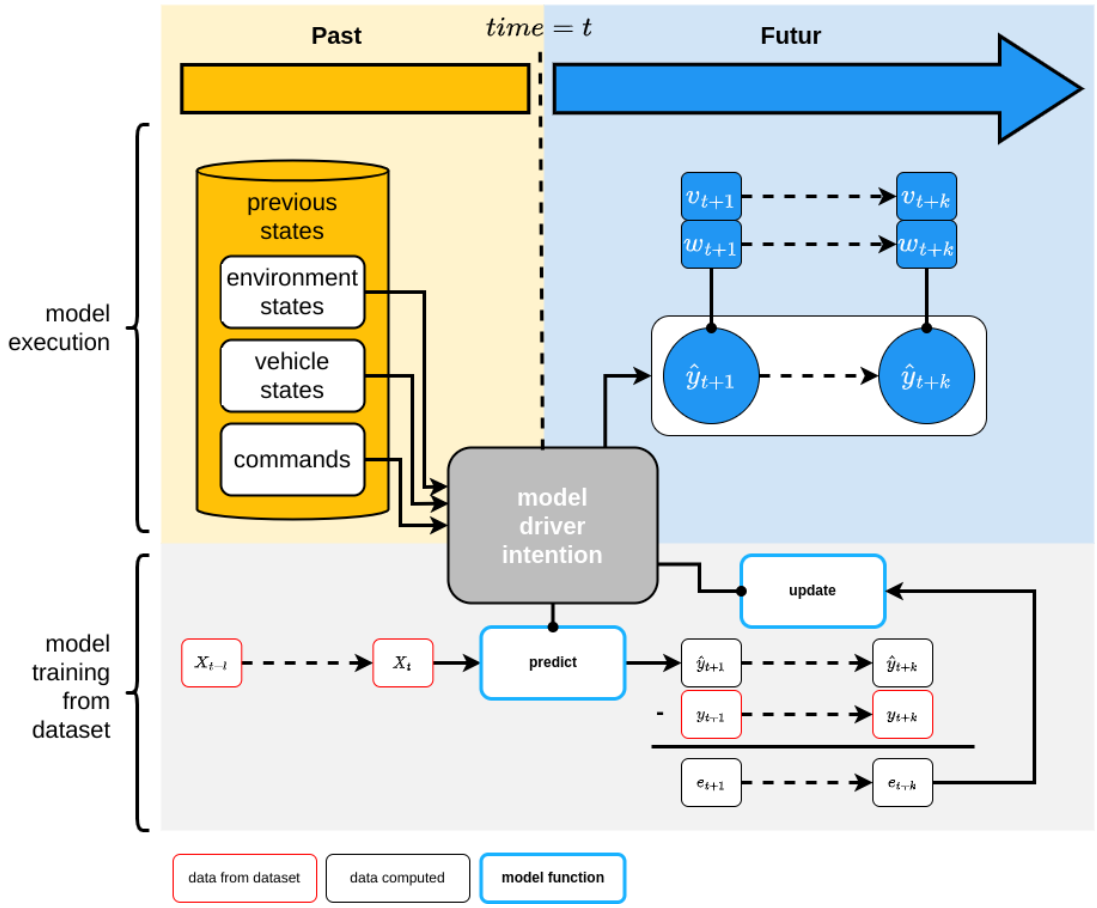
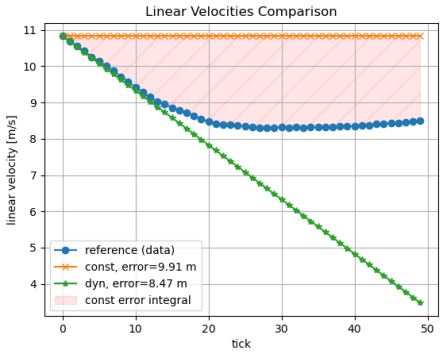


Figure 4.1: Operation of the model during execution and explanation of the model training phase.

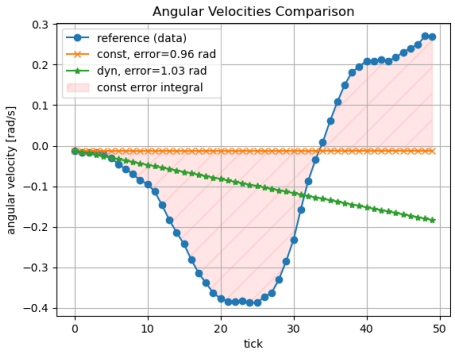
Speed intentions can vary abruptly over a short time horizon, especially for angular velocity, in the case of overtaking or driving in a roundabout. These significant variations, stemming from external factors such as road architecture or the behavior of other road users, compel us to consider these elements in our prediction. Figures 4.2a and 4.2b show the profiles of linear and angular speed during driving in a roundabout. For each profile, two profiles are calculated: a constant profile, assuming constant entry speed, and a dynamic profile that takes into account acceleration and entry speed. The cumulative error (including the time interval) of these approaches is calculated as follows:

$$error(y_{predict}, y_{true}) = \sum_{i=0}^{n} |y_{predict,i} - y_{true,i}| \cdot \delta t \tag{4.3}$$

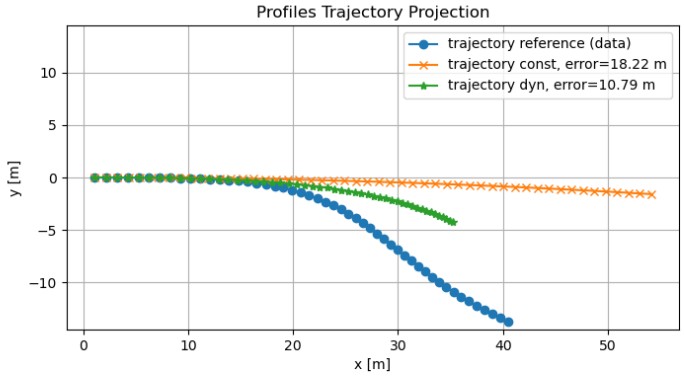
Where δt defines the interval time between two consequently values. For these data, the interval time δt equals to 0.10 seconds. Using this formula, the error induced by the approximation (constant and dynamic) results in a significant error (approximately $9m$ for linear speed and $1rad$ for angular speed) in the case of driving in



(a) Linear velocity.



(b) Angular velocity.



(c) Trajectory.

Figure 4.2: Approximations methods comparison. These graphs show the importance to create model.

a roundabout. Figure 4.2c shows a trajectory projection of these different approximations, with the trajectory of the constant method deviating by more than $18m$ from the reference position and the dynamic method deviating by more than $10m$. As we can see, predicting the actual velocity profile is important. The prediction method described below in this chapter is based in builting a learning model so that the profile errors, as those depicted depicted in figure 4.2, are minimized.

Studies on driving prediction primarily focus on forecasting the behavior of vehicles around the ego-vehicle [91, 92]. A vehicle's operation depends on various factors, including road infrastructure, traffic regulations, and the behavior of other vehicles. This underscores the importance of predicting behaviors to account for the intentions of surrounding vehicles in the commands generated by the autonomous system. In this context, predictions are based on extrinsic vehicle data. In this approach, we define a deep learning model capable of predicting the intentions of the human driver over a short time horizon based on a sequence of data. We also define the dataset used by the model for its training.

4.1.1 Problem Definition

Prediction of human driver behavior is used to forecast future vehicle states or maneuvers. In our context, the model is tasked with predicting the future velocities (linear and angular) of the vehicle over a short time horizon. Our problem can be defined as follows:

$$\begin{aligned} H_\Theta(X_t, X_{t-1}, X_{t-2}, ..., X_{t-n}) &= (y_{t+1}, y_{t+2}, ..., y_{t+m}) \\ &= \begin{bmatrix} (v_{t+1}, w_{t+1}) \\ (v_{t+2}, w_{t+2}) \\ ..., \\ (v_{t+m}, w_{t+m}) \end{bmatrix}^T \end{aligned} \tag{4.4}$$

Here, H_Θ represents the predictive model with parameters Θ, v_{t+1} denotes the linear velocity, w_{t+1} represents the angular velocity at time t+1, and $(X_t, X_{t-1}, X_{t-2}, ..., X_{t-n})$ represents the sequential input used by the model. The prediction horizon is set at 5 seconds. This value is based on previous studies [93], which indicated a significant prediction error beyond 5 seconds. In the context of local driving behavior, a 5-second prediction seems sufficient.

4.1.2 Problem Resolution

The approach proposed consists of defining a model capable of inferring human behavior and making predictions based on it. This approach relies on machine learning techniques, specifically deep learning. This technique requires to define the data on which the model will be able to infer and the structure of the deep learning model.

4.2 Dataset

The choice of the dataset is crucial for developing our model. Several key considerations must be addressed. Firstly, the model needs to be capable of predicting human behavior based on the data. Therefore, it is essential to determine whether the dataset contains information that correlates with human behavior. Secondly, the model's must fit the limitations of our vehicles, which are restricted by the sensors integrated into the vehicle and the available computing capacity. Additionally, the datasets currently available (Sec. 2.4.1) necessitate the use of external perspectives beyond the vehicle itself. The data is collected through sensors placed in the surrounding infrastructure, such as cameras installed on bridges, for example. However, in our specific case, we aim to utilize data that is accessible and visible from within the vehicle. To address these considerations, we have decided to create our own dataset based on our vehicles (Sec 1.3.1).

4.2.1 Roadmap

The dataset records all the information used for model training. This information should be diverse to gain maximum insights into the current state of the vehicle and its environment. This data should be kept in its rawest form, without pre-processing, to preserve sensor-derived information to the fullest extent. Additionally, the dataset should cover a wide range of situations, including different road conditions, speed limits, traffic densities, and varied driving behaviors. This diversity is essential to prevent overfitting during model training. Figure 4.3 presents the various data types recorded by our vehicle. Thus, this data is categorized into three groups: environment, vehicle dynamic state, and vehicle control state (refer to Table 4.1). Table 4.2 presents the characteristics of sensors embedded in the vehicle.

A roadmap was established in advance to ensure diversity in the dataset. Table 4.3 outlines the different characteristics mandated by the roadmap.

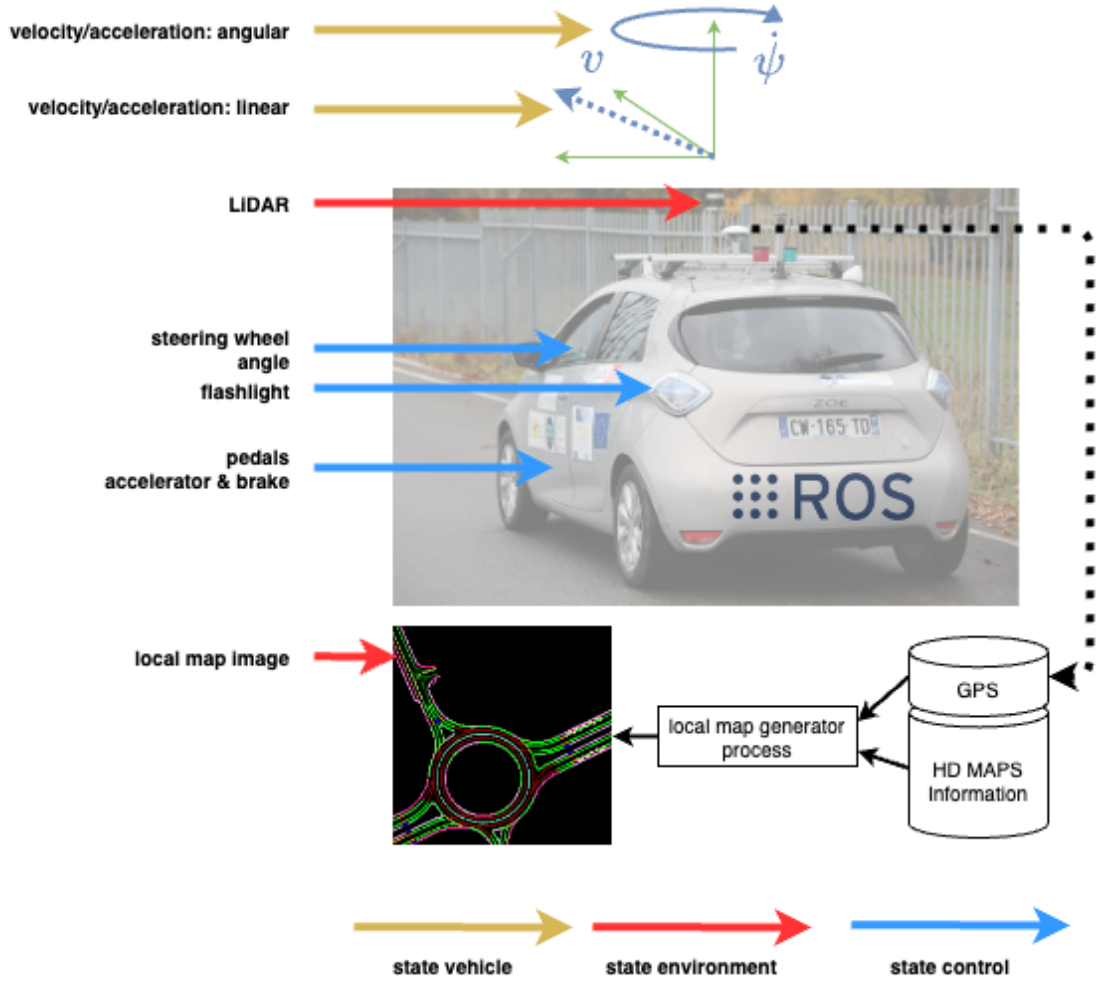


Figure 4.3: Types of data recorded by our vehicle for dataset creation.

Environment	Vehicle dynamic state	Vehicle controls state
- LiDAR [points cloud]; - Front camera [image];	- Velocity linear $[m/s]$; - Velocity angular $[rad/s]$; - Acceleration linear $[m.s^{-2}]$; - Acceleration angular $[rad.s^{-2}]$; - GPS positions [lat and long];	- Steering wheel angle [rad]; - Flashlight states [enum]; - Pedal positions [percents]

Table 4.1: Categorization of dataset.

Sensor	Frequency	Additional information
LiDAR	10 Hz	40 layers 30 000 points per acquisition
camera	10 Hz	1280×720 resolution
GPS	50 Hz	cm precision
BUS-CAN reader	100 Hz	

Table 4.2: Sensor characteristics.

4.2.2 Data Pre-Processing

Figure 4.4 shows different steps of the dataset process record. After recording additional information is generated and injected to the dataset.

Local Map Generation: To give information to our model about the geometry of the road and its orientation, we generate a local map representing this information. To do this, we use the GPS data from the vehicle and information from an HD map. By combining this information, we are able to create an image with the desired road information. Figure 4.5 shows the local map image generated by our algorithm in a situation.

With this approach, we avoid the need to vectorize a set of information. For example, to provide information about a roundabout, we would otherwise have to define the current position of the next roundabout, the relative angle with the ego-vehicle, the dimensions of the roundabout, and the number of lanes it has. This applies to a variety of infrastructure elements, intersections, and road curves for example. In this way, we provide the model with the most raw and diverse information possible.

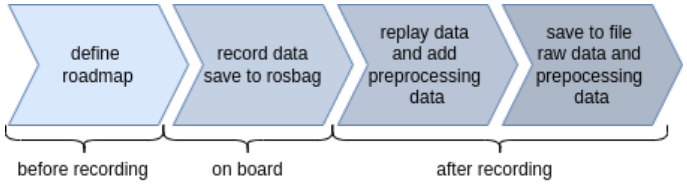
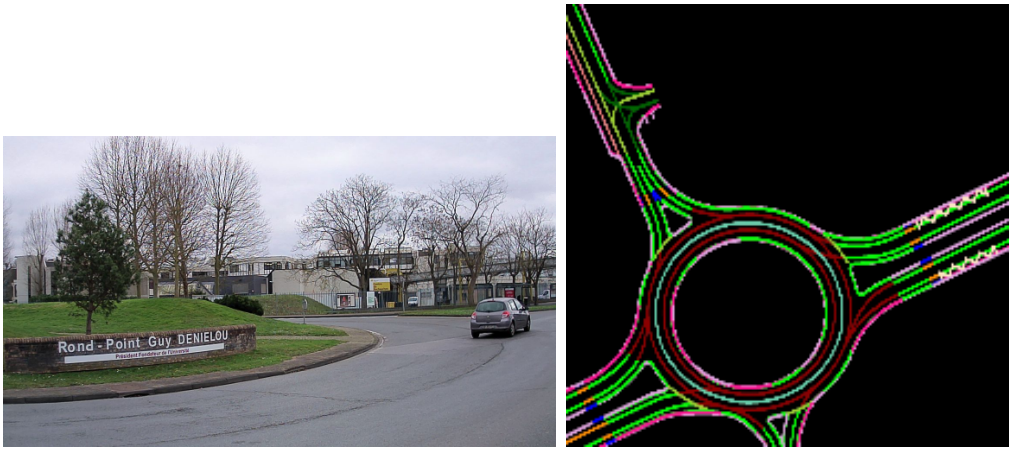


Figure 4.4: Dataset process steps.

Feature	Variants
Lanes number	1, 2
Lanes shape	curved, straight
Speed limit	30, 50, 70, 90 km/h
Roundabouts	with, without
Intersections	with, without

Table 4.3: Dataset characteristics, variation per feature.



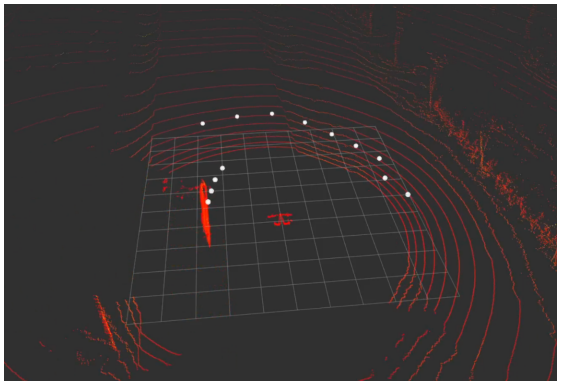
(a) Image from camera front. (b) Map image generated.

Figure 4.5: Map generation (b) from the situation (a).

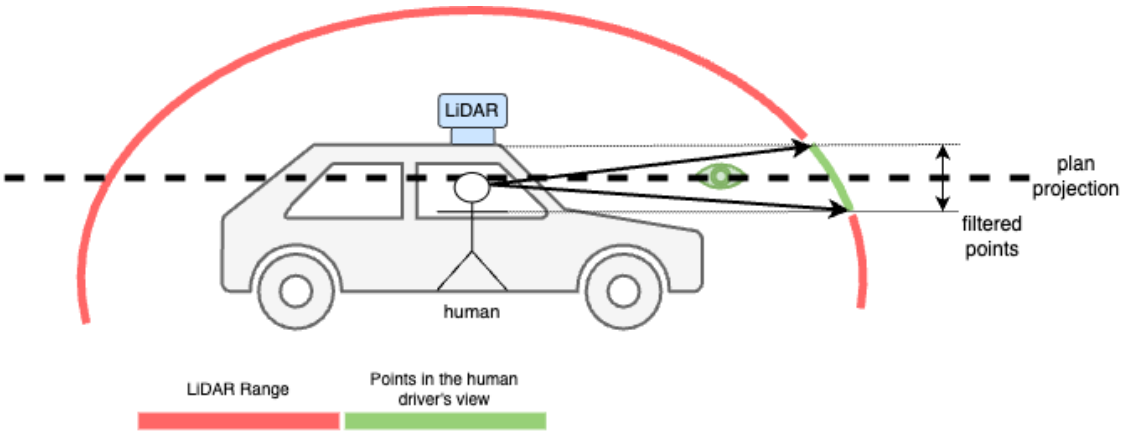
LiDAR Process: The LiDAR gives information about obstacles and building shape around the vehicle. The pointcloud can't be exploited directly by a model because the content is too big. The raw pointcloud generated contains 30 000 points. The article [94] focuses on LiDAR point clouds and offers various techniques for representing LiDAR data. Point clouds obtained from LiDAR are data-intensive, consuming significant memory and computational resources. To address this, we made the decision to reduce the number of points and transform the data from 3D to 2D. In the reduction process, we made certain assumptions. For instance, we considered that points located directly above the vehicle's position don't significantly affect driving decisions. Moreover, the model must be able to interpret the choices of the human; it is important, therefore, to expose the model to the information that is perceptible by the human (Fig. 4.6c). In this way, we limit the points in the driver's field of vision. Additionally, since we weren't performing detailed segmentation, we didn't require a high level of point cloud detail. Our reduction process involved filtering the data vertically and dividing the area around the vehicle into regular angles. This ensured that the closest unfiltered point in each region was used to represent that region.

Rolling Data: The acquisition of temporal sensor data, such as speed, acceleration, and steering wheel angle, introduces noise into the data. To mitigate the

(a) Moving average applied on angular velocities.



(b) LiDAR reduction, red points represent pointcloud and white points represent the reduction.



(c) LiDAR reduction motivation.

Figure 4.6: Data reduction, during preprocessing step.

sensitivity to acquisition noise, we applied a centered moving average, as described in [95].

4.2.3 Dataset Information

The recorded data are saved at 10 Hz. The training dataset (Appendix A) is composed of about 36000 acquisitions regrouping what represents 3600 seconds (1 hour) of acquisitions. The following figure (Figure 4.4) illustrates the various steps involved in processing a record, from defining the roadmap to saving the data. A test dataset was created in order to test the model with data not submitted during the training phase. This dataset test includes new roads not exploiting by the training dataset and some roads are the same but the driver driving in opposite sens. As explain above, it's important to create a dataset with various characteristics, thus

the model can interpret several situations and avoid overfitting to only one. The dataset created has been made available to other research works at the Heudiasyc laboratory.

4.3 Model

As explained in section 4.1, the model must be capable of making temporal predictions of the vehicle's speed, considering past temporal data. To achieve this, we opted for a recurrent model, specifically of the Long Short-Term Memory (LSTM) type. The data we work with contains complex and diverse representations. To handle this, the model performs data compression using encoder networks upstream of the recurrent network. Specifically, we employ Convolutional Neural network (CNN) for this compression. Importantly, this compression is adjusted during the training of our model, rather than being imposed by a preprocessing phase separated from the learning phase of the global model. This approach helps us avoid introducing bias into the learning process. The proposed prediction model takes into account various inputs and modalities of data. The global architecture is schematized in figure 4.7, which illustrates the input and output models. These models are listed as follows:

4.3.1 Input Models

We set up a model for each type of input that will pre-process this data, by developing a Tensorflow-based framework. For each prediction, we submit 50 previous data per input model (= 5s of data). Depending on the nature of the input data, we can take two different approaches:

Raw Data

The raw data cannot be injected directly into the final model because it represents too much information of little significance, which would risk drowning the other more significant data. We decided to process on the data upstream, in order to compress the information as much as possible. This compression is based on an encoding model, like those found in auto-encoder models [96], the final model is given the encoded data of the encoding part. Among the list of data, we have to encode some data, from the map image and from the LiDAR. Data from the map image are encoded by a VGG16 model [97] (convolution2D/pool2D layers then fully connected to dense layers) and the LiDAR is encoded by a similar model but in 1 dimension. Note that for the map, we use a single data and not a time series, as we consider that only the last data is needed for the model to predict the future behavior of the car. Therefore, we repeat the encoded vector so that the output can be adapted to other models.

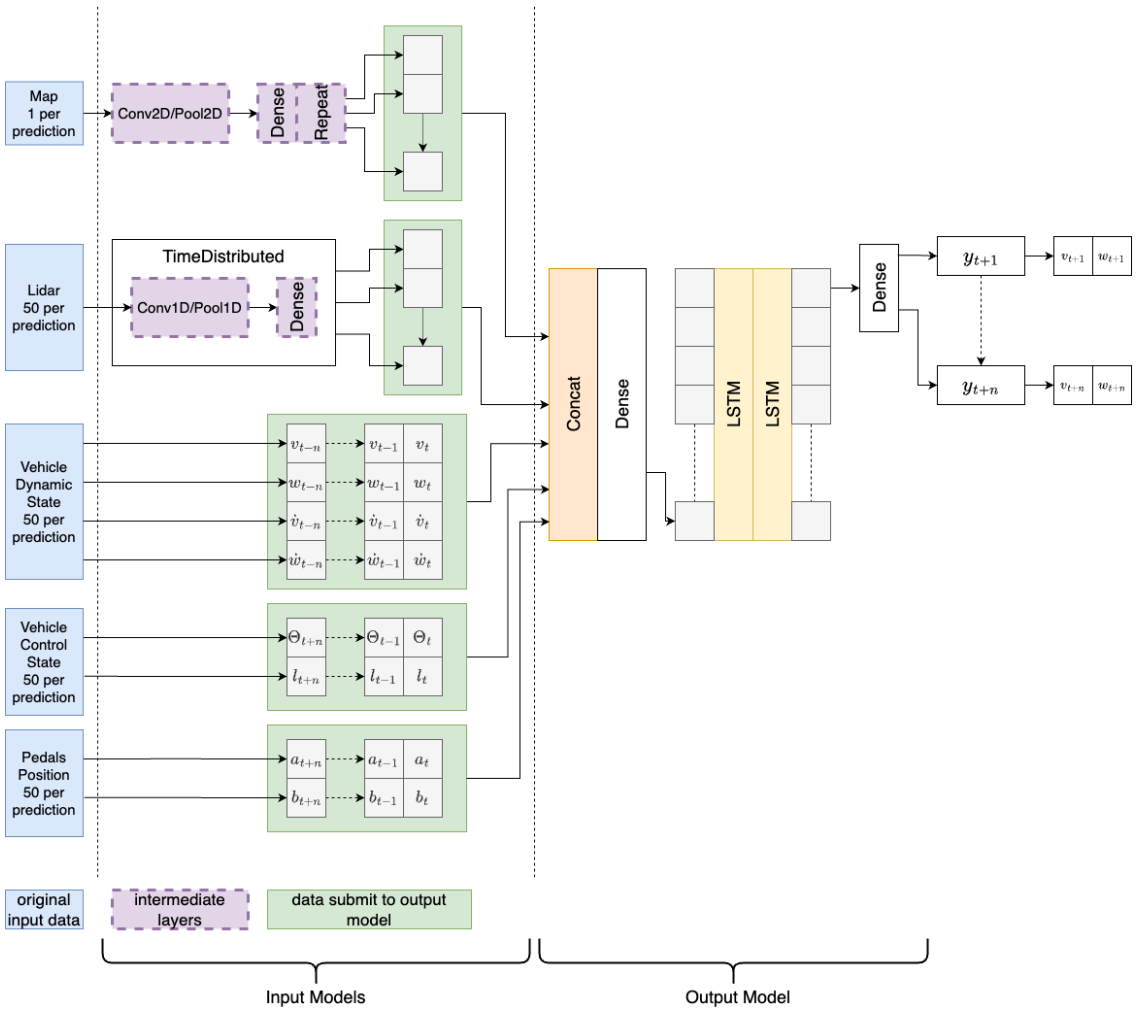


Figure 4.7: Global model architecture.

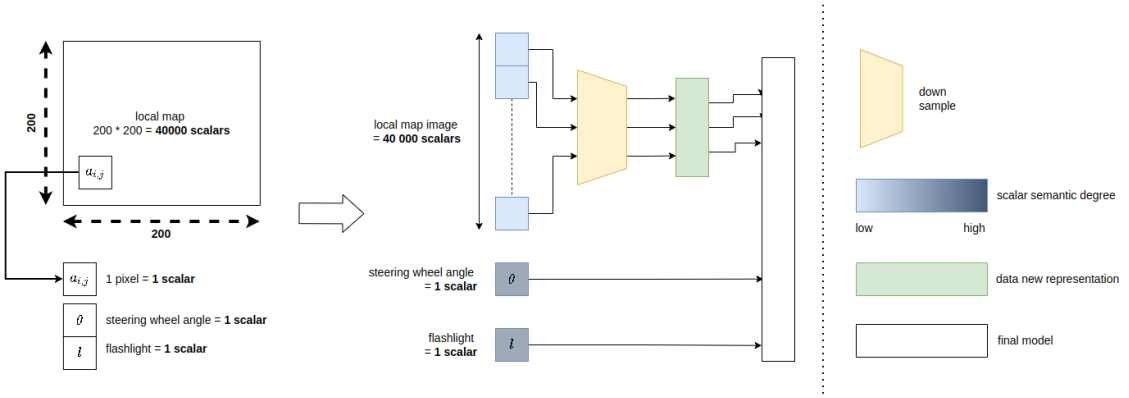


Figure 4.8: Unbalance data explication, example: raw image is defined by 40 000 scalars, compared to steering wheel angle defined by only one scalar.

Semantically Rich Data

The semantic data (for example, vehicle dynamic state and control) are already rich and may not be further compressible. These data are presented to the model as they are. Internal tests conducted allow us to affirm that amplifying the data using a decoder does not lead to better convergence.

Unbalance Data

The figure 4.8 explains the idea of unbalance data. The size of a raw data is more important than the size of a semantically rich data. For this reason, we have decided to create own framework able to create an input model per input, by this way we can apply a previous model able to extract features from heavy data, avoiding to hide high data with a small representation.

In order to properly size the model that encodes the local map, we conducted a preliminary test. In this test, an auto-encoder network was defined with a latent space. The purpose of this test is to find an encoder, latent space, and decoder configuration that maximizes information retention, specifically for a map, when processed by the neural network. This ensures that the information within the latent space is nearly complete. In other words, the raw information from the map input to the network is encoded with minimal loss into the latent space. The input images have dimensions of 200×200, and the proposed encoding reduces the data to a one-dimensional space of size 10.

4.3.2 Output Model

All the outputs of the input models are concatenated into a single time vector tensor. At this stage, the size of the input is reduced to a vector of size 28. This reduction is due to the compression of different inputs. The time vector is passed through a neural network before being fed into the recurrent network to perform an initial inference before submitting the data to the recurrent network. This output is

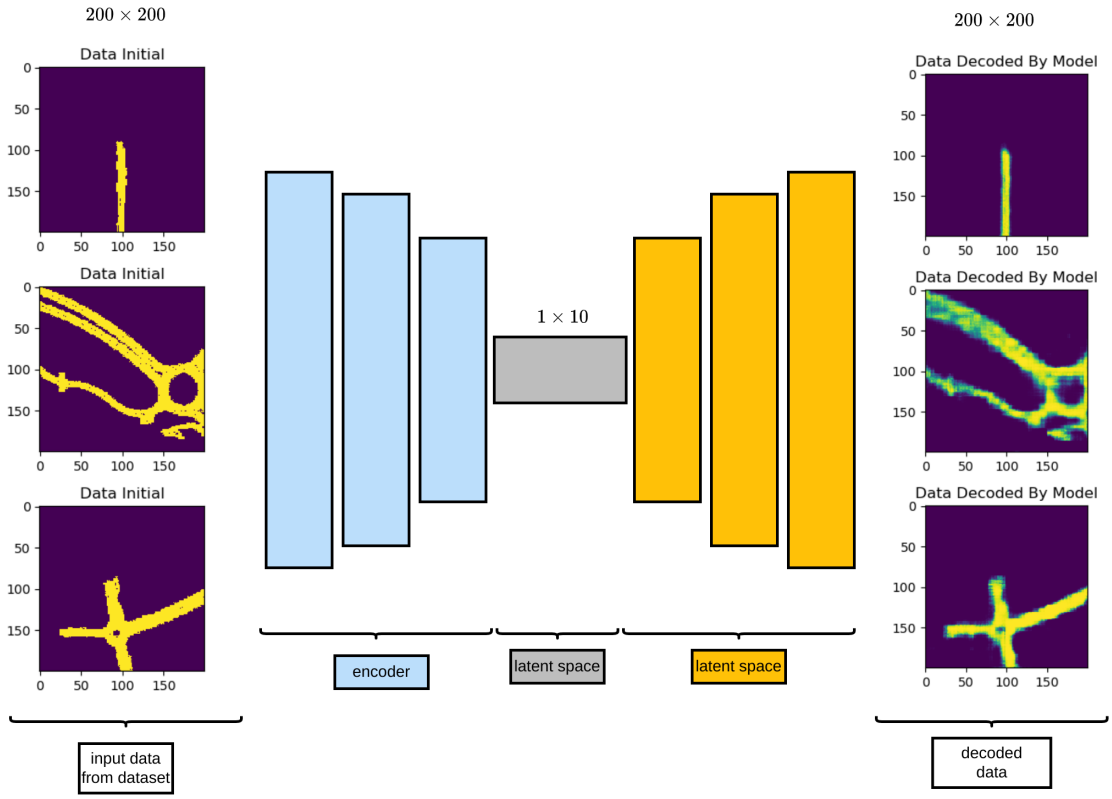


Figure 4.9: Input model maps latent space test. In this test, an autoencoder model has been created and trained on map from the dataset. This autoencoder has to compress information and re-build initial information from the compress information of the latent space.

Test name	N roundabouts	Speed limit	distance	N lanes	Time record	N Tests
roundabout	3	70 km/h	1.6 km	2	125 s	1200
city	2	50 km/h	1.6 km	1	170 s	1650
speed (1 lane)	1	70 km/h	2.8 km	1	170 s	1650
speed (2 lanes)	0	90 km/h	2 km	2	100 s	950

Table 4.4: Tests dataset informations.

then submitted to a recurrent model composed of 2 LSTM layers (100 units) inspired by the literature models [73] and adapted after some tests. However, only the final value of the recurrent network is utilized. The models proposed in the literature [73] make use of all the outputs of the recurrent network, but this requires creating a very large final network, which slows down the model's execution and inference. The output is reshaped into a vector of 50 velocity pairs (v_t, w_t), with a delta time of 100 ms between each pair, resulting in a total prediction time of 5 seconds.

The loss function used during the training is a Mean Error Absolute (MAE) weighted:

$$loss(y_{predict}, y_{true}) = w_v \cdot \sum_{i=0}^{n} |y_{predict,v,i} - y_{true,v,i}| + w_w \cdot \sum_{i=0}^{n} |y_{predict,w,i} - y_{true,w,i}| \quad (4.5)$$

Where $y_{predict}$ defined the output of the model, y_{true} the data from dataset, and w_v, w_w the weights per axis. During the training phase, we observe, the convergence is really fast on linear velocity than on the angular velocity. This is the reason why we choose the weighted version and attribute a weight more important on angular velocity axis.

4.4 Validation And Results

4.4.1 Test

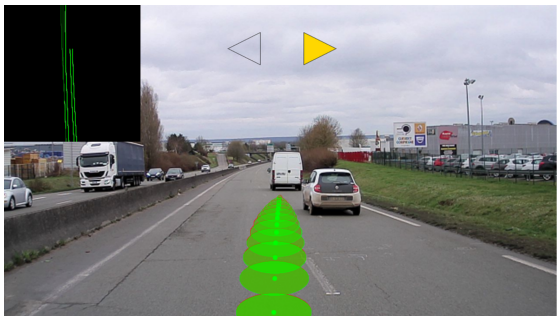
The test dataset was constructed to test several situations to best assess the fit of the proposed model to the situation. The table 4.4 shows information about these tests.

In these various tests, we assess prediction quality by calculating the cumulative error (Eq. 4.3). For each test, the cumulative error is computed over a sliding window of 50 data points, as our prediction consists of 50 values. Thus, on a dataset with 1000 timestamps, we conduct $(1000 - 50) = 950$ tests, where 50 represents the window prediction size. For each dataset, we calculate the median cumulative error over the sequence (Fig. 4.11). We opted for the median over the mean because the mean is sensitive to extremes. This choice ensures that the calculations are less affected by outliers resulting from poor predictions. In addition to these calculations, we have computed the mean of the cumulative error exclusively for

(a) Test roundabout.

(b) Test city.

(c) Test speed (1 lane).



(d) Test speed (2 lanes).

Figure 4.10: Image from the test dataset, for each test. On each image, in red, the projection of the actual velocity from the dataset, and in green, the projection of the prediction made by the model.

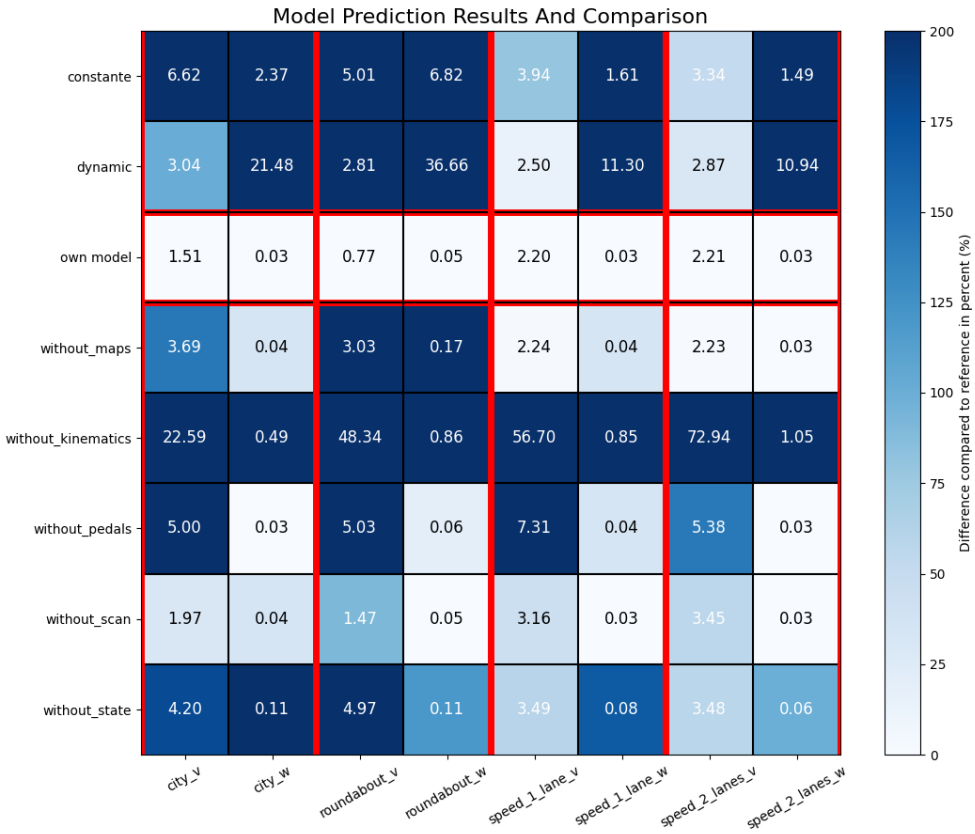


Figure 4.11: This table represents the median errors (Eq. 4.3) for each test. On the horizontal axis, it indicates the type of model used, and on the vertical axis, it specifies the type of test conducted. The color of each cell represents the percentage variation from the reference value (own model).

our model (Fig. 4.12). Since its predictions are not aberrant, the mean provides a reliable representation of its prediction quality. Figures 4.13 shows the prediction realized by our model on the test 'city'. On this plot distance sequence between 2 executions is 50 (window length prediction), by comparison during the evaluation the distance is only 1.

4.4.2 Results

Figure 4.10 illustrates the projection of the predictions made by our model, based on a test from our dataset. Figure 4.11 illustrates the median values for each test scenario: city, roundabout, speed 1 lane, and speed 2 lanes and for each axis (v and w). The "own model" line represents the model defined in this chapter. These various tests demonstrate that the model is capable of making high-quality predictions, whether for linear or angular velocity. The error in linear velocity is

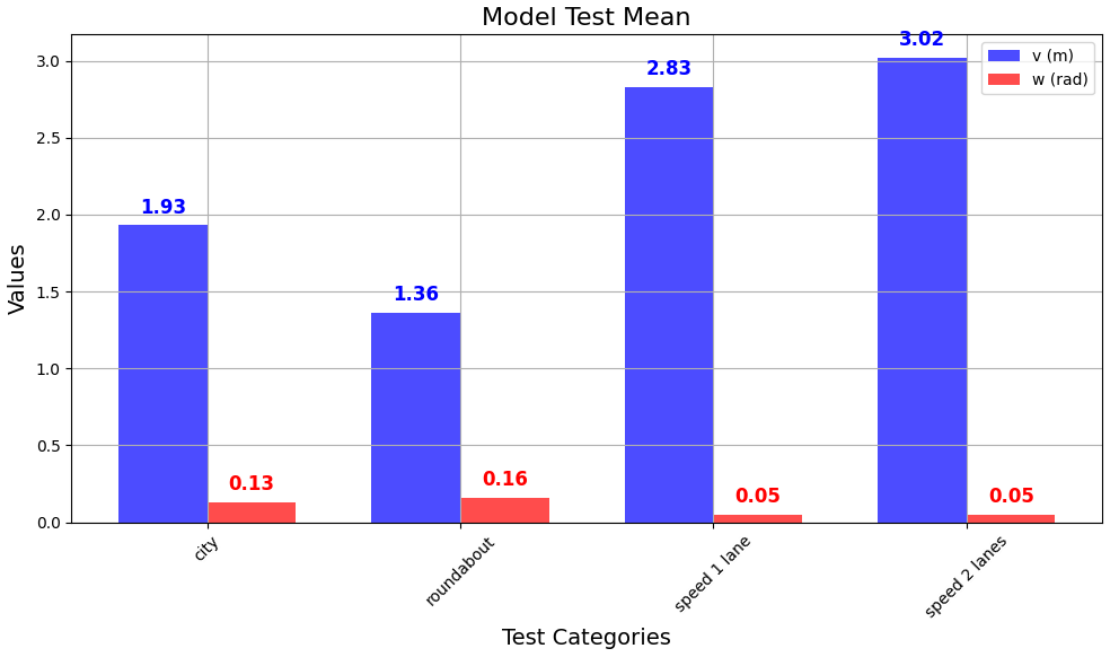
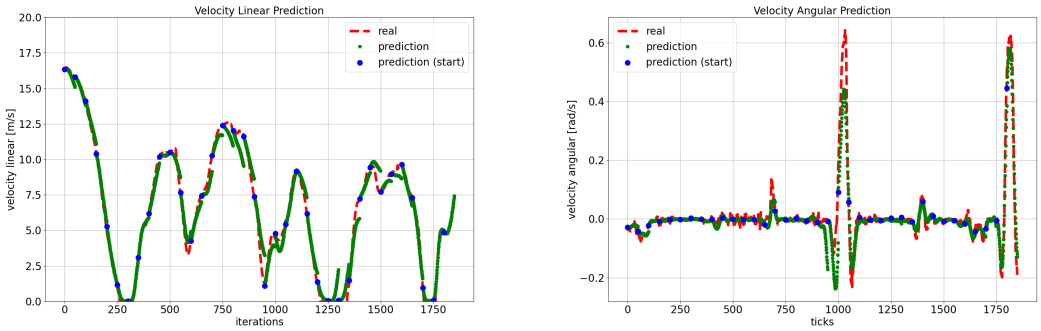


Figure 4.12: Average cumul error (Eq. 4.3) on each sub test. Applied only on the model method, because, average is too sensitive to outliers.



(a) Linear velocity prediction.

(b) Angular velocity prediction.

Figure 4.13: Intention predictions performed by our model on the test "city".

more significant in tests where the vehicle is moving at higher speeds, ranging from $3.02m$ in the worst case to $1.36m$ in the best case (Fig. 4.12). Linear velocity error is greater when the road speed limit is higher, but in the city and roundabout tests, where linear velocity is lower, variations are more significant due to the complexity of driving. This demonstrates the model's ability to adapt to complex driving scenarios. The error in angular velocity is more pronounced in the city and roundabout tests (0.16 rad in the roundabout test compared to 0.05 rad in the fast speed tests), exceeding 2 times the error of other tests, due to the complexity of the scenarios and higher angular velocities in these tests. In addition to the error induced by the model, we have calculated the error induced by the so-called constant and dynamic approximation, as mentioned in the introduction. From these results, it becomes evident that in cases of non-complex driving with relatively low velocity variations, the dynamic method for linear velocity provides results with acceptable errors (Fig. 4.11). However, even in such situations, accurate prediction of angular velocity remains challenging. Furthermore, in complex driving scenarios (city, roundabout), the proposed solutions yield significantly higher errors than our model. These results emphasize the importance of using our model for prediction and the significance of incorporating information beyond vehicle dynamics.

4.4.3 Sensitivity Of The Model To Data

In addition to evaluating the quality of the model, we can examine the sensitivity to the data. To do this, we have hidden the information of some data by placing them in a neutral state in some tests. The idea is to interpret in which situation the model exploits these data. The following table 4.5 shows the neutral data corresponding for each data tested. Based on the results (Fig 4.11), it is possible to interpret the information that is essential to the model in a particular situation. The tests begin by 'without' on Fig. 4.11 represents results with the model where one of input model is put to neutral position. In the absence of kinematics data, the model cannot predict coherent values, indicating that the model has inferred the importance of this data very well for predicting the car's future velocities. From this data, we can see that information about pedal positions is essential for predicting linear velocity, but its absence does not affect angular velocity. Map information is important for making predictions in a roundabout and urban context, as shown by the city and roundabout tests in the "without maps" configuration. The test without lidar data (without scan) shows that lidar data is used to determine linear velocity, as evidenced by the much larger errors in linear velocities. State information (steering wheel positions and turn signals) is used for predicting linear velocity and also has an impact on angular velocity.

The table 4.6 summarizes the impact of the data according to situation and on the prediction axis (linear and angular). Based on this table, in most situations, the data improves our model's prediction for linear and angular velocities. However, we note that the pedal and scan data are only useful for linear speed prediction.

Data	Neutral data
map image	zeros vector (= black image, map missing)
scan	ones vector (= no obstacle)
state	zeros vector (= flashlight never on and steering wheel angle fixed to middle position)
pedals	zeros vector (= pedals not used)
kinematics	zeros vector (= vehicle isn't moving)

Table 4.5: Neutral data corresponding.

Data	Impact
maps	city (linear), roundabout (linear and angular)
kinematics	all situations (linear and angular)
pedals	all situations (linear)
scan	all situations (linear)
state	all situations (linear and angular)

Table 4.6: Impact of data on prediction, depending on situation (city, roundabout, 1 lane or 2 lanes) and axis (linear or angular).

4.5 Conclusion

In this chapter, we have defined a prediction model capable of predicting human intentions in the short term. The creation of this model is justified by the complexity of predicting human intentions, especially in complex situations. This study also provides insight into the information utilized by the model in various scenarios, highlighting the relationship between these data types and prediction in these specific cases. In the future work, we want to build a similar model capable of predicting less predictable and more dangerous behaviors, we plan to include simulated data to add dangerous driving data.

5 Intention Evaluation

Abstract: *Determining the admissibility and quality of driving intentions, whether generated by an automated intelligent vehicle or a human driver, is a crucial task in the context of shared navigation between humans and intelligent vehicles. This chapter presents a generic method for quantifying driving intentions, which are defined as sequences of velocities. This formulation is based on metrics that have been discussed in the existing literature. It offers a way to utilize these metrics to evaluate a single state, make judgments, and extend this evaluation to a sequence of states. This quantification determines whether an intention can be safely achieved and define its quality, considering various criteria such as safety, comfort, context, and energy consumption. This approach enables us to compare and rank intentions based on a set of predefined criteria. In this chapter, we introduce a proposed implementation that has been tested on a driving simulator. We conducted tests using various intentions within a given scenario to demonstrate the value of the solution and the ability to compare these intentions with each other.*

Contents

5.1 Motivation

In order to guide the fusion of human and autonomous system intentions, it is necessary to quantify these intentions, to validate whether these intentions are admissible, i.e., whether the intention is achievable according to certain criteria, and to define the quality of these intentions, i.e., whether they are good or bad. Intentions are defined by a sequence of commands:

$$C_{t,t+m} = \{c_t, \ldots, c_{t+m}\} \tag{5.1}$$
$$C_{t,t+m} = \{(v_t, w_t), \ldots, (v_{t+m}, w_{t+m})\} \tag{5.2}$$

Here, t represents the current time, m the intention sequence length,v_i represents the linear velocity and w_i the angular velocity. This quantification relies on a set of metrics from the literature [77] representing a situation in a numerical way. In addition to the evaluation, intentions are judged based on a set of criteria. These different criteria can be of different natures: safety, context or comfort for example. Thus, unlike traditional approaches [77], the evaluation does not focus solely on safety. As shown in Figure 5.1, the evaluation of this sequence results in two quantities. Thus, the evaluation is based on two concepts:

- **Admissibility**: Defines, according to certain criteria, whether an intention is feasible or not;
- **Quality**: Defines the quality of an intention, between 0 and 1, according to certain criteria;

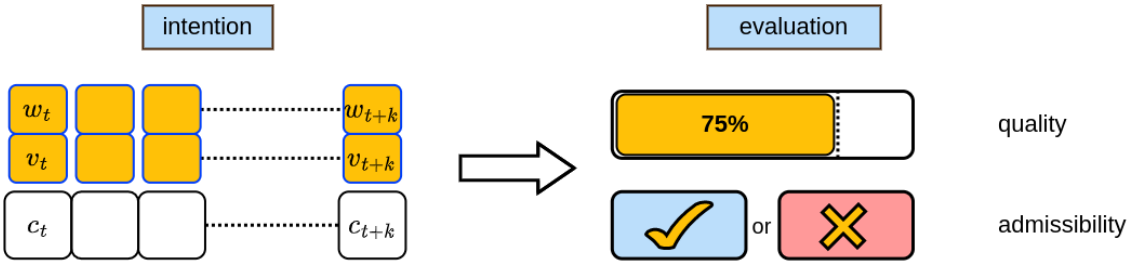


Figure 5.1: Quantifications (quality and admissibility) results on driving intention.

5.2 Assessment Metrics In State Quantification

In the literature, there exists a set of metrics [77] capable of quantifying a state, which can describe the state of the vehicle and/or the state of the environment and other occupants on the road. Our idea draws inspiration from reinforcement learning, where the evaluation of an action involves assessing the resulting state (s_{t+1}) after the action has been applied. Adapted to the problem, a command (c_t) can be evaluated by analyzing the resulting state (s_{t+1}). Figure 5.2 illustrates the concept. Thus, we can extend this approach to a sequence of commands, where

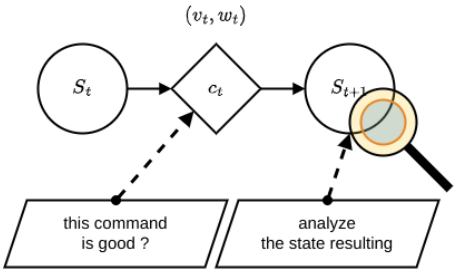


Figure 5.2: Command is evaluated in analyzing the state resulting after applying the command on an initial state.

each command is evaluated using this principle. The overall evaluation will then be the combination of these individual assessments.

5.2.1 Definitions

By knowing the sequence of commands, i.e., the intention C_t, and the current state s_t provided by the sensors, the sequence of future states can be predicted.

$$(C_{t,t+m}, s_t) \xrightarrow{P} \{\hat{s}_{t+1}, \ldots, \hat{s}_{t+m}\} \tag{5.3}$$

Here, $\hat{s}_i$ represents the state after the application of the control input (v_i, w_i). The quantization should be more sensitive to a state that is close rather than far, as the impact of a nearby action is more significant. Additionally, the error in intention prediction performed by the model becomes increasingly important when the predicted command is far from the current state. Let the quality of a state, s_i, noted as $quality(s_i)$, the quality of a control sequence, $C_{t,t+m}$, on the current state, s_t is defined by:

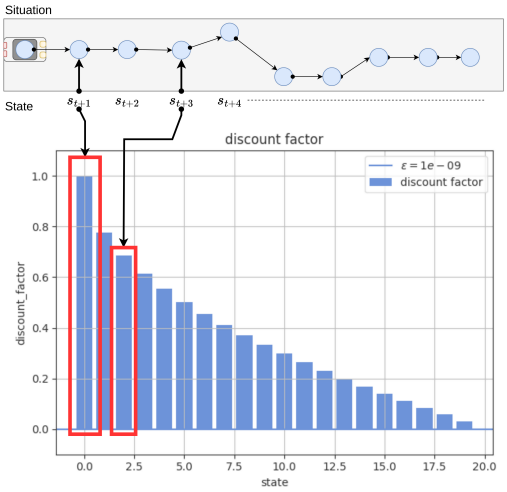
$$Quality(C_{t,t+m}, s_t) = \frac{1}{\sum_i^m \gamma(i)} \sum_{i=1}^{m} \gamma(i) \cdot quality(s_{t+i}) \tag{5.4}$$

With $\gamma : \mathcal{N}^+ \to [0, 1]$, which is a decreasing function referred to as the discount factor. This implies that the quality of a state that occurs later has less impact on the final result than a state that is closer. Figure 5.3a illustrates the relationship between the discount function and the states. For each state, a discount value is associated, representing the impact of studying that state on the final quantification. As shown in the figure, this value decreases as the states progress. This approach is similar to the discount rate used in reinforcement learning [98].

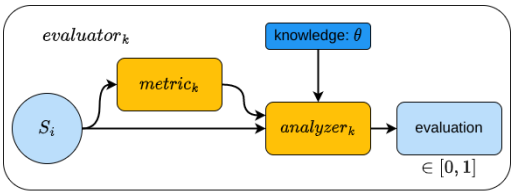
In the section 5.3.3, the discount fonction is adapted according to the error prediction made by the intention predictive model defined in chapter 4.

5.2.2 Evaluator

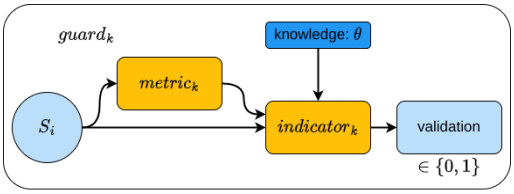
To assess the sequence of commands, it is necessary to determine the qualification criteria for each state, as mentioned in Section 5.1. This quantification process



(a) Discount factor graphic. States are defined by the vehicle speed projection, and it's associated a discount value by state.



(b) Evaluator blocks.



(c) Indicator blocks.

(d) Explanation of two types of blocks and a discount factor graphic.

Figure 5.3: Concepts explanations of intention evaluation.

involves multiple criteria. Our approach involves defining an evaluator, denoted as $eval_k$, for each criterion k. Each evaluator consists of a composition of two functions: a metric function, $metric_k(s_t)$, designed to quantify a situation, without passing judgment. For this reason, the function's output is subsequently analyzed by an analyzer function, which evaluates this result based on prior knowledge defined by Θ, denoted as $analyzer_{k|\Theta}(metric_k(s_t), s_t)$. Each evaluator is defined by a pair of functions: $(metric, analyzer)$. Figure 5.3b illustrates the relationships between these $metric$ and $analyzer$ functions. The $metric$ function exploits information from the state to quantify it, while the $analyzer$ analyzes this $metric$ output, using its own knowledge and taking the state into account for the analysis.

$$eval_k(s_t) = analyzer_{k|\Theta}(metric_k(s_t), s_t) \tag{5.5}$$

For each of these criteria, we can vary its influence on the final result by assigning a weight w_k. Thus, the quality of a state is defined by:

$$quality(s_t) = \frac{1}{\sum_k w_k} \sum_k w_k \cdot eval_k(s_t) \tag{5.6}$$

5.2.3 Quality Generic Formulation

By substituting Eq. 5.4 into Eq. 5.6, the generic formulation is as follows:

$$Quality(C_t, s_t) = \frac{1}{\sum_i^m \gamma(i)} \sum_{i=1}^{n} \gamma(i) \left(\underbrace{\frac{1}{\sum_k w_k} \sum_k w_k \cdot \underbrace{analyzer_{k|\Theta}(metric_k(s_{t+i}), s_{t+i})}_{eval_k}}_{quality_{t+i}} \right) \tag{5.7}$$

Equation 5.7 is the quality of an intention C_t computed at the instant t. This quantity is normalized within the interval $[0, 1]$. This normalization enables easy comparison of qualities between different intentions.

5.2.4 Admissibility Generic Formulation

Similarly to quality, admissibility is defined by a set of $guard_k$ that guarantee for each criterion a check that the command is indeed admissible. Each guard is composed as follows:

$$guard_k(s_i) = indicator_{k|\Theta}(metric_k(s_i), s_i) \tag{5.8}$$

Where the function *indicator* indicates whether the *metric* of a state is admissible ($= 1$) or not ($= 0$).

As shown in Figure 5.3c, the *metric* function quantifies the submitted state. This quantification is then analyzed by the *indicator* function, which determines whether the metric value is acceptable or not based on the studied state and its knowledge. Thus, the admissibility of a sequence of commands C_t, from a state s_t, contains n criteria on a sequence of τ, unlike the quality, the elements are multiplied together so that if one element is inadmissible, the whole sequence is inadmissible.

$$Admissibility(C_t, s_t) = \prod_{i=1}^{\tau} \left(\underbrace{\prod_{k=1}^{n} guard_k(s_{t+i})}_{admissibility_{t+i}} \right) = \prod_{i=1}^{\tau} \left(\prod_{l=1}^{n} indicator_{l|\Theta}(metric_l(s_{t+i}), s_{t+i}) \right) \tag{5.9}$$

5.2.5 Intention Score Generic Formulation

From the generic formulation of quality and admissibility, the generic formulation of score, is defined as follows. In this proposal, we add a hyperparameter ϵ, which allows to modify the domain of definition of the quality $[0, 1] \rightarrow [\epsilon, 1]$. Thus, we can define the value of an intention as follows:

$$Score(C_t, s_t) = Admissibility(C_t, s_t) \times (\epsilon + Quality(C_t, s_t) * (1 - \epsilon)) \tag{5.10}$$

The parameter ϵ represents a very small positive quantity used to prevent a quality of zero from resulting in a score of zero. This difference is crucial in the context of our work as it enables us to distinguish between a valid intention with poor quality and an invalid intention. Thus, we differentiate between a high-quality command that is not feasible and a low-quality command that is feasible.

Figure 5.4 illustrates the entire evaluation process. First, a prediction of the states is made. Then, the admissibility of each state is calculated to determine the admissibility of the entire intention. Subsequently, the quality of each state is independently computed, and these individual evaluations are combined to determine the overall quality of the intention, as shown in the figure. Finally, the overall score for this intention is remapped.

5.3 Implementation And Validation Tests

5.3.1 State Definitions

The current state, denoted as s_t, can be subdivided into two substates: s_t^{env}, which describes the current state of the environment, and $s_t^{vehicle}$, which characterizes the current state of the vehicle.

$$s_t = \{s_t^{env}, s_t^{vehicle}\} \tag{5.11}$$

State Vehicle

The state of a vehicle is defined by the dynamic state of the vehicle. So the state of a vehicle is defined as follows:

$$s_t^{vehicle} = \begin{bmatrix} v \\ w \\ \dot{v} \\ \dot{w} \end{bmatrix}_t \tag{5.12}$$

Here, v represents linear velocity, w represents angular velocity, $\dot{v}$ represents linear acceleration, and $\dot{w}$ represents angular acceleration. By definition, c_t represents the command applied to the vehicle. c_t is defined as the velocity pair (v_t, w_t) that will be applied to the vehicle. Thus, based on the state definition, the next state, denoted as $\hat{s}_{t+1}^{vehicle}$, can be estimated using the following relation, which accounts for the time interval (δt):

$$\hat{s}_{t+1}^{vehicle} = \frac{1}{\delta t} \left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} s_t^{vehicle} + \begin{bmatrix} \delta t & 0 \\ 0 & \delta t \\ 1 & 0 \\ 0 & 1 \end{bmatrix} c_t \right) \tag{5.13}$$

In this relation, accelerations are determined by comparing the initial velocities defined by the state to the new velocities defined by the command over a given time interval δt.

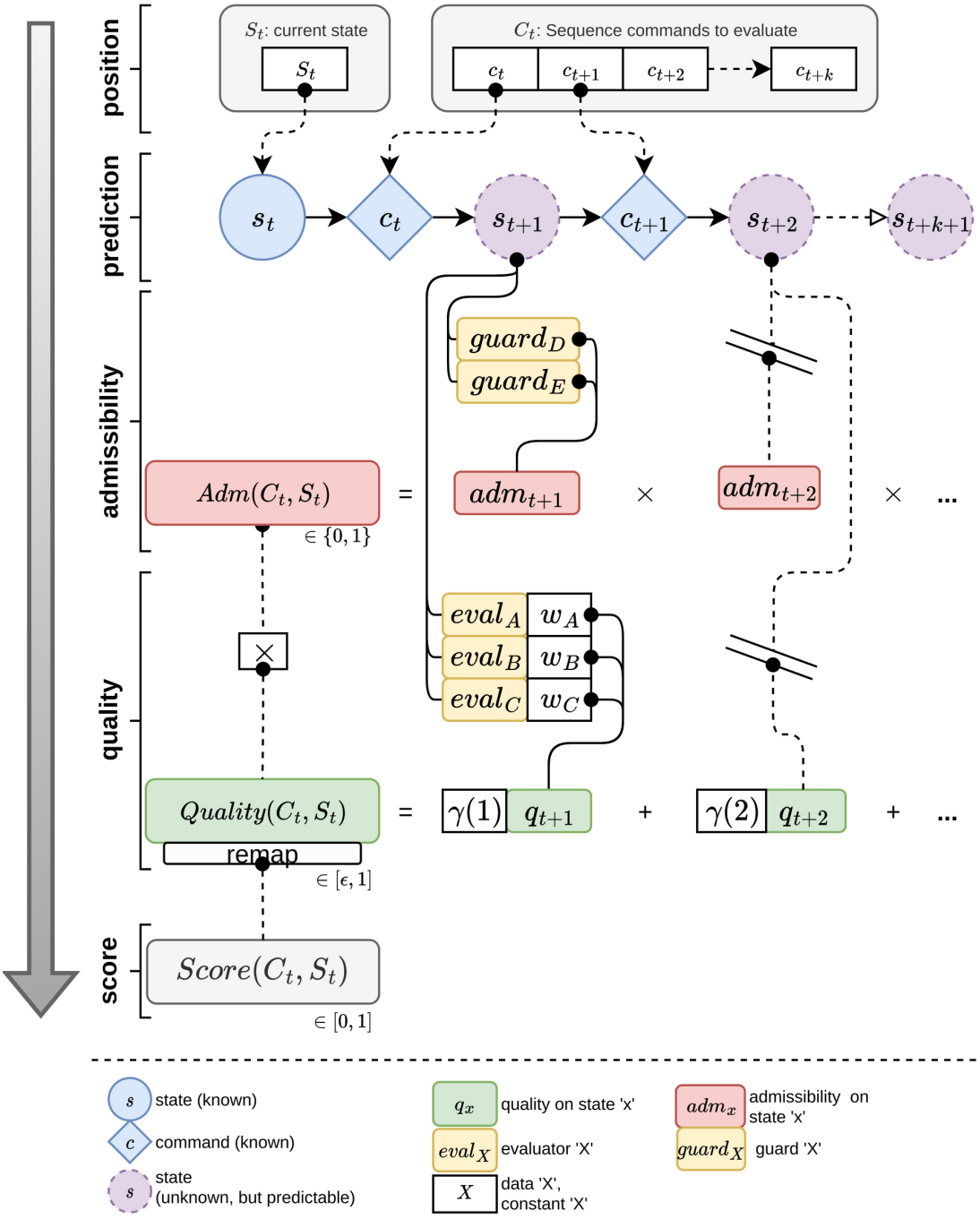


Figure 5.4: Score computation illustration. From the state prediction to the final intention score. Including admissibility and quality computations.

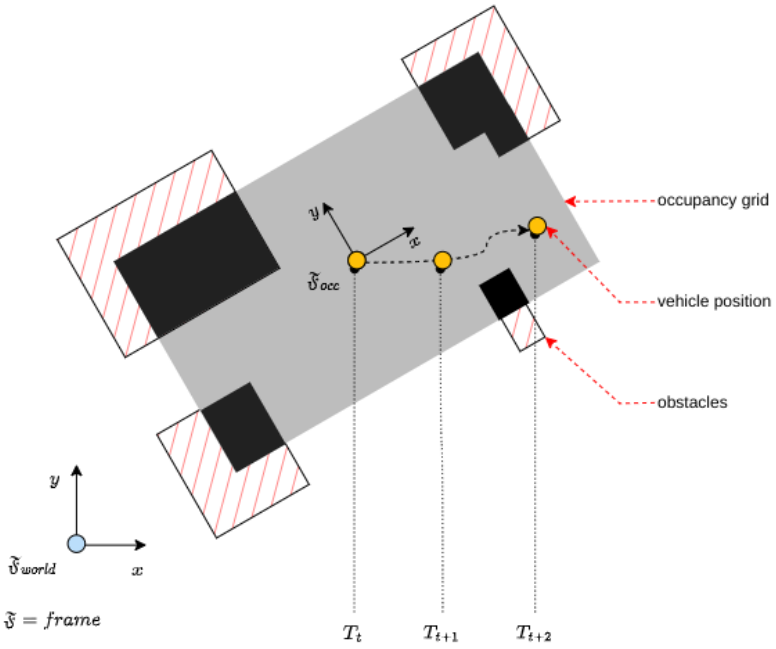


Figure 5.5: Vehicle velocity projection in occupancy grid frame.

State Environment

The state of the environment is defined by an occupancy grid, $grid_t^{occ}$, and the position of the vehicle in this grid,$(x, y, \theta)_t^g$, (where g represents the grid frame,$\mathfrak{F}_{occ}$, avoiding confusion with the real vehicle position in the world frame, $\mathfrak{F}_{world}$):

$$s_t^{env} = [grid_t^{occ}, (x, y, \theta)_t^g] \tag{5.14}$$

Figure 5.5 shows the relation between world frame and grid frame, and the vehicle projection in the grid. At the state S_t, the vehicle position is on the origin of the occupancy grid frame. As explained in the previous sections, we need to predict the future state, i.e. the future occupancy grid in this case. At first, we assume the environment of the car as static, so making a prediction of the future grid is like making a projection of the speed in trajectory and repositioning the position of the car on the current occupancy grid. The projection exploits these following relations:
if $w \neq 0$:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \end{pmatrix} + \frac{v_t}{w_t} \begin{pmatrix} cos(\theta_t) & -sin(\theta_t) \\ sin(\theta_t) & cos(\theta_t) \end{pmatrix} \begin{pmatrix} 1 - cos(w_t.\delta_t) \\ sin(w_t.\delta_t) \end{pmatrix}$$
$$\theta_t = \theta_t + w_t\delta_t \tag{5.15}$$

if $w = 0$:

$$\begin{pmatrix} X_{t+1} \\ Y_{t+1} \end{pmatrix} = \begin{pmatrix} X_t \\ Y_t \end{pmatrix} + \begin{pmatrix} cos(\theta_t) & -sin(\theta_t) \\ sin(\theta_t) & cos(\theta_t) \end{pmatrix} \begin{pmatrix} 0 \\ v_t.\delta_t \end{pmatrix}$$
$$\theta_t = \theta_{t+1} \tag{5.16}$$

The vector $(x, y, \theta)_t^g$ defines the position and the orientation of the vehicle on the occupation grid. We assume this occupancy grid is static, from the commands sequence combined to interval time δt, we can estimate the new position of the vehicle in the initial occupancy grid ($grid_t^{occ}$). The vehicle position can be estimated.

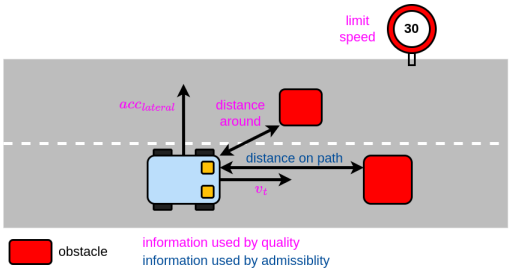
$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_t^g \xrightarrow[\delta t]{(v_t, w_t)} \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t+1}^g \tag{5.17}$$

Thus, the prediction of the next state, $\hat{s}_{t+1}^{env}$, is defined by:

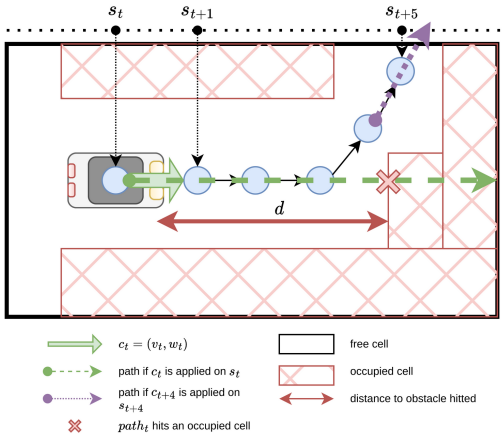
$$\hat{s}_{t+1}^{env} = [grid_t^{occ}, (x, y, \theta)_{t+1}^g] \tag{5.18}$$

5.3.2 Quality

In this validation process, we calculate the quality of intentions on sequences of 30 elements (with $m = 30$, projection time $\approx 3s$). In this implementation, we define 3 evaluators from 3 different categories (Table 5.10a). For each evaluator, we define the metric for quantifying the state and the analysis for assessing this quantification. Figure 5.6a represents information exploited by evaluators.



(a) Global information for quality and admissibility quantification processes.



(b) Determine the distance to the obstacle by applying the command, c_t, as a constant to the current state, s_t.

Figure 5.6: Quality and admissibility information explanation.

Lateral Acceleration (LA)

> This evaluator takes into account the comfort of driving. A high lateral acceleration causes discomfort for the driver, and in the case of a high value, it poses a safety risk to the human. Here, the metric focuses on comfort by defining a value below the safety threshold. Figure 5.7, illustrates the metric and the analyzer used by this function.

Metric From the linear velocity and the angular velocity, the lateral acceleration of the vehicle can be deduced by:

$$metric_{LA}(s_t) = |v_t \cdot w_t| \tag{5.19}$$

To illustrate the relationship between the metric and the analyzer, we consider a candidate example defined by $(w_t = 0.5, v_t = 2)$, resulting in a lateral acceleration $a_t = (v_t \cdot w_t) = 1.0, \mathrm{m/s^2}$. For this example, the metric yields a value of 0.61, as shown in Figure 5.7.

Analyzer This analyzer should penalize excessive lateral acceleration. A reference acceleration needs to be defined to assess whether this acceleration is too high. The article [99] defines the most acceptable lateral acceleration. Therefore, the most acceptable acceleration is defined as $\Theta_{a_{ref}} = 0.3g = 2.943, \mathrm{m/s^2}$.

$$analyzer_{LA|\Theta}(metric_{LA}(s_t), s_t) = \frac{1}{1 + e^{(\theta_{rampe} \cdot metric_{LA}(s_t) - \Theta_{a_{ref}})}} \tag{5.20}$$

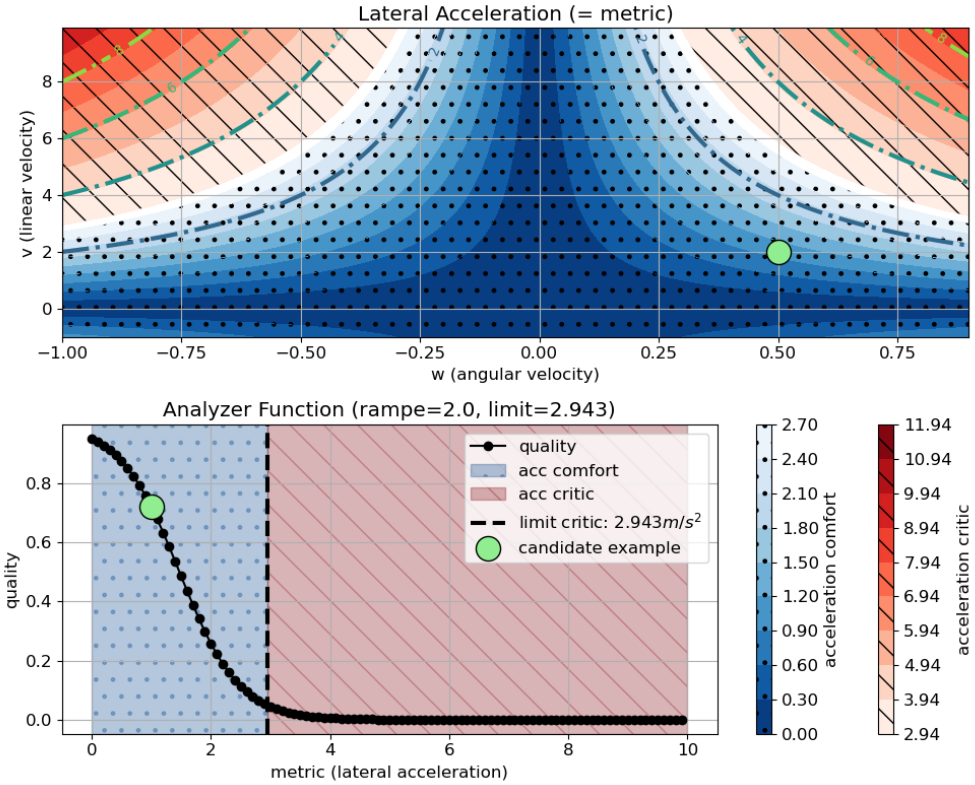


Figure 5.7: The top figure represents a metric derived from the vehicle state. Based on this metric, the function on the bottom side analyzes the result and assigns a judgment score between 0 and 1. This analyzer is defined by two parameters: *rampe* and $shift$.

Collision Around (CA)

This evaluator considers the distance between the vehicle and the nearest obstacle in order to ensure the safety of the driver. The aim is to assess if the driver is not driving too close to an obstacle. It is possible to deduce the distance to the nearest obstacle by calculating the distance between the vehicle's position defined by state s_t and the obstacles that describe this grid.

Metric For a given state s_t, this metric determines the closest obstacle around the vehicle by utilizing the occupancy grid.

$$metric_{CA}(s_t) = min(distance((x_t, y_t), O)) \tag{5.21}$$

where O is the list of obstacle positions defined in the grid frame, and (x_t,y_t) defines the position of the vehicle in the occupancy grid when the car is in state s_t.

Analyzer Similar to the previous analyzer, from a reference distance, $\Theta_{distance_{critic}}$, a distance too short is penalized.

$$analyzer_{CA|\Theta}(mertric_{CA}(s_t), s_t) = \frac{1}{1 + e^{(-\Theta_{rampe} \cdot metric_{CA}(s_t) - \Theta_{distance_{critic}}(s_t))}} \tag{5.22}$$

Speed Limit (SL)

Depending on the context in which the vehicle is operating, this evaluator assesses compliance with the speed limit imposed by the infrastructure. This metric ensures that the proposed intention ensures that the driver is driving at an optimal speed without causing any traffic disruption.

Metric This metric exploits the current speed of the vehicle.

$$metric_{SL}(s_t) = |v_t| \tag{5.23}$$

Analyzer The vehicle's speed should be close to the limit speed Θ_{target}. The analyzer utilizes hyperparameters Θ_{target} (the limit speed) and Θ_{σ} (the tolerance). This analyzer is a normalized Gaussian centered on the speed limit defined by the context.

$$analyzer_{SL|\Theta}(metric_{SL}(s_t), s_t) = e^{(\frac{1}{2} \cdot (\frac{metric_{SL}(s_t) - \Theta_{target}}{\Theta_{\sigma}})^2)} \tag{5.24}$$

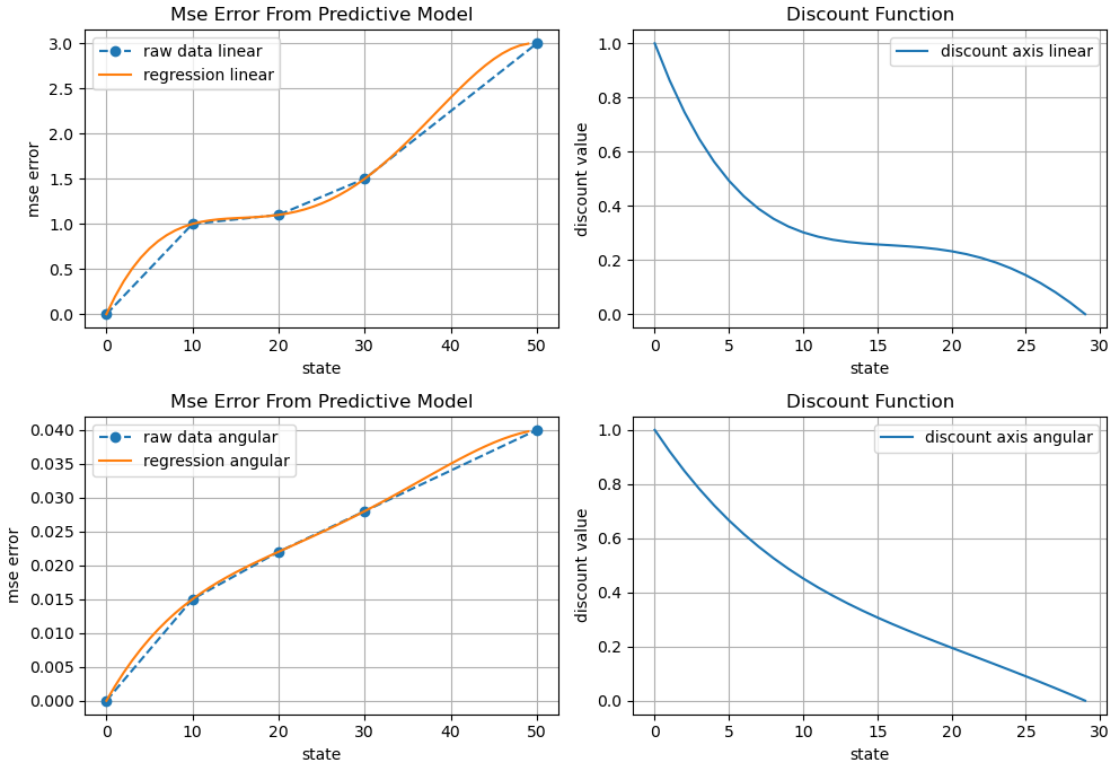


Figure 5.8: Discount function per axis computation.

5.3.3 Discount Factor

In this implementation, we relied on the prediction error made by the human driving prediction model (Chap. 4). From these errors, we can calculate the regression that interpolates these errors on each axis v and w. Let r_v and r_w be the respective regressions on v and w. Thus, from the regressions and the length of the sequence (m) on which we apply the following formulation:

$$\gamma_v(i) = \frac{\max_{j\in[\![0,m[\![}(r_v(j)) - r_v(i)}{\max_{j\in[\![0,m[\![}(r_v(j))} \tag{5.25}$$

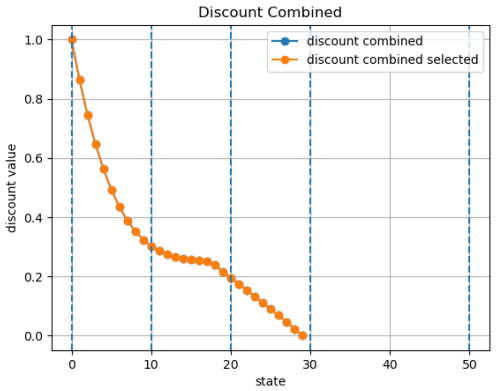
$$\gamma_w(i) = \frac{\max_{j\in[\![0,m[\![}(r_w(j)) - r_w(i)}{\max_{j\in[\![0,m[\![}(r_w(j))} \tag{5.26}$$

With $i \in [\![0, m[\![$. Figure 5.8 shows the error regressions and the associated discount function. The combined discount function is defined at each point by the minimum of the individual discount functions. Thus, the discount function is defined as follows:

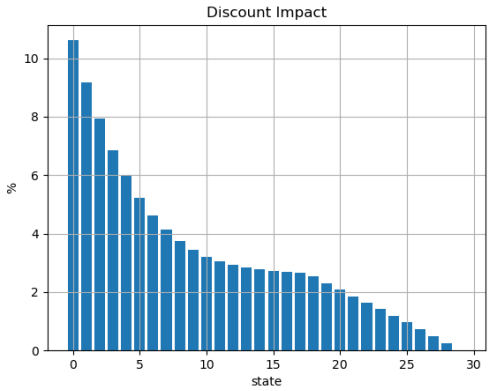
$$\gamma(i) = \min(\gamma_v(i), \gamma_w(i)) \tag{5.27}$$

Figure 5.9a represents the combined discount function from discounts function shown in Figure 5.8.

Note that this calculation is only done once since the discount function does not



(a) Discount combined graphic.



(b) Discount impact per state.

depend on the current state. The impact of a state is calculated as follows:

$$\text{impact}(i) = \frac{\gamma(i)}{\int_0^m \gamma(j)\delta j} \approx \frac{\gamma(i)}{\sum_0^m \gamma(j)} \tag{5.28}$$

Figure 5.9b represents the impact (in percentage) of each state on the final calculation.

5.3.4 Admissibility

Similar to the evaluators, the admissibility guards also consist of metric and indicator functions. For the analysis of the guards, we have selected the first 10 elements of the sequence, which is recorded at a rate of 10Hz. This sequence represents 1.0 second, corresponding to the human reaction time [79]. Additionally, intention prediction is more reliable within the first second, which confirms the decision to focus on the first 10 elements. Figure 5.6a represents information exploited by the guard.

Collision On Path (COP)

This evaluator assesses whether the intended drive avoids collisions with any obstacles in its path.

Metric This metric defines the distance, denoted as d_t, between the vehicle and the nearest obstacle for a given state s_t, assuming that the vehicle is moving at a constant speed defined by c_t. As explained in Section 5.3.1, the state of the environment, denoted as s_i, is determined by the vehicle's position relative to an occupancy grid, $grid_t^{occ}$. From this position, a projection of the current velocity (T is the trajectory), defined by the new dynamic state of the vehicle represented by $s_i^{vehicle}$. So, the position with the nearest obstacle can be known. (O obstacle positions defined by the occupancy grid). Figure 5.6b

illustrates how to calculate the obstacle distance by projecting the estimated path based on the command.

$$metric_{COP}(s_t) = min(distance(T, O)) \tag{5.29}$$

Indicator To evaluate the distance to obstacle, a reference distance need to be defined. Thus, a distance less than this reference distance will be penalized, and conversely, the farther a distance is, the more it will be rewarded. Based on the reaction time [79], $\Theta_{time_{react}} = 1.0s$, and on the maximum deceleration [79], $\Theta_{dec_{max}} = 3.3m/s^2$ and the current velocity, $v_t \in s_t$, the reference distance can be defined as:

$$distance_{react}(s_t) = \frac{v_t}{\Theta_{time_{react}}} \tag{5.30}$$

$$distance_{brake}(s_t) = \frac{v_t^2}{2 \cdot \Theta_{dec_{max}}} \tag{5.31}$$

$$distance_{safe}(s_t) = distance_{react}(s_t) + distance_{brake}(s_t) \tag{5.32}$$

Based on this estimation, the analyzer function is build in taking account this reference:

$$indicator_{COP|\Theta}(s_t) = metric_{COP}(s_t) > distance_{safe}(s_t) \tag{5.33}$$

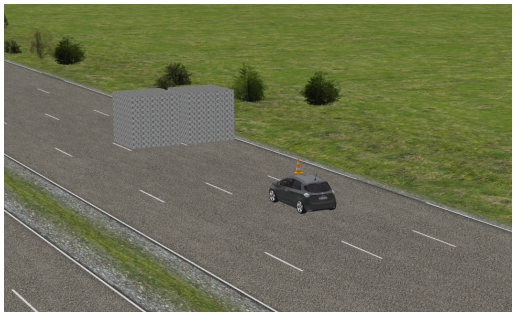
5.4 Results

The implementation described in the previous section is developed in Python. The solution was tested using the SCANeR studio simulator, allowing the generation of data for non-admissible intentions. Moreover, we wanted to conduct tests on recorded data to ensure that the concepts and methods, in order to test in isolation from the other parts the concepts and methods. Tests were conducted on the same scenario with three different intentions to demonstrate the efficiency of the solution in comparing these intentions. In this situation, the car is moving at a constant initial speed, and the vehicle is driving on a lane with obstacles placed in the same lane as the vehicle. The speed limit of the road is set at $30km/h \approx 8.3m/s$. The first intention leads to a collision with an obstacle, the second successfully avoids obstacles with high lateral acceleration, and the third successfully avoids obstacles with even better lateral acceleration.

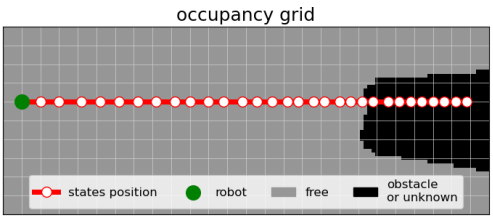
The following subsections study each intention, and further graphics are available in the appendix (Appendix. B).

Criteria	Category	Weight
collision around	safety	0.2
speed limit	context	0.7
lateral acceleration	comfort	0.1

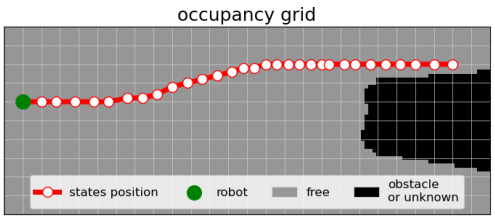
(a) Tests criteria.



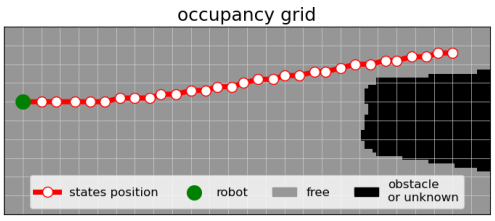
(b) Simulation situation.



(a) Intention 1.



(b) Intention 2.

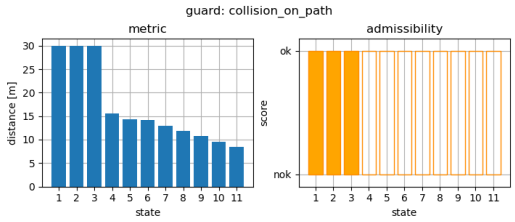


(c) Intention 3.

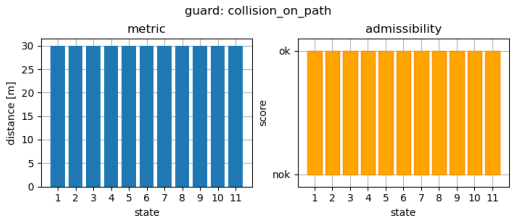
Figure 5.11: Occupancy grid with intention trajectory.

5.4.1 Intention 1: Hit Obstacle

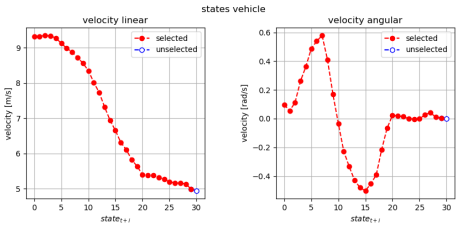
In the first intention, the vehicle is in motion and collides with the obstacle. Figure 5.11a displays the projected states of the intention on the current occupancy grid. If an intention is not admissible, its score is directly set to 0. In this scenario, the guard is not validated, as shown in Fig. 5.12a, because the vehicle is too close to the obstacle and doesn't have time to decelerate before collision. Consequently, the final quality and score in this situation are both 0.



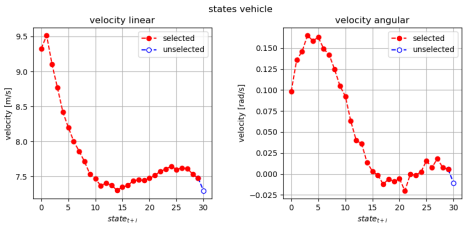
(a) Guard result (intention 1).



(b) Guard result (intention 2).



(a) Vehicle velocities (intention 2).



(b) Vehicle velocities (intention 3).

Figure 5.13: Vehicle states, linear and angular velocities.

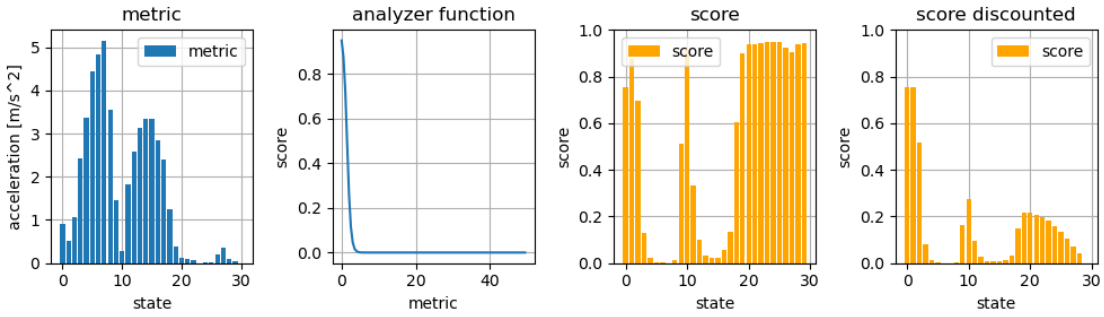
5.4.2 Intention 2: Avoid Obstacle With High Lateral Acceleration

In the intention the driver avoids the obstacle but makes an abrupt lane change. Like the previous intention, the Fig. 5.11b shows the occupancy grid combined with the trajectory of the vehicle estimated from the intention. The following Fig. 5.13a shows the speeds of the vehicle in future states. In this approach, the driver suggests decelerating to reduce the collision distance, which is proportional to the vehicle's speed. Thus, compared to the previous approach, all the guards are valid, indicating that the intention is admissible.

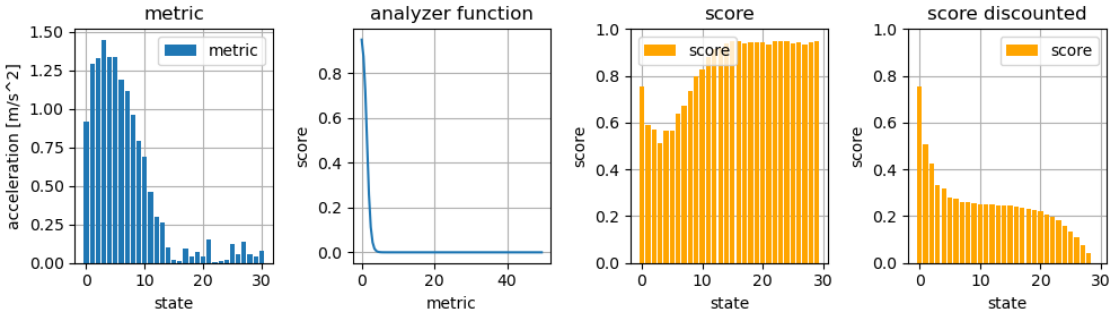
Figure. 5.15a shows the result on lateral accelerator evaluator. This figure shows that high lateral accelerations in the state neighborhoods of 8 and 15 impact the evaluation of this sequence. Refer to Figure 5.7 for a visual representation of the metric and analyzer functions. The final score is equal to 0.579.

5.4.3 Intention 3: Avoid Obstacle With Better Quality

In this case, the intention must be good. Instead of intention 02, the lane change is made earlier and in a more progressive manner, reducing the angular velocity (Fig. 5.13b) and, consequently, reducing lateral acceleration. Figure 5.14b illustrates the impact of the intention on the lateral acceleration evaluator. Figure 5.15b shows the quality per criterion and the global quality. The global quality is better than the previous intention (0.801 compare to 0.580). The intention score is 0.802.

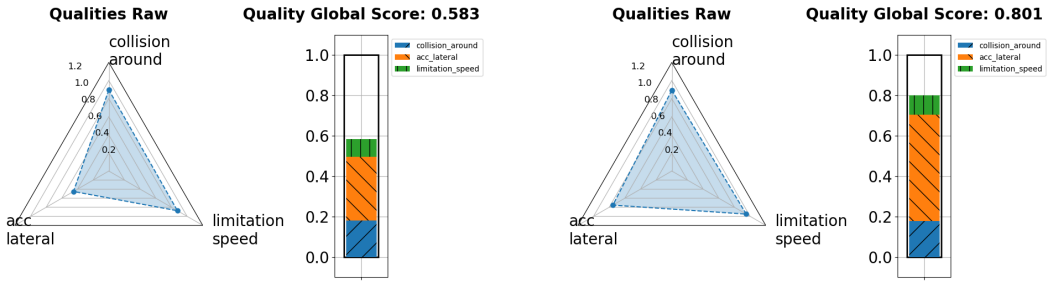


(a) On intention 2.



(b) On intention 3.

Figure 5.14: Accelerator lateral evaluator results.



(a) On intention 2. (b) On intention 3.

Figure 5.15: Quality results, per criteria and the global quality result.

5.5 Discussion

From previous studies, the Tab. 5.1 summarizes and compare the quantification of each intention.

intention	admissibility	quality	score
intention 1	0	-	0
intention 2	1	0.580	0.579
intention 3	1	0.801	0.802

Table 5.1: Summarize tests.

These evaluations (Appendix. B) of different intentions highlight the significance of the formulation and its effectiveness.

The first test (Sec. 5.4.1) emphasizes the need to define admissibility, as it allows an intention that poses a risk to vehicle users to override the quality. The intention is considered as undesirable due to its potential danger. This case underscores the importance of distinguishing between admissibility and quality. The second test (Sec. 5.4.2) illustrates an admissible case where one of the criteria is not met. Unlike the previous case, this intention doesn't present a safety risk but rather an inconvenience, making it a valid option. The quality, particularly in terms of lateral acceleration, highlights the inconvenience associated with this intention. The last test (Sec. 5.4.3) demonstrates that the formulation can be used to identify a superior intention based on the defined criteria. In this case, the intention is free of risks and meets all other criteria, resulting in a higher quality score (0.801).

With this quantification, we can rank these different intentions and select the best one that is both feasible and of higher quality according to the criteria we have defined. Based on Tab. 5.1, the three intentions can be ranked as follows: 'intention 3' is the best, followed by 'intention 2,' and 'intention 1' is considered the least favorable due to its lack of admissibility. It's important to note that the approach involves defining analyzers that rely on knowledge. The definition of analyzers is not trivial and may require a set of hyperparameters, which adds complexity to the implementation of this method.

5.6 Conclusion

In the approach, we have established a generic formulation assigning a score to an intention, reflecting both its feasibility (defined by admissibility) and its quality. The implementation and case study demonstrate that the formulation enables us to perform this quantification task and rank multiple intentions for the same scenario effectively. Furthermore, the implementation underscores the importance of distinguishing between admissibility and quality.

Currently, the solution has been applied in a static environment to facilitate the prediction of the next state. In future approaches, we intend to perform state esti-

mation in dynamic environments using prediction models [92] for the surrounding vehicles.

6 Intention Fusion

***Abstract**: The presented approach aims to merge the intentions of humans and the autonomous system. This fusion leverages the assessment of each intention to guide the fusion towards the highest-quality intention while ensuring the admissibility of the proposed intentions. The collaboration between humans and the autonomous system motivates us to employ game theory within a non-cooperative game. The resolution of our problem can be achieved through a Nash equilibrium. Furthermore, in this approach, the current authority of the human over control is considered to avoid abrupt changes.*

Contents

6.1 Motivation

In the presented approach, the primary goal is to fuse the intentions from both human and autonomous systems. For each entity, the intention predicted (chapters 3 and 4) is expressed as a sequence of linear and angular velocities, as a reminder:

$$I_k = \{(v_t, w_t), ..., (v_{t+T}, w_{t+T})\} \tag{6.1}$$

Where T is the intention sequence length, $k \in \{human, auto\}$. The proposed solution doesn't make a selection between the intention of the human and the autonomous system but rather achieves a fusion that includes both intentions. This fusion is based on the evaluation (chapter 5) of these intentions to guide the fusion towards a higher quality intention. Furthermore, the fusion process employs fusion admissibilities to ensure the admissibility of each intention. If an intention is inadmissible, a complete authority transfer is executed to one of the entities. The final fusion reflects the authority of the human over control, the solution includes the current authority. In this approach, humans and autonomous systems find themselves in competition. Each of them wants their intention to be applied to the vehicle. This human-machine competitiveness is evident in shared control in robotics. Moreover, some researches [31, 44] illustrates this competitiveness using game theory, particularly non-cooperative game theory, highlighting this rivalry between humans and machines. By applying these concepts, a solution is found, ensuring that this solution best satisfies each player. Some literature has already approached this with shared navigation [44, 100, 101]. Furthermore, in the fusion process, we want to include the concept of authority variable. The reason for this choice is to avoid a continual change of authority between the human and the autonomous system, thus avoiding any inconvenience to the driver [102]. In this way, the sudden transfer of authority is avoided, excepted an urgency need to change the authority suddenly. This authority variable (λ) informs about the impact of the human and the system autonome authority $(1 - \lambda)$ on the final control [27]:

$$u_{final} = \lambda \cdot u_{human} + (1 - \lambda) \cdot u_{auto} \tag{6.2}$$

Where u_{final} is the control applied to the system, u_{human} is the control by the human, u_{auto} is the control by the autonomous system and λ represents the authority variable. The following figure 6.1 shows the pipeline of our approach.

6.2 Prerequisites

This section defines the essential concepts and approaches for defining the game and its resolution.

6.2.1 Profile Projection

For each entity, a projection is made onto the linear velocity and angular velocity sequences. This projection aims to condense sequences into a single point while

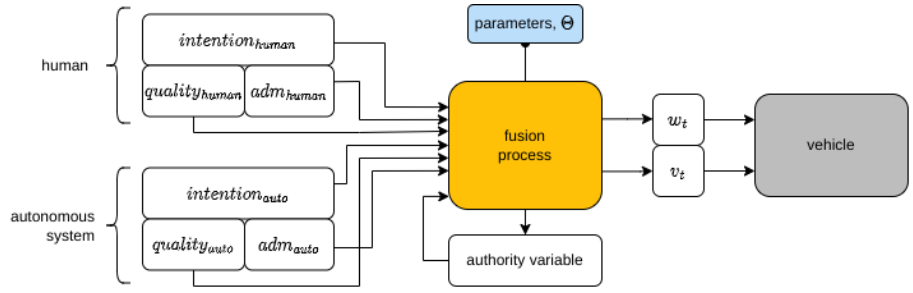


Figure 6.1: Fusion pipeline, from entities intentions and their evaluation to the fusion command.

preserving the original information defined by these sequences. To achieve this, in the initial step, we perform a regression on each sequence for each entity.

$$V_k = \{v_t, v_{t+1}, ..., v_{t+T}\} \approx r_k^v(i) \tag{6.3}$$

$$W_k = \{w_t, w_{t+1}, ..., w_{t+T}\} \approx r_k^w(i) \tag{6.4}$$

Where $k \in \{human, auto\}$, V represents the linear velocity sequence, and W represents the angular velocity sequence, r_k^v is the regression on linear velocity and r_k^w is the regression on angular velocity. Initially, we perform regression on the desired sequence. The following explanation focuses on the linear velocity sequence V_k, and the method applies similarly to the angular velocity sequence.

$$r_k^v(x) = \sum_{i=0}^{n} a_i^k \cdot x^i \tag{6.5}$$

$$r_k^v(j) \approx v_{t+j} \tag{6.6}$$

Where n is the degree of regression, a_i is the regression coefficient of degree i. Each coefficient a_i^k is normalized as c_i^k, such that $c_i \in [-1, 1]$. In this way, the linear velocity profile V_k is defined as:

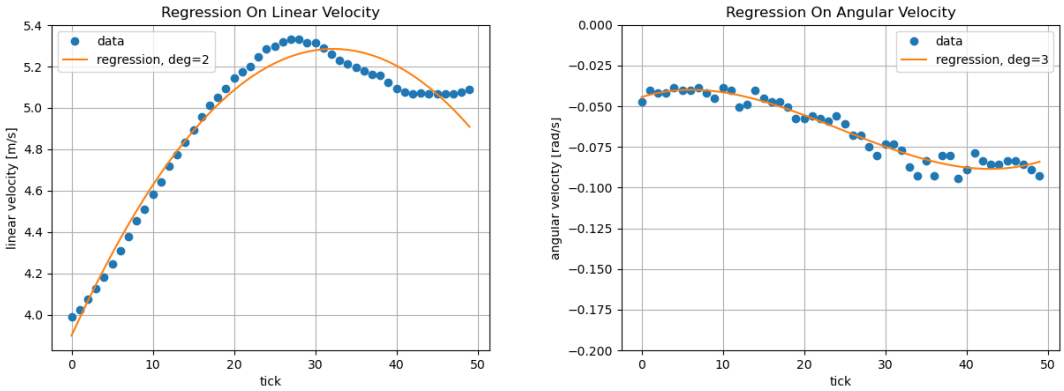
$$P_k^v = (c_0^k, ..., c_n^k) \tag{6.7}$$

Analogously:

$$P_k^w = (c_0^k, ..., c_n^k) \tag{6.8}$$

Each coefficient is normalized because the distance of the profiles is used (Sec. 6.2.2). Without this normalization, the distance would be too influenced by certain coefficients, especially for low-degree coefficients. This normalization is based on a statistical study to assess the variations per coefficient (Appendix C).

This way, for each entity, the intention is defined by 2 profiles (P_k^v, P_k^w). These two profiles collectively encompass the information of the entity's intention. Figures 6.2 represents regressions on a real-world recording. Figure 6.3 depicts an example of linear velocity profiles. Two different profiles are generated by exploiting the regression defined in Figure 6.2a with added noise. Figure 6.3a displays two velocity intentions, and the corresponding profiles are shown in Figure 6.3b.



(a) Velocity linear regression.

(b) Velocity angular regression.

Figure 6.2: Velocity regressions on real data.

6.2.2 Authority Variable

The authority variable ensures that decision-making is influenced by the current authority. This approach helps to regulate abrupt changes at each iteration. However, if there is a significant difference in quality between the players, sudden changes can occur. At the initial state, the authority variable is centered on the human, i.e., $\lambda = 1$. Subsequently, this variable is updated based on the distance between the profiles. The distance between two profiles is defined as follows:

$$D(P_k, P_l) = \sqrt{\sum_{i=0}^{n} (c_i^k - c_i^l)^2} \tag{6.9}$$

The distance of the fusion profile from the human and autonomous system profiles reflects the authority λ_{fusion} of the human over the vehicle:

$$\lambda_{fusion} = 1 - \frac{D(P_{human}, P_{fusion})}{\epsilon + D(P_{human}, P_{auto})} \tag{6.10}$$

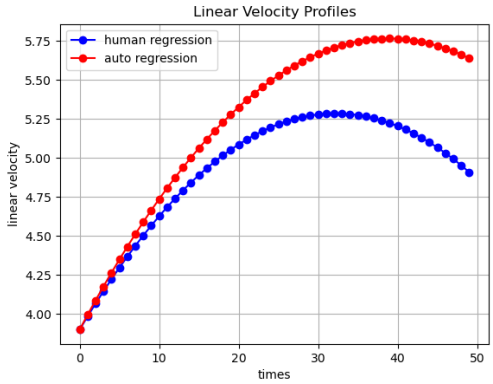
In order to avoid a divide by zero, the small quantity ϵ is added to the denominator.

6.2.3 Similarity

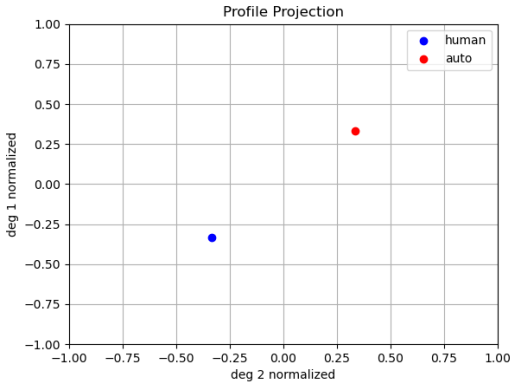
In the described approach, the system is centered on the human, meaning that if the intentions are too different, the fusion system prioritizes human driving. This intention difference is reflected in by the distance between the profile points. Thus, the formulation of similarity is defined as follows:

$$similarity(P_k, P_l) = \frac{1}{1 + D(P_k, P_l)} \tag{6.11}$$

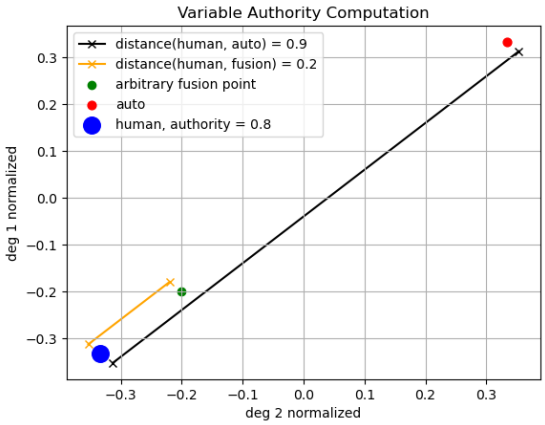
The figures 6.3e,6.3f show the similarity between human and auto profile. These figures illustrate the similarity of the profiles in relation to the human profile (blue



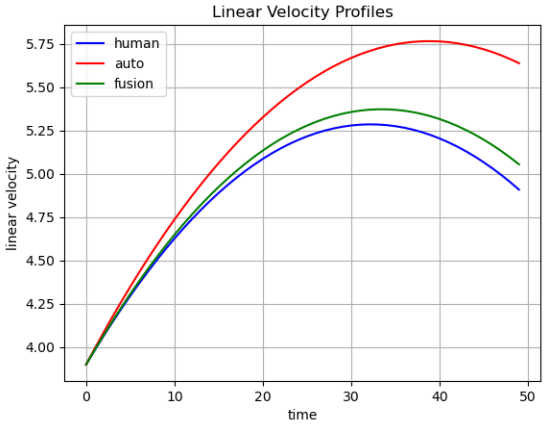
(a) Intentions examples.



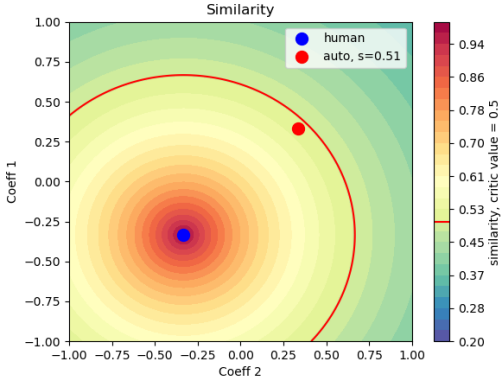
(b) Profile projection examples



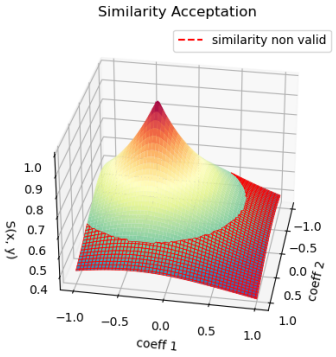
(c) Authority computation based on profile positions.



(d) Profiles based on left profiles positions, showing human authority.



(e) Similarity between human and auto profiles.



(f) Similarity between human and auto profiles.

Figure 6.3: Example support, from the profile regressions to an arbitrary fusion. Including metrics and concepts representations.

Admissibility		λ	Authority
human	auto		Interpretation
any	any	1	human

(a) *similarity* $<$ *Sim*

Admissibility		λ	Authority
human	auto		Interpretation
yes	yes	λ_{t-1}	both
no	yes	0	auto
yes	no	1	human
no	no	1	human

(b) *similarity* $\geq$ *Sim*

Figure 6.4: Authority adaptation, Sim is the similarity acceptability variable.

dot). The closer a profile is to the human profile, the more similar it is. The figure 6.3f illustrates the similar profiles in 3D. This concept of similarity defines the resemblance between the two profiles, thus enabling us to evaluate if the choices of each entity are close or not. Thus, by defining a parameter Sim as an acceptable similarity threshold, it is possible to evaluate whether a profile is acceptable if the similarity is greater than or equal to Sim. On figure 6.3e, the red circle represents the variable Sim, the value beyond which similarity is no longer acceptable. Each entity has two profiles, one for linear velocity and one for angular velocity. Similarity must be assessed for each of these profiles, and an intention is considered similar if and only if both profiles exhibit acceptable similarity.

Indeed, two intentions are considered similar if, and only if, all the profiles are similar. If one profile is different (v or w), than the overall intention is different. For example, if two intentions aim to navigate around an obstacle on the left and right sides of the obstacle, the linear velocities are close, but the angular velocities are different. The result of such a fusion could lead to a command guiding the vehicle into the obstacle, potentially causing a collision.

6.2.4 Deal

As explained in the introduction, the system utilizes the authority variable to achieve the fusion of intentions. This authority variable, initially calculated based on profile distances, can also be used to grant control of the vehicle to one of the entities. So, if the authority variable $\lambda = 0$, then the authority of the human is null, indicating that the vehicle follows the intention of the autonomous system. Conversely, if $\lambda = 1$, the fusion prioritizes the human's intention.

This orientation of the authority variable depends on the similarity of the profiles. If the profiles are too dissimilar (less than Sim, the acceptable similarity), authority is granted to the human. Additionally, the admissibility values assigned through intention evaluation influence the choice of authority in cases where the profiles are similar (greater than Sim). Tables 6.4 summarize the choice of the authority variable based on similarity and the admissibility of each intention. The variable λ_{t-1} defines the authority variable computed from the previous fusion.

6.3 Game Theory

6.3.1 Game Definition

We define the human and the autonomous system as two competing players, each of them desiring the fusion to be most similar to their intention. Thus, we define a player's strategy as $S_k = (s_1, s_2, ..., s_N)$, specifying the position of the fusion point they desire. Each player defines a loss function, considering the position of their profile, the current authority variable, and the strategy of each player (S_{human}, S_{auto}).

Each profile is defined by a position in (N+1)-dimension space $(c_0^k, c_1^k, ..., c_n^k)$, which corresponds to the regression coefficients normalized. So human and auto system profiles are defined by:

$$P_{human} = (c_0^h, c_1^h, ..., c_n^h) \tag{6.12}$$

$$P_{auto} = (c_0^a, c_1^a, ..., c_n^a) \tag{6.13}$$

Based on these profiles, for each player, we define the following loss functions:

$$L_{human}(S_{human}, S_{auto}) = q_h \sum_{i=0}^{n} \cdot (c_i^h - (\lambda S_{human,i} + (1-\lambda) S_{auto,i}))^2 \tag{6.14}$$

$$L_{auto}(S_{human}, S_{auto}) = q_a \sum_{i=0}^{n} \cdot (c_i^a - (\lambda S_{human,i} + (1-\lambda) S_{auto,i}))^2 \tag{6.15}$$

Here, q_h and q_a are respectively the intention quality of the human and autonomous system. The players update their strategies simultaneously. We chose this approach over a sequential game because the resolution is the same, and the convergence time is faster in a simultaneous game. In the case of a sequential game, the resolution occurs sequentially, meaning that optimization only takes place along certain axes, while the other axes are updated in the subsequent iteration.

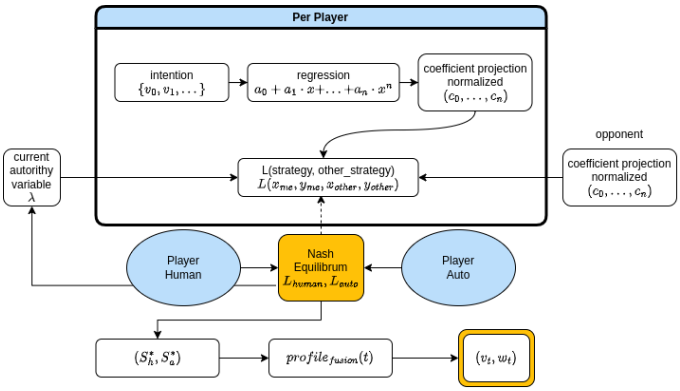
6.3.2 Nash Equilibrium

By definition [103], the Nash equilibrium point is established when a player cannot improve their strategy, considering the strategies of the other players. Thus, this point minimizes the loss functions of each player, best considering all the loss functions (L_k, Eq. 6.15). Applied to our game, the existence of the equilibrium point is proven if a player acknowledges that there exists an optimal strategy (S_k^*) such that no other strategy can be better. Thus, we translate the problem as follows: for each player, we have:

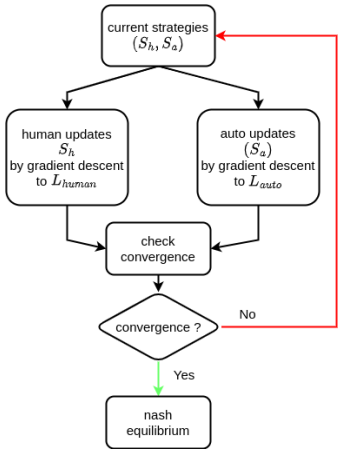
$$L_k(S_k^*, S_l^*) \leq L_k(S_k, S_l^*), \forall S_k \in \mathcal{R}^n \tag{6.16}$$

Where k is the player playing, and l is the opponent player. The elements of the sum of loss functions,L_k (Eq. 6.15), can be written as follows:

$$f(x, y) = (c - (\alpha x + (1-\alpha) y))^2 \tag{6.17}$$



(a) Fusion processes in applying Nash equilibrium.



(b) Nash's resolution diagram.

Figure 6.5: Resolution precesses.

Where $c \in \mathcal{R}$ the coefficient normalized, $\alpha = \lambda$ if the player is the humain and $\alpha = 1 - \lambda$ if the player is the autonome system, x the player strategy and y the opponent player strategy. In the case where y is fixed, the convexity of the function (f) depends only on whether $\alpha > 0$.

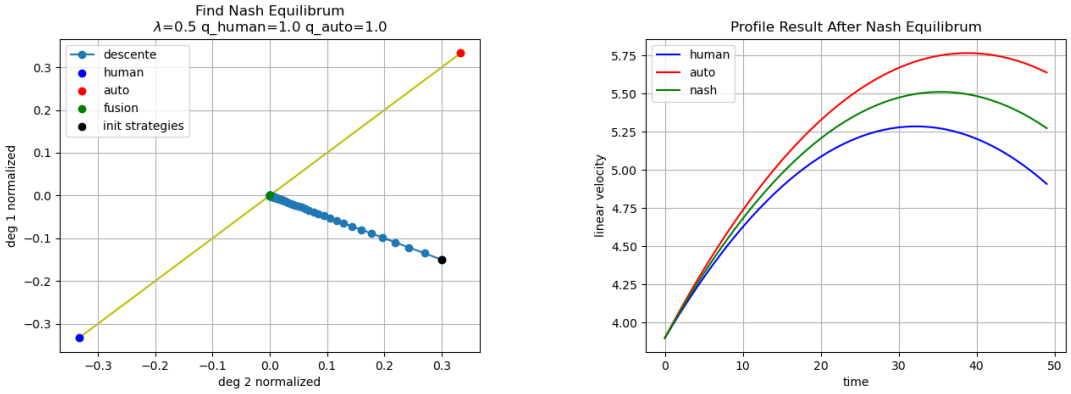
In our study $\alpha \in [0, 1]$. The case where $\alpha = 0$ is trivial because the loss function would then be constant, then the inequality (Eq. 6.16) is proven. In the case of convexity ($\alpha \in]0, 1]$), then the loss function (L_k) is also convex because it is a sum of convex functions, and according to the properties of convexity, the sum is also convex. Knowing this convexity, we assert the inequality to be proven. In other words, the existence of the Nash equilibrium is established.

6.3.3 Resolution

Solving the game means finding the fusion point. This solution is found after discovering the Nash equilibrium point. Figure 6.5a illustrates the resolution process. To find this point, both players play simultaneously in each turn until the solution reaches a convergence point; the players continue playing. Figure 6.5b depicts the resolution diagram. The following figures 6.6 shows the profile fusion in applying Nash equilibrium.

6.3.4 Game Lock

The authority value (λ) can take an extreme value of either 0 or 1. In this scenario, the authority value cannot change. In fact, even if the quality of the human changes (Fig. 6.7a) or the profile distance between human and auto changes (Fig. 6.7b), the authority doesn't change. This occurs because the automated player cannot influence the fusion strategy due to its excessively low assigned authority. Consequently, the authority remains unchanged from one step to the next. It should



(a) Nash's equilibrium position.

(b) Fusion profile from Nash equilibrium.

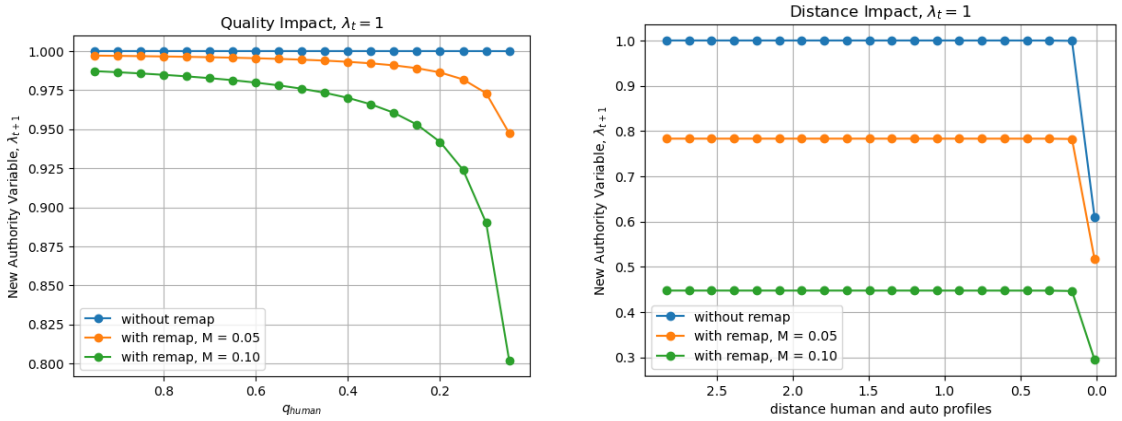
Figure 6.6: Nash's equilibrium result illustration.

be noted that even if the automated system closely aligns with the human's intention, the authority does not change, as depicted in Fig. 6.7b. In Fig. 6.7a, the human and auto profiles are fixed, the quality of the autonomous system's intention is set to 1.0 (representing the best possible case), and the quality of the human varies between 1 and 0. In Fig. 6.7b, the quality of the autonomous system's intention is 1.0, the human's quality is 0.01 (representing a case where the autonomous profile is much better than that of the human), and the distance between the human and autonomous profiles approaches zero.

To avoid this scenario, we implement a remapping of the authority calculated after equilibrium, thereby reintroducing the authority into the equation. This way, the authority retains its fundamental influence, but it can be discussed in the next turn. The remapping is defined as:

$$\lambda_{game}(t) = \lambda_{fusion}(t)(1 - 2 \cdot M) + M \tag{6.18}$$

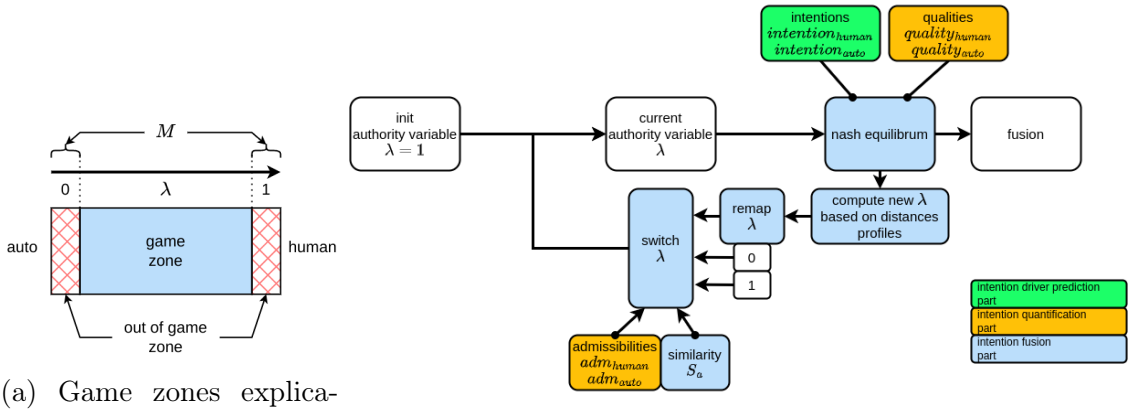
Where M represents the out-of-game zone where there is a risk that the position remains unchanged. The choice of M depends on the desired location for the inflection point. Figure 6.8a shows the game zones. Figure 6.8b represents the diagram of the authority variable, including the remapping. This figure represents the impact of other parts developed in other chapters.



(a) Human intention quality change.

(b) Distance human/auto change.

Figure 6.7: Authority variable (λ_{fusion}) variation in case where $\lambda = 1$.



(a) Game zones explications.

(b) Authority variable pipeline.

Figure 6.8: Including authority remapping solution.

Figure 6.7a,6.7b involve the same test but with remapping onto the authority variable's domain. Thus, we examine the impact of remapping on the calculation of the new authority value. Whether it is remapping with $M = 0.05$ or $M = 0.1$, it brings back into play the authority variable, and in the case of better system autonomy quality, the value will be impacted again. The value of M influences the reintroduction, in other words, the need to have a greater or lesser gap in the quality of players for the authority variable to evolve. As shown in Figure 6.7a, the inflection point is increasingly closer if M is small.

6.4 Validation

To apply the methodology, we use the recordings made in Chapter 5. The figure 6.9 represents the projection of these intentions into trajectories. The intentions 0,1 and 3 are admissible with different qualities, and the intention 2 is not admis-

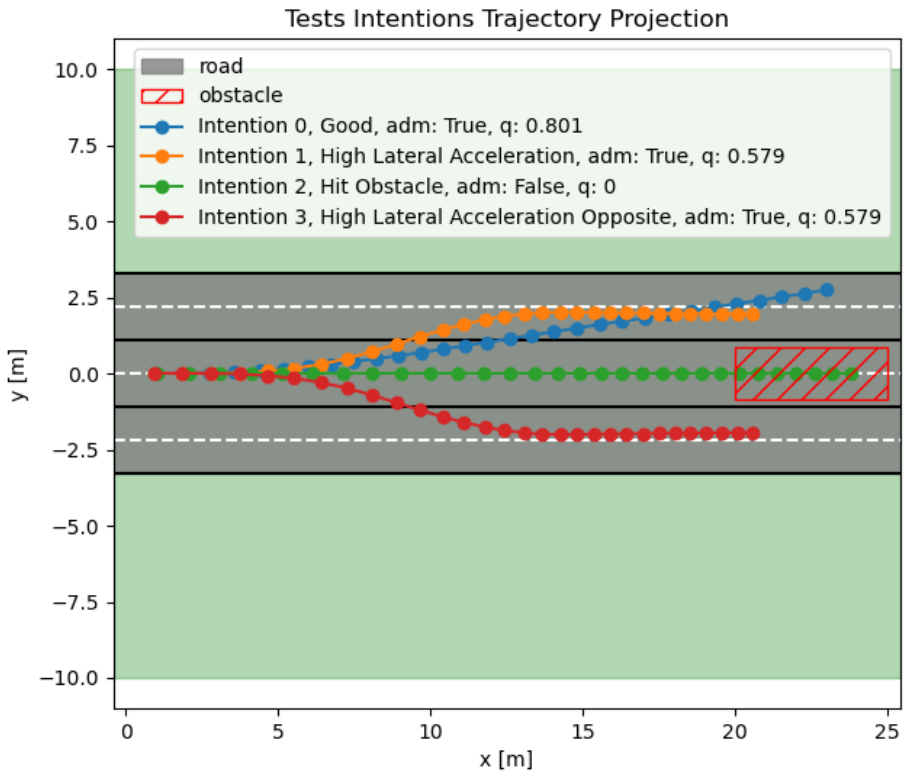


Figure 6.9: Intention velocities support for the validation, projected to trajectory.

sible. The intention 1 and 3 are considered as not similar. Thus, validation is performed through three tests, as described in Table 6.1.

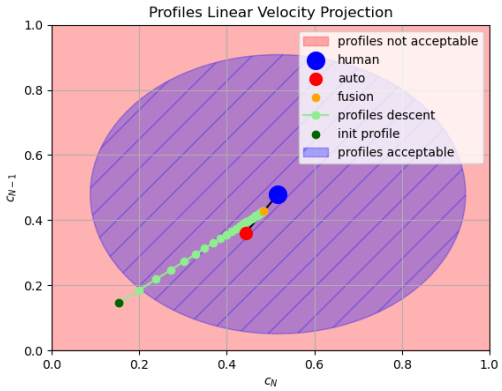
In these tests, we have defined the initial state, such that the authority (λ) is initialized to 0.5, which means that the human and the autonomous system have equal authority over the vehicle. Furthermore, the similarity parameters are set to $Sim = 0.7$ for the linear velocity profile and $Sim = 0.7$ for the angular velocity profile.

Test	Goal	Intention Human		Intention Auto		Similar
		Admissible	Number	Admissible	Number	
01	Intentions both admissible	Yes	0	Yes	1	Yes
02	One intention isn't admissble	Yes	0	No	2	Yes
03	Intentions are not similar	Yes	1	Yes	3	No

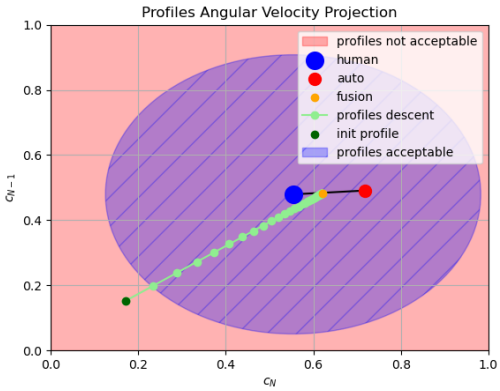
Table 6.1: Tests intention definitions, the intention number refer to the intention in the figure 6.9.

6.4.1 Test Admissible

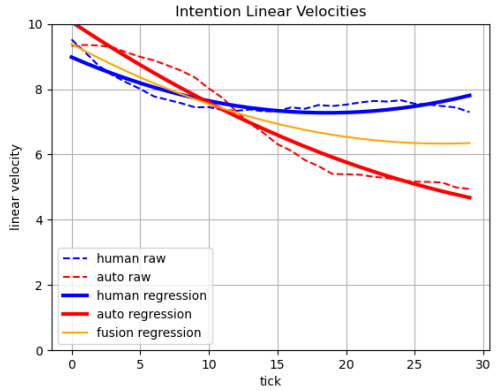
In this first test, the intentions of the human (intention 0) and the autonomous system (intention 1) are admissible but have different qualities, 0.971 for the human intention and 0.744 for the autonomous system intention. This difference in quality is due to a lateral acceleration that is considered too high. For linear velocity, the regression is of order 2, and for angular velocity, the regression is of order 3. The use of a higher-degree regression for angular velocity is justified due to more significant variations compared to linear velocities. Figures 6.10c and 6.10d illustrate the regressions performed on linear and angular velocities. These figures also display the regression of the fusion profile once the Nash equilibrium is found. Figures 6.10a and 6.10b illustrate the positions of the profiles. It is important to note that this projection uses only 2 coefficients for visual representation to position the fusion profile. Due to equal initial authority, the fusion point without an update should be located at the midway point of the candidates. However, in this case, the quality of the human is more important than the autonomous system, affecting the position of this point, which is closer to the human. The trajectory projection of the fusion profile is displayed in Figure 6.10e, demonstrating that the fusion profile is more influenced by the human profile. Additionally, Figure 6.10f shows the evolution of the authority variable. As the fusion profile approaches the human, once the fusion is completed, the resulting authority is higher.



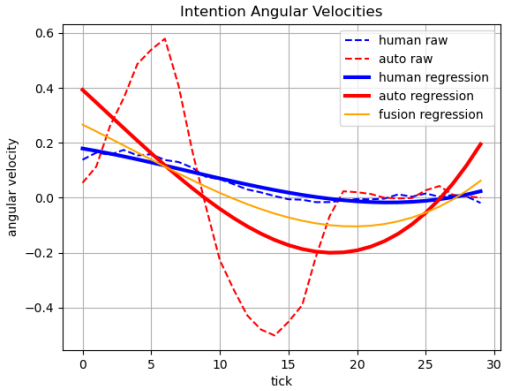
(a) Linear profile plot (only degree N and degree N-1 normalized).



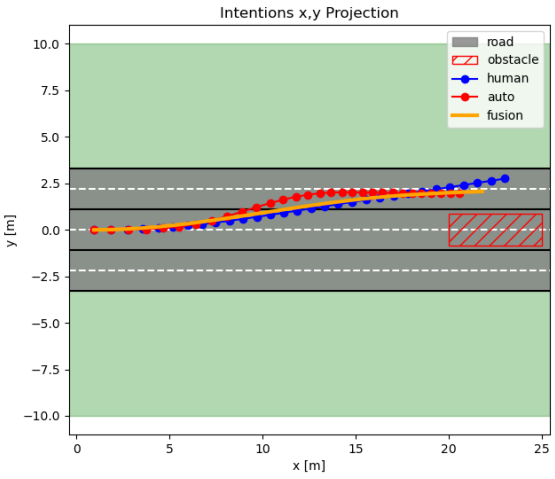
(b) Angular profile plot (only degree N and degree N-1 normalized).



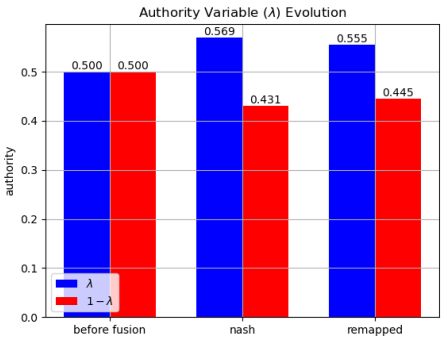
(c) Linear regressions.



(d) Angular regressions.



(e) Profiles trajectory projection.

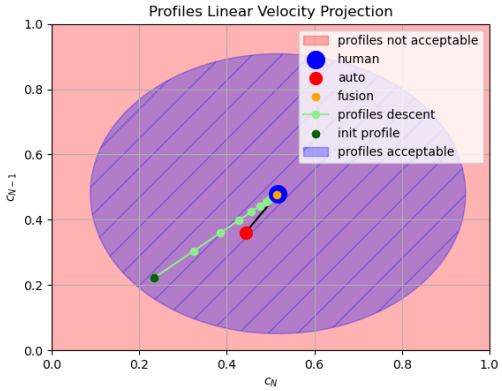


(f) Authority evolution.

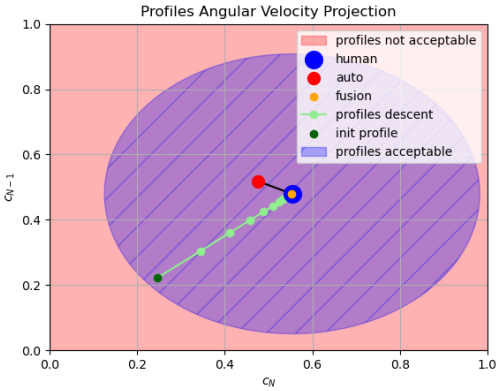
Figure 6.10: Test 01: Results after fusion.

6.4.2 Test Not Admissible

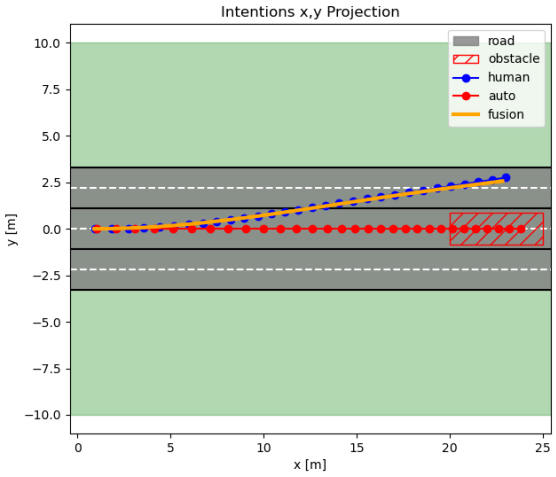
In this second test, this time the intention of the human is admissible, but not the intention of the autonomous system. Thus, in this case, the fusion system enforces authority on the human, meaning that $\lambda = 1$. This authority is reflected in the fusion, as shown in Figures 6.11a and 6.11b, where the fusion profile is aligned with the human profile, indicating that the fusion will apply the human's intention. The trajectory projection of the fusion (Fig. 6.11c) overlays the human trajectory. The authority (Fig. 6.11d) after fusion is, therefore, $\lambda = 1$, which demonstrates the importance of remapping to avoid continuous authority being locked on the human.



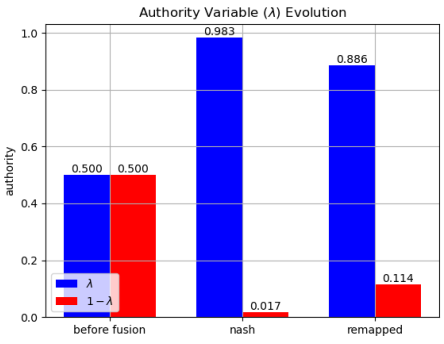
(a) Linear profile plot (only degree N and degree N-1 normalized).



(b) Angular profile plot (only degree N and degree N-1 normalized).



(c) Profiles trajectory projection.

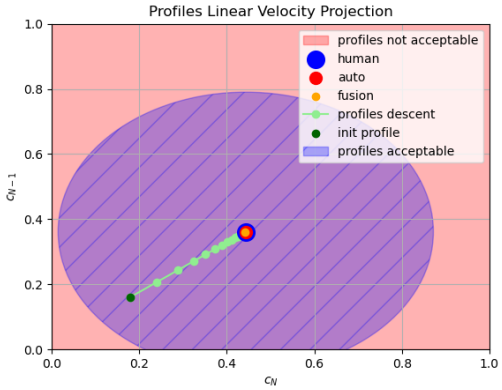


(d) Authority evolution.

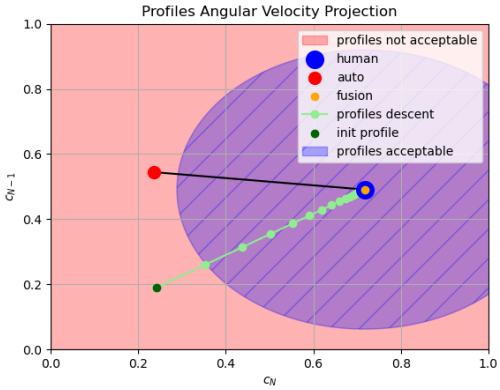
Figure 6.11: Test 02: Results after fusion.

6.4.3 Test Not Similar

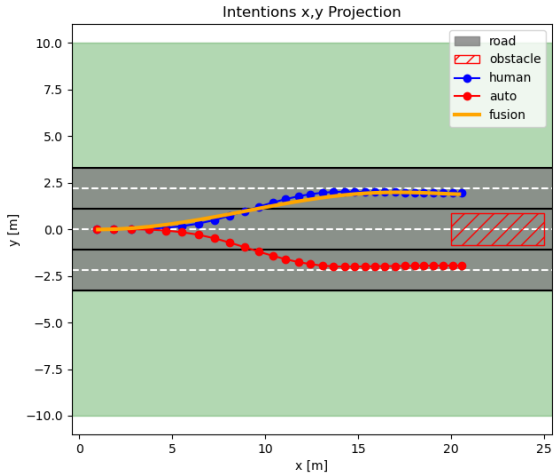
This test examines the fusion behavior in the case where the intentions are not similar. The human intention is defined by the intention 1 and autonomous system is defined by the intention 3. In this example, each entity chooses to avoid the obstacle on opposite sides, resulting in opposing angular velocities. As shown in Figures 6.12a and 6.12b, the linear velocity profiles are similar, but the angular velocities are not. Hence, the intentions are considered dissimilar, and authority is given to the human. As depicted in Figure 6.12c, the projection of the fusion intention follows the same trajectory as the human intention due to the dissimilarity of intentions.



(a) Linear profile plot (only degree N and degree N-1 normalized).



(b) Angular profile plot (only degree N and degree N-1 normalized).



(c) Profiles trajectory projection.

Figure 6.12: Test 03: Results after fusion.

6.5 Discussion

The tests conducted in section 6.4 assess the effectiveness of fusion as well as the adaptability of fusion based on intentions and their evaluation.

The first test demonstrates that the quality of intentions can influence the resulting fusion, while also taking the current authority into account. Fusion is thus directed towards the entity with the highest quality, all while considering the previous authority to avoid divergence.

The second test shows a different perspective. It illustrates the adaptability of the system when faced with a situation where one of the two intentions is inadmissible. The results show that this admissibility doesn't disrupt the resolution of the game. In fact, it directs the final merge towards the entity whose intention is admissible, proving that the system can adapt and make sound decisions even in non-ideal scenarios.

The third test introduces a scenario where the intentions involved are dissimilar. Despite this disparity. The system aligns itself with human behavior. This highlights the capability of the system to handle this case.

6.6 Conclusion

This approach achieves the fusion of two intentions by using game theory and incorporating the authority variable. The methodology has been validated on simulated data. The tests demonstrate the behavior of the fusion and the ability to adapt based on the qualities and admissibility of intentions, as well as the similarity of intentions. In this way, we are able to achieve a fusion of intention, representing both the choices of the human and the autonomous system. This fusion will subsequently be used as a reference intention by the lower-level module. This module will be able to use this reference intention to apply it to the car. Or a lower-level sharing (torque level) can take place between the human and the autonomous system, exploiting this intention as a reference. Beyond these developments, we plan to include the intentions of other vehicles around our subject vehicle. This can be achieved through vehicle-to-vehicle communication or by predicting the behavior of other agents. As highlighted in our literature review, this approach is already addressed, with predictions aiming to prevent collisions. We view the intentions of other users as a fusion constraint, not for a common goal, but for individual goals considering other agents. Our idea is to limit fusion to the scale of the car, including the intentions of other users, using concepts like velocity obstacles to limit vehicle speeds, avoiding collisions, and maintaining safe interaction with other road users. We could extend this concept to our intention fusion, adjusting the gradient descent to respect this new constraint, expressed either as a gradient descent limitation or as a penalty in the loss function, similar to neural networks where penalties keep model weights near zero and prevent divergence or excessive influence of certain parameters. Experimental tests are also underway on the vehicles of the Heudiasyc laboratory.

7 Conclusion

***Abstract**: This chapter concludes the thesis and defines future perspectives.*

Contents

7.1 Conclusion

In this thesis, in the context of navigation sharing between the autonomous system and the human, we have presented all the elements required for performing intention fusion at the vehicle level. From predicting these intentions to the fusion process, relying on quantifying these intentions.

Our work has primarily focused on enhancing lane detection, obstacle avoidance, intention prediction, and intention fusion, utilizing concepts from visual servoing, deep learning networks, and game theory.

We began the exploration by developing an autonomous system capable of driving within a lane while avoiding obstacles, using combined lane detection and lidar. This was further optimized by proposing an advanced method for candidate selection. The robustness of the controller was significantly improved by integrating a deep learning network for lane detection, and safety was prioritized by introducing a LiDAR filter layer that limits the system's actions.

These methods were tested using an indoor mobile robot, a virtual vehicle in the SCANeR Studio driving simulator and a real car vehicle. These experiments demonstrated the adaptability of our controller in various scenarios. However, we recognized the challenge of fine-tuning the controller due to its dependence on a set of hyperparameters, suggesting that future work could explore the dynamic adjustment of parameters based on situational changes.

Another key aspect of our work was the development of a prediction model capable of foreseeing short-term human intentions. This model addresses the complexity of predicting human behavior, especially in complex situations, and lays the groundwork for future advances in predicting less predictable and more dangerous behaviors.

We also established a generic formulation for scoring intentions, effectively quantifying and ranking multiple intentions in given scenarios. This formulation distinguished between the admissibility and quality of intentions, highlighting the need to consider both factors in decision-making processes.

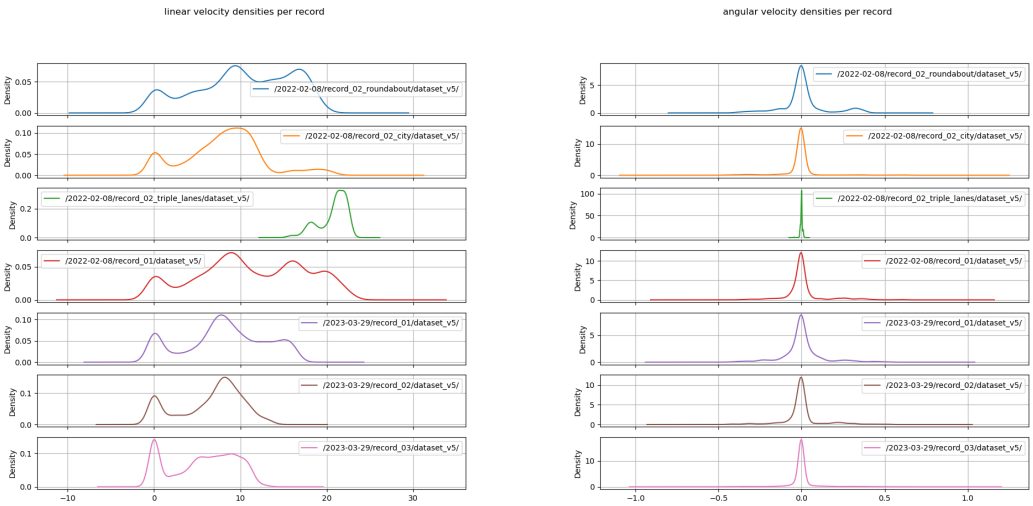
Additionally, our approach merged two intentions using game theory and included an authority variable. This methodology was validated on simulated data, showing promising adaptability and efficiency in considering the qualities, admissibility, and similarity of intentions. Furthermore, this method was proposed with the goal of being limited by the external intentions of other road users, such that these restrictions would limit the gradient descent by taking into account the intentions of these other users.

7.2 Perspective

- **Methodology Expansion:**
  - Extend our methods to dynamic environments;
  - Improve intention prediction by integrating intentions from others vehicles;
  - Perform experimental tests on vehicles from the Heudiasyc laboratory;
- **Experimental Implementation and Validation:**
  - Implement the concepts mentioned in this thesis on a real vehicle;
  - Verify the pipeline's functionality and assess execution times;
- **Control Sharing and Integration:**
  - Finalize control sharing, specifically how the intention fusion will be used by the controller;
  - Determine how the controller should exploit this reference intention to calculate vehicle commands;
- **Introduction of Multi-Agent Perspective:**
  - Introduce the multi-agent notion into our intention fusion, the intention fusion method was developed to facilitate the multi-agent aspect;
  - Evaluate our method from two perspectives:
    - Nearby vehicles may constrain intention fusion;
    - Integrating other vehicles as additional players in game theory;
- **Publication Plan:**
  - In parallel with these efforts, plan to submit an article to an international scientific journal, including the work from this thesis;

A Dataset Human Driving

Figures A.1a,A.1b show the distributions of speeds from the various tests in the dataset used for training our prediction model. Figure A.2 displays the various positions from the different tests in the test dataset, thereby illustrating the diversity of our tests to ensure a distinct evaluation based on the situations.



(a) Linear velocity. (b) Angular velocity.

Figure A.1: Velocity (linear and angular) density per recording.

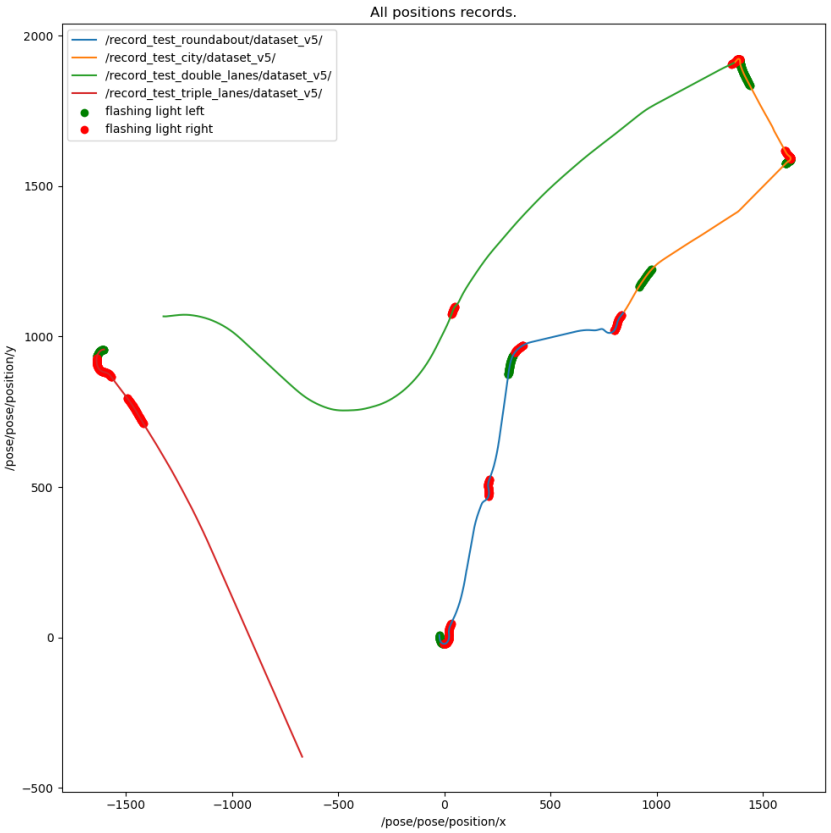


Figure A.2: Datasets tests positions.

B Evaluation Intention

Additionals graphics on the evaluation of the intentions of chapter 5. Metric graphic represents the value in applying metric function. The analyzer function graphic represents the curve of the analyzer function applied on metric value and score graphic represents the metric value applied to the analyzer function.

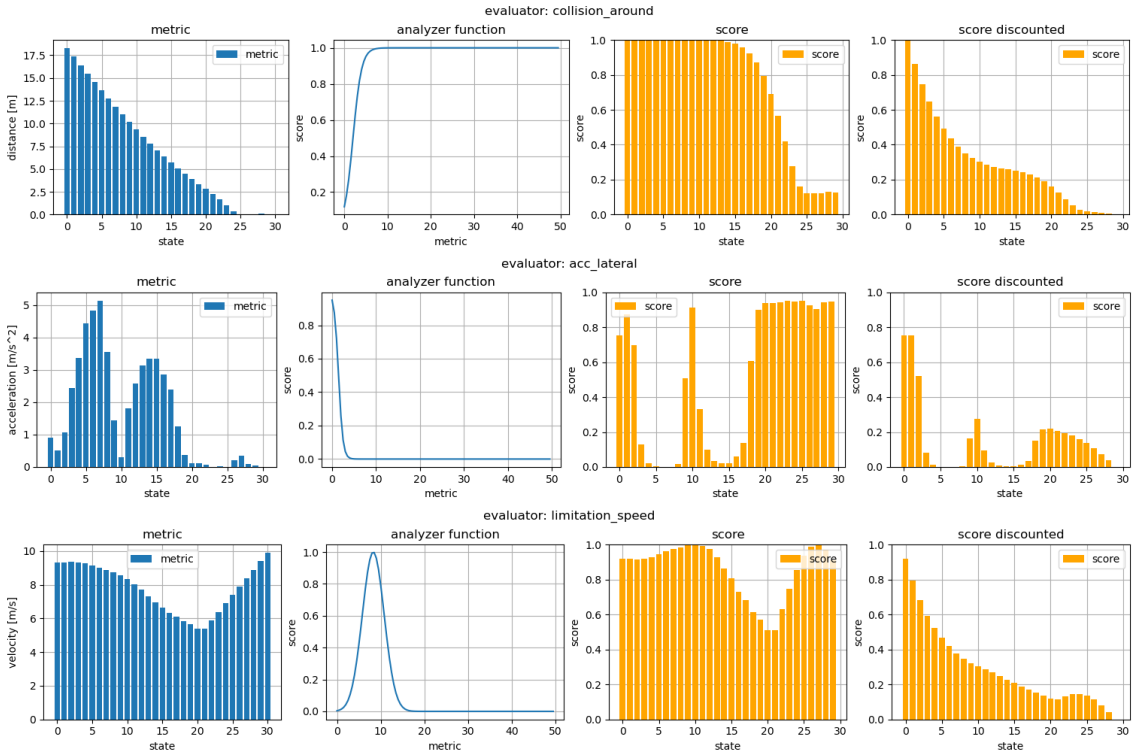


Figure B.1: Evaluators on intention 02.

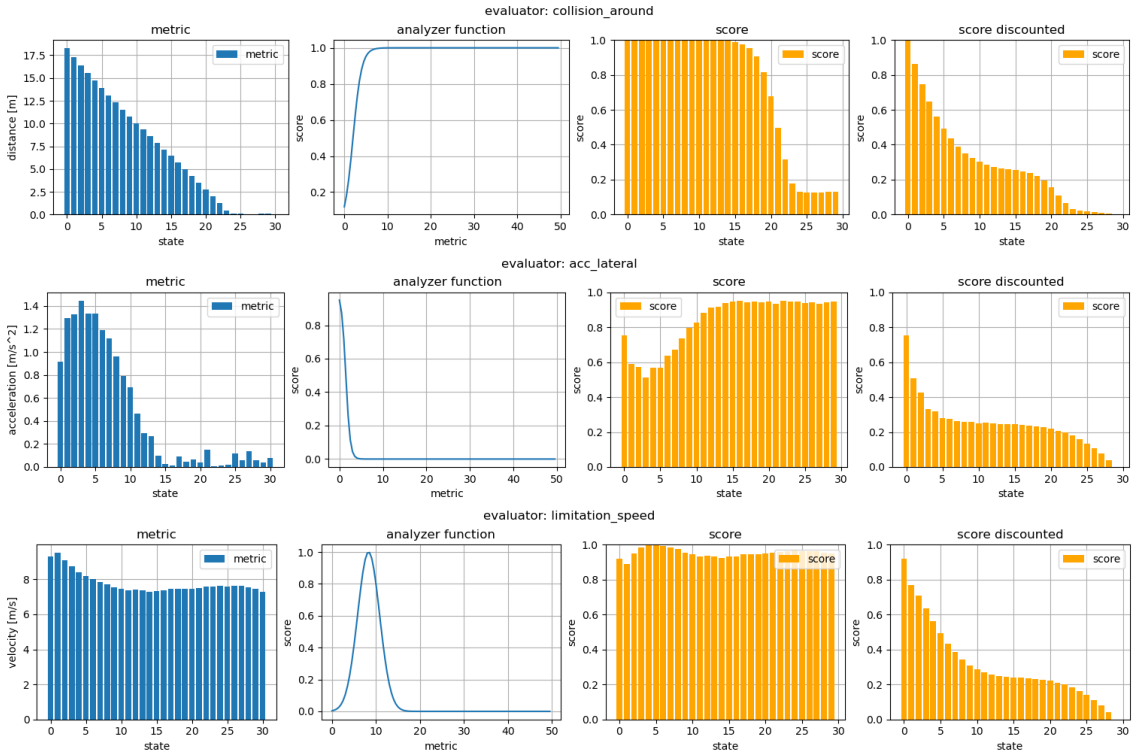
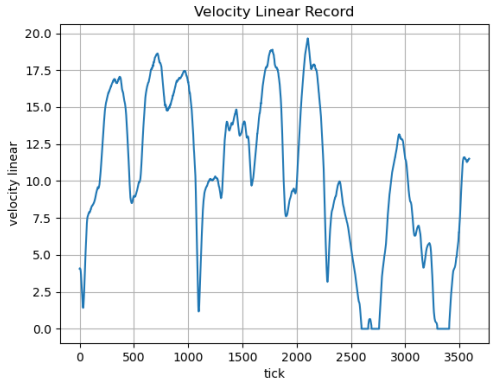


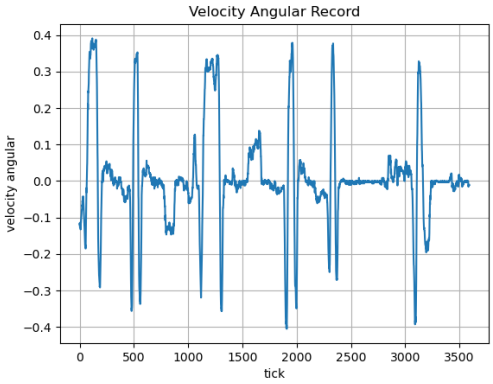
Figure B.2: Evaluators on intention 03.

C Regression Coefficients Distribution

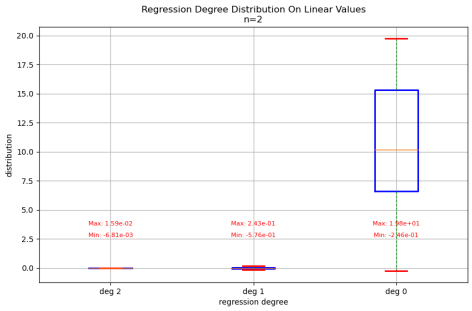
In this study, we recorded the linear and angular velocities of the car (Fig. C.1a,C.1b). This recording comprises 3500 data points, sampled at 10 Hz. For the entire dataset, we performed regression on sequences of size 30 (i.e., 3 seconds). From these regressions, we obtained the following distributions, as shown in Fig. C.1c and C.1d. The boxes represent the distribution by coefficient, excluding outliers, and for each box, the minimum and maximum values are indicated.



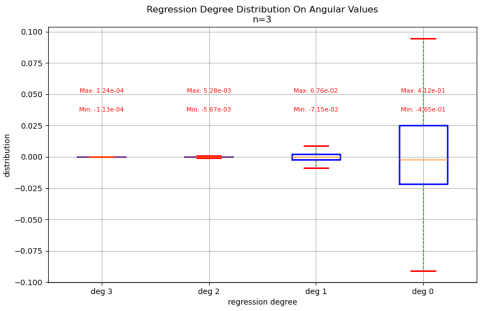
(a) Velocities linear from recording.



(b) Velocities angular from recording.



(c) Coefficients regression distribution on linear velocities sequences.



(d) Coefficients regression distribution on angular velocities sequences.

Bibliography

[1] Mauricio Marcano, Sergio DIaz, Joshue Perez, and Eloy Irigoyen. A Review of Shared Control for Automated Vehicles: Theory and Applications. *IEEE Transactions on Human-Machine Systems*, 50(6):475–491, 2020.

[2] C. Sentouh, P. Chevrel, F. Mars, and F. Claveau. A sensorimotor driver model for steering control. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 2462–2467, San Antonio, TX, USA, October 2009. IEEE.

[3] Franck Mars, Louay Saleh, Philippe Chevrel, Fabien Claveau, and Jean-François Lafay. Modeling the visual and motor control of steering with an eye to shared-control automation. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 55, pages 1422–1426. SAGE Publications Sage CA: Los Angeles, CA, 2011.

[4] Michael Flad, Clemens Trautmann, Gunter Diehm, and Sören Hohmann. Individual driver modeling via optimal selection of steering primitives. *IFAC Proceedings Volumes*, 47(3):6276–6282, 2014.

[5] Mingjun Li, Haotian Cao, Xiaolin Song, and Yanjun Huang. Shared control driver assistance system based on driving intention and situation assessment. *IEEE Transactions on Industrial Informatics*, 14:4982–4994, 08 2018.

[6] Amir Benloucif, Anh-Tu Nguyen, Chouki Sentouh, and Jean-Christophe Popieul. Cooperative trajectory planning for haptic shared control between driver and automation in highway driving. *IEEE Transactions on Industrial Electronics*, 66:9846–9857, 12 2019.

[7] Takahiro Wada, Kohei Sonoda, Takuya Okasaka, and Takahiro Saito. Authority transfer method from automated to manual driving via haptic shared control. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002659–002664. IEEE, 2016.

[8] Weibin Yang, Bin Fang, and Yuan Yan Tang. Fast and accurate vanishing point detection and its application in inverse perspective mapping of structured road. *IEEE Trannsactions on Systems, Man, and Cybernetics: Systems*, 48(5):755–766, 2016.

[9] Yue Kang. *Sensor-based navigation for robotic vehicles by interaction of human driver and embedded intelligent system.* PhD thesis, Compiègne, 2016.

[10] Danilo Alves de Lima and Alessandro Corrêa Victorino. A visual servoing approach for road lane following with obstacle avoidance. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 412–417. IEEE, 2014.

[11] Can Zhao, Li Li, Xin Pei, Zhiheng Li, Fei-Yue Wang, and Xiangbin Wu. A comparative study of state-of-the-art driving strategies for autonomous vehicles. *Accident Analysis & Prevention*, 150:105937, 2021.

[12] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. In *AAAI Conference on Artificial Intelligence (AAAI)*, February 2018.

[13] Ahmad Shour, Hugo Pousseur, Alessandro Correa Victorino, and Veronique Cherfaoui. Shared decision-making forward an autonomous navigation for intelligent vehicles. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1634–1640. IEEE, 2021.

[14] Federico Faruffini, Hugo Pousseur, Alessandro Corrêa Victorino, and Marie-Helene Abel. Context modelling applied to the intelligent vehicle navigation. In *IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2021.

[15] Emanuele Venzano, Hugo Pousseur, Alessandro Corrêa Victorino, and Pedro Castillo Garcia. Motion control for aerial and ground vehicle autonomous platooning. In *2022 IEEE 17th International Conference on Advanced Motion Control (AMC)*, pages 213–218. IEEE, 2022.

[16] Hugo Pousseur and Alessandro Correa Victorino. Prediction of human driving behavior using deep learning: a recurrent learning structure. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 647–653. IEEE, 2022.

[17] Sélim Chefchaouni Moussaoui, Hugo Pousseur, Alessandro Corrêa Victorino, and Marie-Hélène Abel. Dynamic context awareness in autonomous navigation. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2808–2813. IEEE, 2022.

[18] Marcone Ferreira Santos, Alessandro Corrêa Victorino, and Hugo Pousseur. Model-based and machine learning-based high-level controller for autonomous vehicle navigation: lane centering and obstacles avoidance. *IAES International Journal of Robotics and Automation*, 12(1):84, 2023.

[19] Hugo Pousseur and Alessandro Corrêa Victorino. Gradient descent dynamic window approach to the mobile robot autonomous navigation. In *IEEJ International Workshop on Sensing, Actuation, Motion Control, and Optimization (SAMCON 2022)*, 2022.

[20] Takumi Ueno, Hugo Pousseur, Binh Minh Nguyen, Alessandro Correa Victorino, and Hiroshi Fujimoto. Proposal of on-board camera-based driving force control method for autonomous electric vehicles. In *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 424–429. IEEE, 2023.

[21] Hugo Pousseur and Alessandro Correa Victorino. General and multi-criteria approach to study the admissibility and quality of a driving intention. In *2023 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2023.

[22] Theunissen Mathilde, POUSSEUR Hugo, Castillo Pedro, and Victorino Alessandro. Cooperative architecture using air and ground vehicles for the search and recognition of targets. In *2023 IEEE/ITSC*. IEEE, 2023.

[23] David A Abbink, Tom Carlson, Mark Mulder, Joost CF de Winter, Farzad Aminravan, Tricia L Gibo, and Erwin R Boer. A topology of shared control systems—finding common ground in diversity. *IEEE Transactions on Human-Machine Systems*, 48(5):509–525, 2018.

[24] Thomas B Sheridan, William L Verplank, and TL Brooks. Human/computer control of undersea teleoperators. In *NASA. Ames Res. Center The 14th Ann. Conf. on Manual Control*, 1978.

[25] Toshiyuki Inagaki et al. Adaptive automation: Sharing and trading of control. *Handbook of cognitive task design*, 8:147–169, 2003.

[26] David A Abbink, Mark Mulder, and Erwin R Boer. Haptic shared control: smoothly shifting control authority? *Cognition, Technology & Work*, 14:19–28, 2012.

[27] Mauricio Marcano, Sergio Díaz, Joshué Pérez, and Eloy Irigoyen. A review of shared control for automated vehicles: Theory and applications. *IEEE Transactions on Human-Machine Systems*, 2020.

[28] John A. Michon. *A Critical View of Driver Behavior Models: What Do We Know, What Should We Do?*, pages 485–524. Springer US, Boston, MA, 1985.

[29] A. Hamish Jamson, Daryl L. Hibberd, and Natasha Merat. Interface design considerations for an in-vehicle eco-driving assistance system. *Transportation Research Part C: Emerging Technologies*, 58:642–656, 2015.

[30] D.L. Hibberd, A.H. Jamson, and S.L. Jamson. The design of an in-vehicle assistance system to support eco-driving. *Transportation Research Part C: Emerging Technologies*, 58:732–748, 2015.

[31] Xiaodong Wu, Chengrui Su, and Liang Yan. Human–machine shared steering control for vehicle lane changing using adaptive game strategy. *Machines*, 11(8):838, 2023.

[32] Louay Saleh, Philippe Chevrel, Fabien Claveau, Jean-François Lafay, and Franck Mars. Shared steering control between a driver and an automation: Stability in the presence of driver behavior uncertainty. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):974–983, 2013.

[33] Boussaad Soualmi, Chouki Sentouh, Jean-Christophe Popieul, and Serge Debernard. Automation-driver cooperative driving in presence of undetected obstacles. *Control engineering practice*, 24:106–119, 2014.

[34] Ziya Ercan, Ashwin Carvalho, H Eric Tseng, Metin Gökaşan, and Francesco Borrelli. A predictive control framework for torque-based steering assistance to improve safety in highway driving. *Vehicle system dynamics*, 56(5):810–831, 2018.

[35] Michael Flad, Lukas Fröhlich, and Sören Hohmann. Cooperative shared control driver assistance systems based on motion primitives and differential games. *IEEE Transactions on Human-Machine Systems*, 47(5):711–722, 2017.

[36] Chouki Sentouh, Serge Debernard, Jean-Christophe Popieul, and Frédéric Vanderhaegen. Toward a shared lateral control between driver and steering assist controller. *IFAC Proceedings Volumes*, 43(13):404–409, 2010.

[37] Abbink D.A. and Mulder M. Neuromuscular analysis as a guideline in designing shared control. In Mehrdad Hosseini Zadeh, editor, *Advances in Haptics*, chapter 27. IntechOpen, Rijeka, 2010.

[38] Shirine El Zaatari, Mohamed Marei, Weidong Li, and Zahid Usman. Cobot programming for collaborative industrial tasks: An overview. *Robotics and Autonomous Systems*, 116:162–180, 2019.

[39] David A Abbink and Mark Mulder. *Neuromuscular analysis as a guideline in designing shared control*. INTECH Open Access Publisher, 2010.

[40] Chouki Sentouh, Anh-Tu Nguyen, Mohamed Amir Benloucif, and Jean-Christophe Popieul. Driver-Automation Cooperation Oriented Approach for Shared Control of Lane Keeping Assist Systems. *IEEE Transactions on Control Systems Technology*, 27(5):1962–1978, September 2019.

[41] Ziya Ercan, Ashwin Carvalho, H. Eric Tseng, Metin Gökaşan, and Francesco Borrelli. A predictive control framework for torque-based steering assistance to improve safety in highway driving. *Vehicle System Dynamics*, 56(5):810–831, 2018.

[42] Alexandre Sanfelice Bazanella, Lucíola Campestrini, and Diego Eckhard. *Data-driven controller design: the H2 approach*. Springer Science & Business Media, 2011.

[43] Sterling J. Anderson, James M. Walker, and Karl Iagnemma. Experimental performance analysis of a homotopy-based shared autonomy framework. *IEEE Transactions on Human-Machine Systems*, 44(2):190–199, 2014.

[44] Shriram Jugade. *Shared control authority between human and autonomous driving system for intelligent vehicles*. PhD thesis, Université de Technologie de Compiègne, 2019.

[45] Joaquín López, Pablo Sánchez-Vilariño, Rafael Sanz, and Enrique Paz. Efficient local navigation approach for autonomous driving vehicles. *IEEE Access*, 9:79776–79792, 2021.

[46] Isaac Skog and Peter Handel. In-car positioning and navigation technologies—a survey. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):4–21, 2009.

[47] Yutian Wu, Yueyu Wang, Shuwei Zhang, and Harutoshi Ogai. Deep 3d object detection networks using lidar data: A review. *IEEE Sensors Journal*, 21(2):1152–1171, 2020.

[48] Jiaxing Zhang, Wen Xiao, Benjamin Coifman, and Jon P Mills. Vehicle tracking and speed estimation from roadside lidar. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13:5597–5608, 2020.

[49] Weiyu Hao. Review on lane detection and related methods. *Cognitive Robotics*, 3:135–141, 2023.

[50] Xian-Bin Cao, Hong Qiao, and John Keane. A low-cost pedestrian-detection system with a single optical camera. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):58–67, 2008.

[51] Kaijun Zhou, Lingli Yu, Ziwei Long, and Siyao Mo. Local path planning of driverless car navigation based on jump point search method under urban environment. *Future Internet*, 9(3), 2017.

[52] Zhu Wennan, Chen Qiang, and Wang Hong. Lane detection in some complex conditions. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 117–122. IEEE, 2006.

[53] Gurveen Kaur and Dinesh Kumar. Lane detection techniques: A review. *International Journal of Computer Applications*, 112(10), 2015.

[54] Yang Xing, Chen Lv, Long Chen, Huaji Wang, Hong Wang, Dongpu Cao, Efstathios Velenis, and Fei Yue Wang. Advances in Vision-Based Lane Detection: Algorithms, Integration, Assessment, and Perspectives on ACP-Based Parallel Vision. *IEEE/CAA Journal of Automatica Sinica*, 5(3):645–661, 2018.

[55] Amol Borkar, Monson Hayes, and Mark T. Smith. Polar randomized hough transform for lane detection using loose constraints of parallel lines. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1037–1040, 2011.

[56] Mohamed Aly. Real time detection of lane markers in urban streets. In *2008 IEEE Intelligent Vehicles Symposium*. IEEE, June 2008.

[57] J.C. McCall and M.M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):20–37, 2006.

[58] Qin Zou, Hanwen Jiang, Qiyu Dai, Yuanhao Yue, Long Chen, and Qian Wang. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE transactions on vehicular technology*, 69(1):41–54, 2019.

[59] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[60] Sajjad Mozaffari, Omar Y Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):33–47, 2020.

[61] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14, 2014.

[62] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[63] Florent Altché and Arnaud de La Fortelle. An lstm network for highway trajectory prediction. In *2017 IEEE 20th international conference on intelligent transportation systems (ITSC)*, pages 353–359. IEEE, 2017.

[64] Roger Grosse. Lecture 15: Exploding and vanishing gradients. *University of Toronto Computer Science*, 2017.

[65] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[66] Alex Zyner, Stewart Worrall, James Ward, and Eduardo Nebot. Long short term memory for driver intent prediction. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1484–1489. IEEE, 2017.

[67] Derek J Phillips, Tim A Wheeler, and Mykel J Kochenderfer. Generalizable intention prediction of human drivers at intersections. In *2017 IEEE intelligent vehicles symposium (IV)*, pages 1665–1670. IEEE, 2017.

[68] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019.

[69] Shengzhe Dai, Li Li, and Zhiheng Li. Modeling vehicle interactions via modified lstm models for trajectory prediction. *IEEE Access*, 7:38287–38296, 2019.

[70] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 336–345, 2017.

[71] Alex Zyner, Stewart Worrall, and Eduardo Nebot. A recurrent neural network solution for predicting driver intention at unsignalized intersections. *IEEE Robotics and Automation Letters*, 3(3):1759–1764, 2018.

[72] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[73] Kyuhwan Yeon. EGO-VEHICLE SPEED PREDICTION USING A LONG SHORT-TERM MEMORY BASED RECURRENT NEURAL NETWORK. *International Journal of ...*, 13(2):293–300, 2019.

[74] Nachiket Deo and Mohan M Trivedi. Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1468–1476, 2018.

[75] Kyuhwan Yeon, Kyunghan Min, Jaewook Shin, Myoungho Sunwoo, and Manbae Han. Ego-vehicle speed prediction using a long short-term memory based recurrent neural network. *International Journal of Automotive Technology*, 20:713–722, 2019.

[76] Ilija Ilievski, Taimoor Akhtar, Jiashi Feng, and Christine Shoemaker. Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[77] SM Sohel Mahmud, Luis Ferreira, Md Shamsul Hoque, and Ahmad Tavassoli. Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. *IATSS research*, 41(4):153–163, 2017.

[78] Christer Hydén. The development of a method for traffic safety evaluation: The swedish traffic conflicts technique. *Bulletin Lund Institute of Technology, Department*, 1987.

[79] Nobuhiro Uno, Yasunori Iida, Shinji Itsubo, and Shinji Yasuhara. A microscopic analysis of traffic conflict caused by lane-changing vehicle at weaving section. In *Proceedings of the 13th mini-EURO conference-handling uncertainty in the analysis of traffic and transportation systems, Bari, Italy*, pages 10–13, 2002.

[80] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[81] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[82] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *ieee Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[83] François Chaumette, Seth Hutchinson, and Peter Corke. Visual servoing. In *Springer Handbook of Robotics*, pages 841–866. Springer, 2016.

[84] François Chaumette. Image moments: a general and useful set of features for visual servoing. *IEEE Transactions on Robotics*, 20(4):713–723, 2004.

[85] Andrea Cherubini, François Chaumette, and Giuseppe Oriolo. Visual servoing for path reaching with nonholonomic robots. *Robotica*, 29(7):1037–1048, 2011.

[86] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[87] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[88] Emanuele Venzano, Hugo Pousseur, Alessandro Correa Victorino, and Pedro Castillo Garcia. Motion control for aerial and ground vehicle autonomous platooning. In *2022 IEEE 17th International Conference on Advanced Motion Control (AMC)*, pages 213–218, 2022.

[89] Johann Borenstein, Yoram Koren, et al. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991.

[90] Matej Dobrevski and Danijel Skočaj. Adaptive dynamic window approach for local navigation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6930–6936. IEEE, 2020.

[91] Long Xin, Pin Wang, Ching-Yao Chan, Jianyu Chen, Shengbo Eben Li, and Bo Cheng. Intention-aware long horizon trajectory prediction of surrounding vehicles using dual lstm networks. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1441–1446. IEEE, 2018.

[92] Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung, and Jun Won Choi. Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1672–1678. IEEE, 2018.

[93] Sajjad Mozaffari, Omar Y. Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–15, 2020.

[94] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A. Chapman, Dongpu Cao, and Jonathan Li. Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2021.

[95] Florent Altche and Arnaud De La Fortelle. An LSTM network for highway trajectory prediction. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-March:353–359, 2018.

[96] Qinxue Meng, Daniel Catchpoole, David Skillicom, and Paul J. Kennedy. Relational autoencoder for feature extraction. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 364–371, 2017.

[97] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[98] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[99] Shahram Azadi, Reza Kazemi, and Hamidreza Rezaei Nedamani. Chapter 6 - dynamic behavior and stability of an articulated vehicle carrying fluid. In Shahram Azadi, Reza Kazemi, and Hamidreza Rezaei Nedamani, editors, *Vehicle Dynamics and Control*, pages 213–260. Elsevier, 2021.

[100] Ahmad Shour, Hugo Pousseur, Alessandro Correa Victorino, and Veronique Cherfaoui. Shared decision-making forward an autonomous navigation for intelligent vehicles*. *IEEE International Conference on Systems, Man and Cybernetics*, 2021.

[101] Xiaodong Wu, Chengrui Su, and Liang Yan. Human–machine shared steering control for vehicle lane changing using adaptive game strategy. *Machines*, 2023.

[102] Béatrice Pano, Philippe Chevrel, Fabien Claveau, Chouki Sentouh, and Franck Mars. Obstacle avoidance in highly automated cars: Can progressive haptic shared control make it safer and smoother? *IEEE Transactions on Human-Machine Systems*, 52, 08 2022.

[103] Francisco Facchinei and Christian Kanzow. Generalized nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211, 2010.