

Synthetic Data Generation and Knowledge Representation for Overtaking Scenarios in Autonomous Driving

Mariana Filipa Pires Pinto

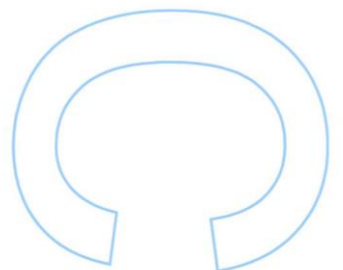
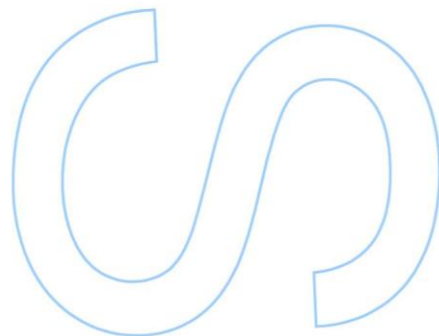
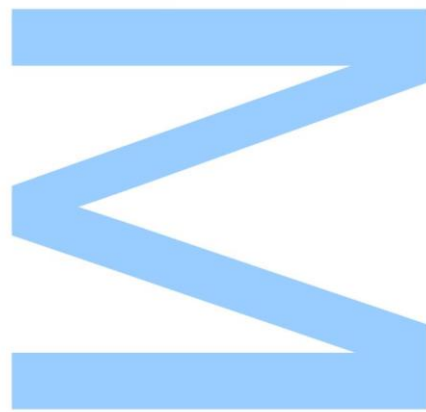
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência dos Computadores
2022

Orientador

Inês de Castro Dutra
Professora Auxiliar
Faculdade de Ciências da Universidade do Porto

Supervisor

Joaquim Fonseca
Bosch Car Multimedia Portugal, S.A.

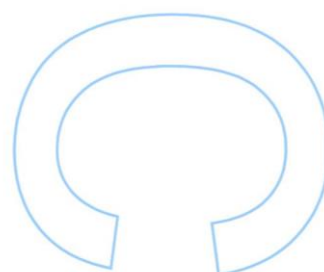
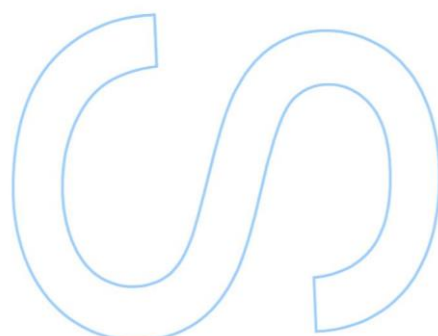
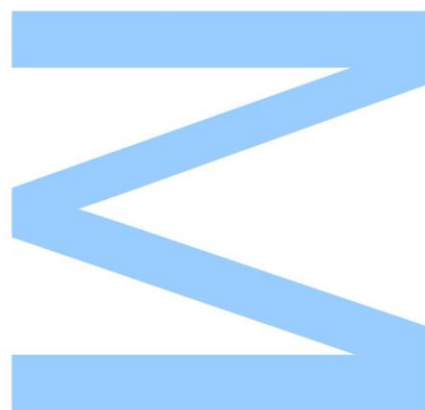




Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ____/____/____



Declaração de Honra

Eu, Mariana Filipa Pires Pinto, inscrita no Mestrado em Engenharia de Redes e Sistemas Informáticos da Faculdade de Ciências da Universidade do Porto declaro, nos termos do disposto na alínea a) do artigo 14.º do Código Ético de Conduta Académica da U.Porto, que o conteúdo da presente dissertação reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação, declaro, ainda, que a mesma é resultado do meu próprio trabalho de investigação e contém contributos que não foram utilizados previamente noutros trabalhos apresentados a esta ou outra instituição.

Mais declaro que todas as referências a outros autores respeitam escrupulosamente as regras da atribuição, encontrando-se devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Não são divulgados na presente dissertação quaisquer conteúdos cuja reprodução esteja vedada por direitos de autor. Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Mariana Pinto

17 de setembro de 2022

Agradecimentos

Gostaria de agradecer à minha orientadora, a professora Inês Dutra da Faculdade de Ciências da Universidade do Porto, pela orientação e assistência prestada durante esta dissertação. Ao meu supervisor Eng. Joaquim Fonseca da Bosch Car Multimedia por todo o apoio, disponibilidade e preocupação.

Gostaria também de agradecer à Bosch Car Multimedia e a toda a equipa “*THEIA Perception Innovation*” por me proporcionar a oportunidade de desenvolver este trabalho num ambiente acolhedor e desafiante.

Agradeço também aos meus pais por todo o encorajamento e motivação e à minha irmã por, mesmo ausente, se fazer presente e me ajudar sempre que necessito. Sem vocês, isto não era possível.

Às minhas amigas Cláudia, Rita e Sara e ao meu amigo Lucas por ser o meu braço direito durante todos estes anos dentro e fora da faculdade.

Ao meu namorado Mário pela disponibilidade em todos os momentos e por todos os valores e conselhos que me deu, obrigada!

Por fim, um especial obrigado aos meus avós que, mesmo não estando entre nós, espero tê-los deixado orgulhosos!

Este trabalho é apoiado pelos Fundos Estruturais e de Investimento Europeus na componente FEDER, através do Programa Operacional Competitividade e Internacionalização (COMPETE 2020) [Projeto nº 047264; Referência de financiamento: POCI-01-0247-FEDER-047264].

Abstract

In recent times, autonomous driving has become one of the most popular research topics within Artificial Intelligence. An autonomous vehicle is understood as a system that combines perception, decision-making, planning, and control. Decision-making can be seen as the brain of an autonomous vehicle. The uncertainty and unpredictability of driving environments pose complex challenges for fully optimized driving, whose main requirement is to ensure the safety of all road users. The overtaking manoeuvre is one of the most critical actions of driving. The process involves lane changes, acceleration and deceleration actions, and estimation of the speed and distance of the vehicle in front or in the lane in which it is moving.

This dissertation is developed within the THEIA Innovation project, which consists of the research and development of several modules that incorporate a perception system for Autonomous Driving. In specific, this work contributes with the process of decision-making for overtaking scenarios. Given the lack of public available datasets for this purpose, a new dataset was created using the CARLA simulator. In order to create this dataset, a thorough review of the literature was performed and features were either created or taken from other works related with automatic overtaking scenarios. From the data generated by the simulator, a knowledge base was created containing characteristics and factors important for overtaking, such as driver behaviour and the sequence of actions that ensure safety and legality during the overtaking manoeuvre. This knowledge was coded as a set of rules. Finally, these rules were added to the synthetic dataset as new boolean features. Each simulation was manually labelled as “successful legal overtake”, “successful illegal overtake”, “non-successful without collisions”, “non-successful with collisions” and “no attempt to overtake”. Statistical analysis and machine learning methods were then applied with two purposes: (1) to highlight the most important features involved in successful and unsuccessful overtaking scenarios, and (2) to support car decisions by predicting if a sequence of actions will result in a successful or unsuccessful overtaking manoeuvre.

Keywords: Autonomous Driving; Synthetic Data Generation; Knowledge Representation; Decision-making; Machine Learning; Overtaking

Resumo

Nos últimos tempos, a condução autónoma tem vindo a ser um dos temas de investigação mais populares no âmbito da Inteligência Artificial. O veículo autónomo é entendido como um sistema que combina perceção, tomada de decisão, planeamento e controlo. A tomada de decisão pode ser vista como sendo o cérebro de um veículo autónomo. A incerteza e a imprevisibilidade dos ambientes de condução representam complexos desafios para uma condução totalmente otimizada cujo principal requisito é assegurar a segurança de todos os utilizadores da estrada. A manobra de ultrapassagem é uma das ações mais críticas da condução. O processo envolve manobras de mudança de via, ações de aceleração e desaceleração e estimativa da velocidade e distância do veículo em frente ou na via para a qual se move.

Esta dissertação é desenvolvida no âmbito do projeto de Inovação THEIA, que consiste na investigação e desenvolvimento de diversos módulos que incorporam um sistema de perceção para Condução Autónoma. Mais especificamente, este trabalho contribui com o processo de tomada de decisão para cenários de ultrapassagem. Dada a falta de conjuntos de dados disponíveis ao público para este fim, foi criado um novo conjunto de dados utilizando o simulador CARLA. A fim de criar este conjunto de dados, foi feita uma revisão exaustiva da literatura e foram criadas ou retiradas características de outros trabalhos relacionados com cenários de ultrapassagem autónoma. A partir dos dados gerados pelo simulador, foi criada uma base de conhecimento contendo características e fatores importantes para a ultrapassagem, tais como o comportamento do condutor e a sequência de ações que garantem a segurança e a legalidade durante a manobra de ultrapassagem. Este conhecimento foi codificado como um conjunto de regras. Finalmente, estas regras foram acrescentadas ao conjunto de dados sintéticos como novas características booleanas. Cada simulação foi rotulada manualmente como "ultrapassagem legal bem sucedida", "ultrapassagem ilegal bem sucedida", "ultrapassagem mal sucedida sem colisões", "ultrapassagem mal sucedida com colisões" e "sem tentativa de ultrapassagem". A análise estatística e os métodos de *machine learning* foram então aplicados com dois objetivos: (1) destacar as características mais importantes envolvidas em cenários de ultrapassagem bem ou mal sucedidos e (2) apoiar as decisões do automóvel, prevendo se uma sequência de ações resultará numa manobra de ultrapassagem bem ou mal sucedida.

Palavras-chave: Condução Autónoma; Geração de dados sintéticos; Representação do conhecimento; Tomada de decisão; Aprendizagem automática; Ultrapassagens

À minha avó

Contents

Declaração de Honra	v
Agradecimentos	vii
Abstract	ix
Resumo	xi
Contents	xv
List of Tables	xviii
List of Figures	xxi
Listings	xxiii
Acronyms	xxv
1 Introduction	1
1.1 Context	1
1.2 Motivation	3
1.3 Objectives	3
1.4 Contributions	4
1.5 Document Structure	4
2 Basic Concepts	5

2.1	Sensors	6
2.2	Structured Data	8
2.3	Data Augmentation	9
2.4	Knowledge Representation	10
2.5	Decision-Making	14
2.6	Feature Engineering	16
2.7	Evaluation Metrics for Classification	18
3	Related Work	21
3.1	Current State of Autonomous Driving	22
3.2	Inputs for Decision-Making	25
3.2.1	Perception	25
3.2.2	Datasets and Simulators	28
3.2.3	Driveability Factors	33
3.3	Decision-Making in Overtaking	36
3.4	Knowledge Representation	38
3.5	Inference, Reasoning and Learning	40
3.5.1	Inference Methods	40
3.5.2	Learning to Make Decisions	41
4	Synthetic Data Generation and Knowledge Base	43
4.1	Data Generation	44
4.1.1	Limitations of the Simulator	45
4.1.2	Scenarios	47
4.1.3	Collected Data	48
4.1.4	Planning and Structuring	50
4.1.5	Classification	52
4.1.6	Feature Engineering	54
4.1.7	Traffic Rules	56

4.2	Materials and Methods	59
4.2.1	Carla Simulator	59
4.2.2	Scenarios and Data Collected	63
4.2.3	Data Exploration	64
4.2.4	Traffic Rules	67
4.3	Results	70
4.3.1	Features Statistics	70
4.3.2	Data Visualization	74
4.3.3	Traffic Rules	81
4.3.4	Data Validation	85
4.3.5	Discussion of the Results	88
5	Prediction of Overtaking Scenarios	91
5.1	Materials and Methods	91
5.1.1	Prediction and Features Importance	91
5.1.2	Temporal Data	95
5.2	Results	96
5.2.1	Prediction and Features Importance	96
5.2.2	Temporal Data	103
5.2.3	Discussion of the Results	105
6	Conclusions	107
6.1	Future Work	108
	Bibliography	111
A	Statistics of Static Features	123

List of Tables

2.1	Advantages and disadvantages of various knowledge representation techniques [154] [76] [67] [133].	13
2.2	Characteristic of different methods for decision-making. Adapted from [93]. . . .	16
3.1	Currently systems implemented in autonomous vehicles [19] [33] [30]. Comfort and Safety are represented on a scale of 1 to 5: 1 is (-), 2 is (-), 3 is (+), 4 and 5 are (++).	23
3.2	Comparison of sensors used for autonomous driving. Adapted from [69] and [92]. Legend: (++): perfectly adapted sensors; (+): sensors having good performances; (-): sensors potentially meeting the criteria but with possible disadvantages; (--) : sensors that can be used with adaptations and additional heavy treatments; (0): sensors unable to meet the criteria.	28
3.3	Comparison of main datasets' features. Adapted from [37] and [86]. Legend: (-) indicates that no information is provided.	29
3.4	Comparison of popular simulators. Adapted from [84]. Legend: Y indicates that simulator has the corresponding requirement; N indicates the opposite; (-) indicates that no information is provided.	32
4.1	Stored variables en each simulation. Legend: idv represents the vehicle id and id_lane represents the id of the lane the vehicle is in.	48
4.2	Static Features Created.	54
4.3	Dynamic Features Created.	54
4.4	Abbreviation of predicates and constants.	69
4.5	Statistics of Dynamic Features.	71
4.6	Statistics of Static Features.	72
4.7	Information about frame 70 of simulation 1 extracted from dataset.	75

5.1	Confusion Matrix.	97
5.2	Confusion Matrix.	99
5.3	Distribution of true and negative values for Success_I class.	99
5.4	Results of the metrics applied to confusion matrix.	99
5.5	Statistics of the speed and distance measured up to the first moment of overtaking.	104
5.6	Statistics of the speed and distance measured up to the second moment of overtaking.	105

List of Figures

1.1	The five steps of automated driving. Adapted from: [33][36].	2
1.2	Overview of the autonomous navigation process [162].	2
2.1	Example of multi-sensors architectures and topologies for environment perception [69].	6
2.2	How self-driving vehicles “see” the world [170].	8
2.3	Example of a semantic network [67].	12
2.4	Example of a frame system [76].	12
3.1	Outcome of the SBS method applied to the factors in the research by Yang et al. [174].	38
3.2	A small part of a knowledge base [64].	39
3.3	Example of hypergraph: vehicle driving knowledge base [178].	40
3.4	The scene understanding flowchart by event reasoning framework for autonomous driving [172].	41
4.1	Pipeline of the first phase of the work.	44
4.2	Third-person view in four weather conditions [54].	45
4.3	Overtaking scenarios with multiple weather conditions.	48
4.4	ER diagram developed.	51
4.5	Classes chosen to represent the different overtaking scenarios. Legend: The blue colour represents the ego vehicle; the green colour represents the vehicle that the ego vehicle wants to overtake; the yellow colour represents simple cars; the dashed black line represents the car’s intended path.	53
4.6	Representation of the 3 stages of a successful overtaking.	53

4.7	Example of the two-second rule approach.	57
4.8	Logic for calculating the distance between two cars from the data collected by the simulator. Legend: pink point: centre point to vehicle location (x, y, z); purple point: loc_e + a; yellow point: loc_p - b; red arrow: distance between the two vehicles = b-a	58
4.9	Street Portion Layout.	64
4.10	Features Relation. Legend: (a) corresponds to pair plot output for all features; (b) corresponds to correlation points between SE and SP; (c) corresponds to correlation points between DSEP and SE.	66
4.11	Portion of dataset with R column added	70
4.12	Screenshot of simulation 1 video at frame 70.	75
4.13	Features Dataset Portion.	75
4.14	Frequency of classes.	76
4.15	Histogram for each feature.	77
4.16	Box plot diagram for each feature.	78
4.17	Box and swarm plots for SE feature grouped by classes.	79
4.18	Correlation matrix.	80
4.19	P-values.	80
4.20	Histogram for violated traffic rules grouped by class.	82
4.21	Correlation Matrix for traffic rules violations.	83
4.22	P-values.	83
4.23	Histograms of the mean values of each continuous feature for variables B_C, B_V, and B_None of value 1.	85
4.24	Total overtaking time for different types of vehicles.	86
4.25	Speed profile (km/h) of the overtaking and overtaken vehicles.	87
4.26	Overtaking time for different vehicle types (seconds).	88
5.1	Pipeline of the second phase of the work.	91
5.2	Number of occurrences of each class before and after applying SMOTE.	98
5.3	Plot of AUC-ROC curve for Random Forest Classifier.	100

5.4	AUC-ROC curve for SVM and LSVM metrics.	101
5.5	Feature importance scores provided by the model.	102
5.6	Decision tree built from the features presented.	103
5.7	Correlation matrix computed from actions features.	103
6.1	Prototype of the dashboard with predictive model implemented. Image adapted from [75].	109
A.1	Statistics of Static Features.	124

Listings

4.1	Code sample for calculating the minimum distance for safe overtaking.	57
4.2	Code sample to display the histogram and the frequencies of each class.	64
4.3	Code sample to compute the correlation matrix.	66
4.4	Code sample to display a dataset portion.	75
4.5	Code sample to display box and swarm plots.	78
5.1	Code sample to split the dataset into test and training.	93
5.2	Code sample to scale the data.	93
5.3	Code sample to apply RandomForestClassifier method.	93
5.4	Code sample to obtain the average accuracy of the random forest.	96
5.5	Code sample to apply cross-validation technique.	97
5.6	Code sample to apply SMOTE and RandomForestClassifier method.	98
5.7	Code sample to generate the features importance graph.	101
5.8	Code sample to prove the importance of feature D in the classification of an overtake.102	

Acronyms

AI	Artificial Intelligence	LiDAR	Light Detection and Ranging
AUC	Area Under the Curve	NHTSA	National Highway Traffic Safety Administration
CNN	Convolutional Neural Networks	RNN	Recurrent Neural Network
DNN	Deep Neural Network	ROC	Receiver Operating Characteristic
ER	Entity Relationship	ROS	Robot Operating System
GAN	Generative Adversarial Network	SAE	Society of Automotive Engineers
ITS	Intelligent Transport Systems	SBS	Sequential Backward Selection
KB	Knowledge Base	SMOTE	Synthetic Minority Oversampling TEchnique
KR	Knowledge Representation	SVM	Support Vector Machines
KRR	Knowledge Representation and Reasoning		

Chapter 1

Introduction

With the significant advancement of autonomous driving technologies, various decision-making systems for overtaking manoeuvres have been developed [174]. However, there are a variety of challenges that arise in uncertain driving scenarios. Determining what type of information should be examined during the decision and developing a methodology to derive a decision from the specified factors are among these challenges. To safely and reliably overcome, it is required to thoroughly study the variables that are important to the decision and adopt an appropriate strategy.

1.1 Context

National Highway Traffic Safety Administration (NHTSA) defines autonomous vehicles as those in which at least some aspects of a safety-critical control function (e.g., steering, throttle, or braking) occur without direct driver input. Vehicles that provide safety warnings to drivers (forward crash warning, for example) but do not perform a control function are, in this context, not considered automated, even though the technology necessary to provide that warning involves varying degrees of automation (e.g., the necessary data are received and processed, and the warning is given, without driver input).” [4]. The number of autonomous vehicles capable of facilitating driver’s lives has been increasing in the past years. It can warn the driver about the possible danger provoked by other cars’ manoeuvres, breaking when the driver ahead stops suddenly, warning of possible vehicle failures or even advising the driver to take a break if he is tired. This technology is ensured by a combination of hardware (sensors, cameras, radar, LiDAR, etc.) and software to guarantee safety and allow the required actions to avoid accidents.

The Society of Automotive Engineers (SAE) defines a total of six levels of automated driving, from no automation (level 0) to fully automation (level 5) [155]. Figure 1.1 describes each level in detail.

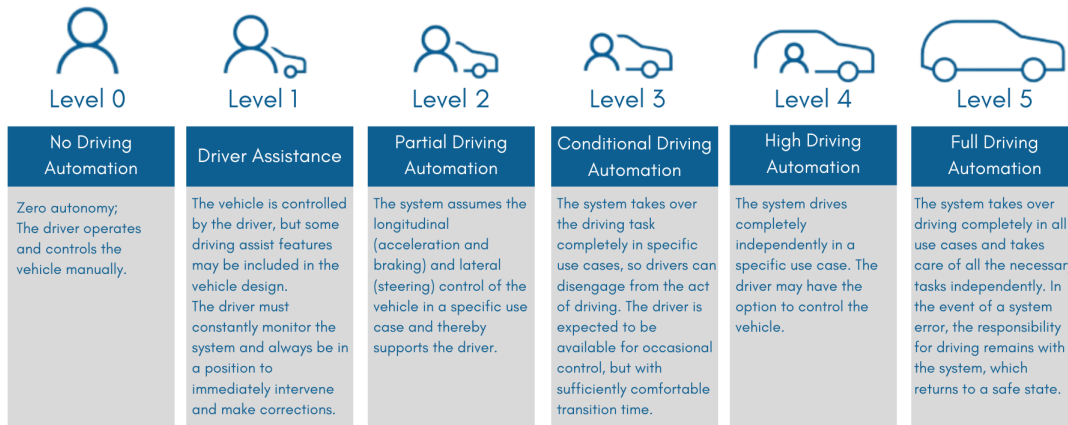


Figure 1.1: The five steps of automated driving. Adapted from: [33][36].

Consumer Reports claims that “there are no self-driving cars on the road today that consumers can buy” [24]. Almost every new car sold in the U.S. corresponds to levels 0, 1 and 2. In 2021, Honda became the first manufacturer to provide a legally approved Level 3 vehicle [71]. Cars in level 4 or 5 aren’t yet available for the public to buy, but many of today’s cars offer automatic systems that improve the driver’s experience and increase their safety. The interior monitoring system, present in the cars, does not just focus on drivers. The system can detect the presence of any other occupants and can then provide more specific reminders to fasten the seat belt, deactivate the airbag if a child safety seat is present or change temperature in the vehicle interior [148].

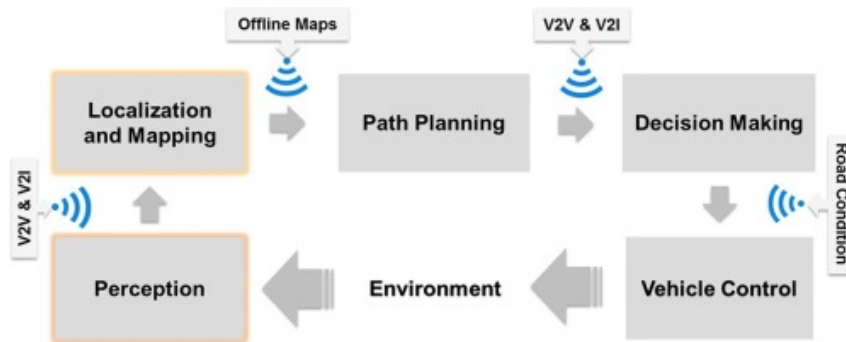


Figure 1.2: Overview of the autonomous navigation process [162].

As can be observed in (Figure 1.2), autonomous driving process can be split into 5 areas, namely Perception, Localization and Mapping, Path Planning, Decision-Making and Vehicle Control. Perception uses sensors to understand and identify everything present in the autonomous car’s surroundings. Localization and mapping algorithms use sensor data and other perception outputs to calculate the global and local location of the vehicle and map the environment. Path planning uses knowledge of the environment and its position based on perception, localization, and mapping information to determine possible safe routes. The decision-making component is responsible for ensuring the optimal route. Features such as weather conditions, road signs, the

current vehicle state are important in this stage. The vehicle control consists in calculating the appropriate vehicle command like acceleration or torque.

1.2 Motivation

Many researchers regard decision-making to be the “brain” of the entire autonomous navigation process. Serious accidents can result from mistakes committed during this phase. The entire environment around the car must be fully understood, and the biggest issue is determining what information is critical to include at this stage. This involves determining the decision’s motive, acquiring the most relevant data, and weighing all the alternatives and repercussions of the decision. Finally, one must evaluate the decision and, if required, enhance the entire process. Therefore, this work aims to generate, in a synthetic way, the data that is needed to understand and analyse overtaking scenarios as effectively as possible. Additionally, it aims to compile the most pertinent data to support decision-making that ensures safety and follows traffic regulations.

This dissertation is developed under the THEIA Innovation project, which comprehends the research and development of a perception system to address the different tasks involved in the autonomous driving activity. This project aims to investigate new methods of knowledge representation and applying inference techniques to improve the perception layer, minimizing the need for high representativeness of specific classes in the training dataset (cars, people, bicycles, trucks, etc.). As a result, it intends to add a logic layer to the perception layer, allowing autonomous vehicles to reason about their surroundings and determine what are the most important factors to take into account during the decision-making process.

1.3 Objectives

There are several challenges encountered while collecting data for autonomous driving. The difficulty in reproducing inaccurate and often dangerous real-world scenarios brings as an alternative the use of synthetic data. The main goal of this dissertation is to generate a synthetic dataset focusing on overtaking scenarios in the context of autonomous driving, which can support decision-making evaluation with the aid of a prediction model. In this sense, two main steps are highlighted. In the first stage, simulations are generated from the CARLA autonomous driving simulator that represent possible overtaking scenarios on a one-way highway. With this, it is intended to build a dataset with all relevant information collected by the ego vehicle – the automated vehicle that contains the sensors that perceive the environment around the vehicle – during the simulation. In a second phase, the collected data is analysed to assist more effectively the decision-making process. Given a collecting data, a prediction model is used to forecast how the overtaking manoeuvre would be classified. With this implementation, the relevance and influence of each factor present in the simulation is obtained, as well as the relationship between them and with the manoeuvre classification. A sequence of actions by the driver to ensure legality

and safety to the overtaking manoeuvre is also studied.

1.4 Contributions

To drive safely, autonomous driving vehicles rely on accurate and efficient perception and decision-making. However, current methods for solving these challenges often require large amounts of data that are difficult to obtain in the real world. In order to fully evaluate decision-making, accident scenarios must be represented in the datasets. As such, the main contributions regarding this dissertation are: the discovery of new methodologies for generating scalable and realistic data that can represent real driving scenarios through simulation environments; the collection of several important factors to be considered for decision-making during overtaking; the extraction and representation of knowledge that supports decision-making and a vast literature review gathering the main studies in the area of autonomous driving.

1.5 Document Structure

This document is organized in five additional chapters. Chapter 2 presents a set of concepts and fundamentals of the work. There is a brief theoretical explanation as well as some background material needed to comprehend the terminology and expressions used in the remaining chapters. Chapter 3 cites and discusses relevant research present in the literature within the scope of the work. Here, the research's strengths and weaknesses are discussed, as well as whether this work meets them and how it can complement them. The experimental/practical phase of the work is divided into two major chapters: Synthetic Data Generation and Knowledge Base and Prediction of Overtaking Scenarios. The first, Chapter 4, describes the entire phase of data generation, structure, and classification as well as feature creation and encoding of traffic laws. The methods used and experimental results for each task included are presented. For the second, Chapter 5, an overtaking class prediction model is employed, the importance of the features included in the dataset is calculated, and a study is done on the temporal data prior to the moment of overtaking in order to obtain a sequence of actions during driving that guarantees the success of the manoeuvre. As in the previous chapter, the methodologies are described in this one, along with the experimental findings and discussions. Finally, Chapter 6 draws conclusions by summarizing the work's main findings and limitations, and raises topics for future work.

Chapter 2

Basic Concepts

This chapter provides the contextualization of some important themes and concepts for this work. In the context of autonomous driving, some concepts are presented in detail, such as the types of sensors used to collect information about the environment around the vehicle. Afterwards, some ideas about data processing, its structuring and some notions of data augmentation are presented. Although the latter was not applied in the work, its study proved to be important. Subsequently, the approaches to build a knowledge base and the methods for learning-based decision-making are presented and the stages of the feature engineering process are explained. Finally, some metrics to evaluate model performance are presented.

An overview of the sections described is provided in the following list:

- **Sensors:** overview of the sensors used in autonomous driving, Section 2.1
- **Structured Data:** importance of data structuring, Section 2.2
- **Data Augmentation:** some brief notes about data augmentation, Section 2.3
- **Knowledge Representation:** a brief summary of what knowledge representation consists of, Section 2.4
- **Decision-Making:** presentation of the characteristics of the different decision-making methods, Section 2.5
- **Feature Engineering:** description of the procedure for converting raw data into features, Section 2.6
- **Evaluation Metrics for Classification:** description of various metrics that evaluate model performance, Section 2.7

2.1 Sensors

In order to drive autonomously and safely, self-driving cars always need to be focused on their surroundings and their interiors with the aid of their sensors — the key components. They collect all the information required by the vehicle to perceive its entire surroundings in order to control some car's features and to make decisions like increase or decrease its velocity, brake, change direction, etc. Cameras, radar, ultrasonic, and Light Detection and Ranging (LiDAR) sensors are the eyes and ears of an automated vehicle. There are still additional sensors enabling the system to make autonomous decisions whether the driver is capable of taking control of the vehicle again if necessary. Besides of immediate surroundings, automated cars can use up-to-date, digital maps and connect real-time from the cloud or even access external information such as traffic or weather data [149]. The fusion of all this data allows a solid basis of information for the smart vehicle computer to make exact calculations and good decisions. To guarantee more security and efficiently, the vehicle must be able to detect and localize all relevant road users within the entire vehicle environment. The Figure 2.1 shows the necessity to have a distributed architecture that allows a 360 degree perception field of view in order to take into account not only the front and rear areas but also lateral fields [69].

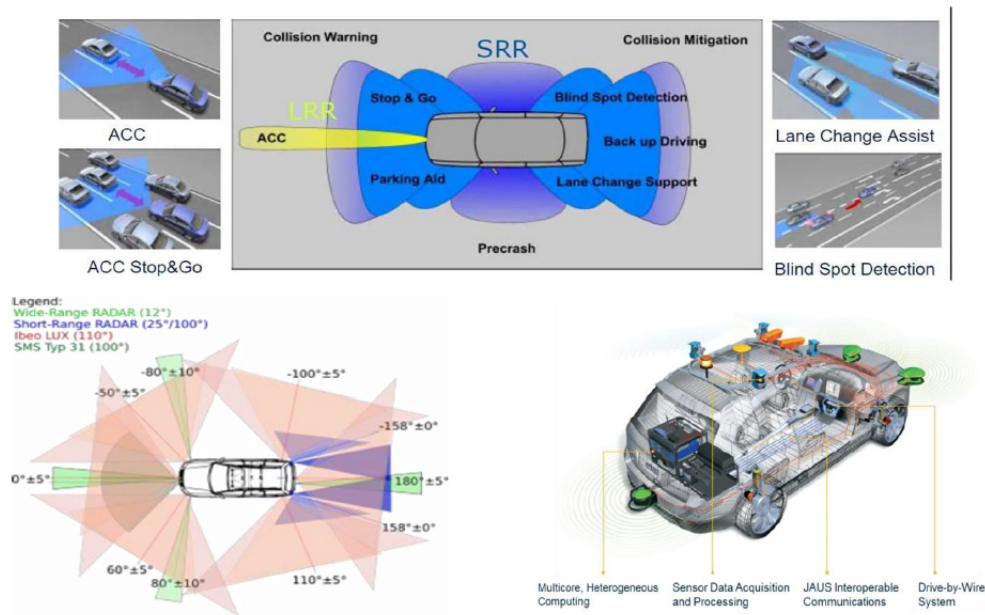


Figure 2.1: Example of multi-sensors architectures and topologies for environment perception [69].

The process of choosing sensors is based on the following characteristics [163]:

- Accuracy: is the error between the true value and the sensor's reported measurement. Accuracy value is dependent on noise levels and external interference parameters.
- Resolution: The minimum variance between two measurements.

- **Sensitivity:** The minimum value that can be measured.
- **Perspective:** Usually called field of view.
- **Dynamic Range:** The minimum and maximum values that can be accurately reported.
- **Active or Passive:** In the context of Intelligent Transport Systems (ITS), passive sensors do not emit energy to perform their measurement while active sensors emit energy, and then they measure the amount returned by the environment[69].
- **Timescale:** The refresh rate of the sensor and the frequency of the measurement bandwidth with time
- **Output Interface:** The sensor's output.

The main types of sensors are:

- **Cameras:** The cameras of the vehicle detect and categorize objects and other road users, traffic signs and road markings robustly and reliably. Near-range cameras additionally generate a three-dimensional image of the immediate vehicle's surroundings.
- **Ultrasonic:** Emit short ultrasonic pulses that are reflected by obstacles. These sensors are mostly used in close range, especially for parking.
- **LiDAR:** Light Detection and Ranging – sends out laser pulses and calculates the distance to objects and other road users. The lasers are pulsed, and the pulses are reflected by objects. These reflections return a 3D point cloud that represents these objects.
- **Radar:** Radar uses radio waves to detect the precise position, relative speed and direction of movement of objects in almost all weather conditions (e.g. darkness, tunnels) with just one measurement based on the time it takes for the waves to return.

Figure 2.2 represent the different viewpoints of each sensor. Cameras allow the technology to see colour, contrast, and texture. The filter 2.2b shows what happens in segmentation – technology to understand what (categorize) and where (position) each object is. Radar (2.2c) provides ego vehicles with information about the location, speed, and direction of moving objects. Radio waves from the transmitter reflect off objects and then return to the receiver. LiDAR (2.2d) is able to create a very effective 3D world model. It works based on the light echolocation – shoots out pulses of infrared light invisible to the human eye. It then receives light as it bounces and determines the location of objects and how far away they are (with a 360° view).

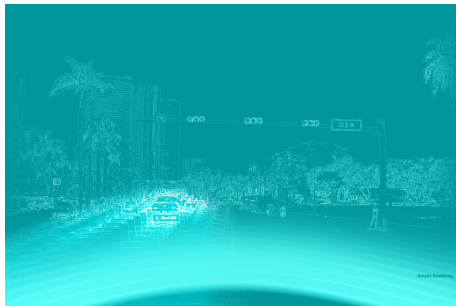
In the context of our project, LiDAR is considered as the main sensor for the perception's phase. In practice, the raw material of LiDAR is light. It works with the same concept as sonar and other radars, but instead of emitting waves to receive them back and make the distance and tracking calculations, everything is done with light, at a much higher speed. An on-board



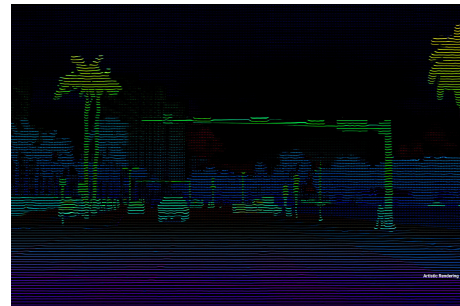
(a) Human Vision



(b) Cameras Vision



(c) Radar Vision



(d) LiDAR Vision

Figure 2.2: How self-driving vehicles “see” the world [170].

computer registers each reflective point of the laser, translating that rapidly updating point cloud into an animated 3D representation of its surroundings, all through software and in real time, which makes the car’s operation safer and more accurate. For all this to work, there are four main components: a transmitter to emit laser pulses, a receiver to intercept pulse echoes, an optical analysis system to process input data, and a computer to visualize a live three-dimensional image of the system’s environment.

2.2 Structured Data

Data in a standardized format with a well-defined structure is referred to as structured data. The data follows a data model, has a consistent arrangement, and is easily understood and retrieved by humans and programs in this format. The outcomes of using structured data are far more accurate than unstructured data. At the beginning of the last decade, the IDC (International Data Corporation) estimated that 1.2 zettabytes (1.2 trillion gigabytes) of new data were created in 2010, up from 0.8 zettabytes the year before. According to the report, by 2020, the amount of new data created would increase 44 times and reach 35 zettabytes. In 2018 there were 33 zettabytes which led IDC to predict that by 2025, 175 zettabytes of new data will be created worldwide [128]. However, the vast majority of data (about 80%) is unstructured [160], with barely 0.5 percent being used and analysed [48]. Text files, email content, social networking data, information from websites, communications, and media are examples of unstructured data.

The preference for unstructured data could be related to the fact that data, while not always well organized or accessible, is frequently the key to receive a comprehensive answer to the most pressing concerns [101]. Despite the fact that structured data accounts for only 20% of all data generated, it is the current foundation of big data. Data is a company's most valuable source of information. This can take many various forms, and measuring data that isn't quantitative, such as sentiments, behaviours, or expressions, can be challenging. As a result, organized data is required to make conclusions and gather information more easily. This information is easily incorporated into machine learning and artificial intelligence algorithms, yielding precise forecasts.

Structured data is organized in columns and rows, much like a database, such that the format, description, and meaning of the data are all clear. Data in the same group is classified into classes, and data in the same class has the same properties. The source of the data may be varied and may not require human intervention, such as sensor-collected or computer-generated data. When provided by humans, this can include information from survey responses, website clicks, and so on. Semi-structured data can be a good alternative to structured data. Although not as rigorous as structured data, this contains some of the same characteristics. This information cannot be saved in relational databases, but it does contain metadata and tags that assist its organization and indicate how it is kept. Although the items in this group may not have the same features or attributes, semi-structured data is grouped hierarchically. Structured data, in summary, allows for easy storage and access, as well as the addition of data, the simplification of search operations, and data security. Once the database structure is fully defined, however, it has storage inflexibility. That is, if the data requirements change, all structured data will most likely need to be changed. Another issue is the data's limited application. They can only be utilized for their intended purpose because they have a specified structure, which limits their flexibility and usability [150].

2.3 Data Augmentation

To achieve safe driving that is true to the complexity of the road, the autonomous car must be able to identify everything around it. This is possible due to the sensor suite and different methods that understand the data obtained by these sensors and respond to a wide variety of scenarios. Machine learning models are trained with data that originates from simulation and real-life situations that we experience on the road. To create robust and reliable models, a large and diverse training dataset is required. The process of collecting this dataset is time-consuming and expensive. The existence of unpredictable and dangerous scenarios makes the task even more difficult. The solution is data augmentation techniques. Data augmentation allows us to increase the quantity and diversity of data we observe without additional collecting or labelling costs. The principle behind data augmentation is simple. It consists of artificially augmenting the data set using techniques such as rotation, cropping, image mirroring, colour shifting, etc. that transform the data without changing its meaning [167].

Generative Adversarial Network (GAN) is a machine learning model designed by Ian Goodfellow and his colleagues in 2014. The basic idea of GAN is to simultaneously train two neural networks (generator and discriminator). Typically, run unsupervised and use a cooperative zero-sum game framework to learn. Generator is a deconvolutional neural network that is trained to produce fake samples that can be easily confused with real data. The discriminator is a convolutional neural network that is trained to distinguish real samples of the dataset from fake samples produced by the generator. Given an initial training dataset, this technique learns to generate new data with the same characteristics as that dataset. The technique has been successfully used for high-fidelity natural image synthesis, data augmentation tasks, improving image compressions, and more. GANs can be implemented, for example, to generate new photographs that look authentic to the human eye and have very realistic features. So first the GAN identifies the desired final result and collects a set of initial training data based on these parameters. This data is randomized and sent to the generator to acquire basic accuracy in producing the outputs. The generated images and actual data are subsequently fed into the discriminator. The discriminator returns a probability between 0 and 1 to represent the authenticity of each image (1 if the image matches the real image and 0 matches otherwise). These values are then manually checked for success and repeated until the desired result is achieved. As the feedback loop between adversarial networks continues, the generator will begin to produce higher quality results and the discriminator will become better at marking data that has been artificially created [181]. GANs can be implemented in several interesting tasks such as predicting subsequent video frames, generating photographs of human faces, image-to-text or text-to-image translation, creating deepfake, photos to emojis, 3D object generation, and more.

2.4 Knowledge Representation

Knowledge Representation (KR) is basically the study of how an intelligent agent's beliefs, intents, and judgments can be expressed suitably for automated reasoning. Knowledge Representation and Reasoning (KRR) stands for real-world information that a computer can understand and apply to work through challenging situations in the real world. The intelligence of humans is achieved by processes of reasoning that operate on internal representations of knowledge. In Artificial Intelligence (AI), this approach to intelligence is embodied in knowledge-based agents. The principal component is the Knowledge Base (KB), a set of sentences. A sentence is expressed in a knowledge representation language and represents some assertion about the world. When the sentence is not derived from others and is taken for given, it is called an axiom. The basic actions in the KB are TELL and ASK. Their functions are to add new sentences to the knowledge base and to query what is known, respectively [133]. It is possible to construct a knowledge-based agent based on two approaches. Declarative approach means basic knowledge about something and TeLL the agent only what it needs to know. This involves knowing that something is the case (e.g., A is the first letter of the alphabet or Lisboa is the capital of Portugal). On the other hand, the procedural approach involves knowing how to do something (e.g., ride a bike). While

declarative knowledge is conscious and can often be verbalized, procedural knowledge requires implicit learning, which a learner may not be aware of or be unable to explain. However, a successful agent often combines both declarative and procedural elements in its design.

Knowledge representation is needed to formalize knowledge and to be more precise and non-ambiguous. It is used to make knowledge “computable” and to be able to express relations, handle categorical data, perform sound reasoning and to support learning new knowledge. There are several languages capable of capturing Knowledge Representation, such as Predicate Logic (example: Horn Clause Logic), Description Logic (example: OWL, RDF, knowledge graphs) or knowledge models, Probabilistic Logic (example: Markov Logic Network, ProbLog, and Bayesian Logic), Shallow Knowledge Base [3]. Knowledge Representation languages, such as Markov Logic Network and ProbLog, have recently been developed to combine reasoning and Machine Learning into a cohesive framework, allowing a system to use reasoning where they could not operate alone.

In the context of autonomous driving, the knowledge representation allows the autonomous vehicle to understand the driving environments in order to make appropriate decisions. A good knowledge representation is one that makes the important objects and relations explicit, brings objects and relations together, suppress irrelevant information, is simple to comprehend, thorough yet succinct, and requires low time to store and retrieve [76]. There are several ways to represent knowledge, including: the natural language; rules; databases; decision trees; frames; logic; scripts; ontologies; semantic, casual and genetic networks; constraints; markov models, etc [67]. Hereafter, some of these strategies will be thoroughly detailed. A **semantic network** is a directed graph with concepts as its nodes and semantic relationships as its edges. As a result, they offer a text-based structural representation of propositions about a certain domain that is more suited for computing. Concepts are typically nouns, whereas relations are verb phrases. Semantic networks resemble another type of representation, the **frame** systems. In frame systems, concepts are drawn in boxes (frames) and relations as slots inside frames that can be filled by other frames. Frame is a representation of a complex object. It is identified by a name and consists of a set of slots. Each frame has at least one hierarchically superior frame and, therefore, constitutes a base with inheritance mechanism. Figures 2.3 and 2.4 represent examples of the semantic network and frame system forms, respectively.

animal”. Rule-based knowledge representation systems are especially suitable for reasoning about concrete and simple instance data. They are problematic if more complex and general statements about the domain are derived that do not fit within the head of a rule. They are problematic if more complex and general statements about the domain are derived [67].

Both semantic networks and rules used **logic** to obtain a precise semantics. They are vague and ambiguous without this formalization, making them problematic for computational purposes. The logical representation defines several crucial communication principles and represents a conclusion based on a number of conditions. Each sentence can be translated into logic using syntax and semantics. The most prominent and fundamental logical formalism traditionally applied for knowledge representation is first-order logic. Instead of being structureless sentential symbols, the sentences of first-order logic have the type $P(t_1, \dots, t_n)$, a predicate with one or more “arguments” [133]. The majority of natural language statements may be expressed as logical sentences about objects in the domain of interest, with an appropriate choice of predicates and function symbols. For example, for the sentence “Every car has 4 wheels”, the corresponding logical representation would be $(\forall x)(car(x) \Rightarrow fourWheeler(x))$. Due to the universal quantifier ($\forall$), the variable x in the logical formula varies over all object domains, and its reading is “everything that is a car also has four wheels”. For the sentence “Some Corvettes are cars”, the corresponding logical representation $(\exists x)(Corvette(x) \wedge car(x))$. Here, the existential quantifier ($\exists$) indicates that “there is at least one corvette that is a car”. Since axiomatization requires domain expertise in order to process knowledge meaningfully, logic is increasingly used to express knowledge [67]. All the mentioned forms, nevertheless, have advantages and disadvantages. Some are shown in Table 2.1.

Table 2.1: Advantages and disadvantages of various knowledge representation techniques [154] [76] [67] [133].

FORMS TO REPRESENT KNOWLEDGE	ADVANTAGES	DISADVANTAGES
Semantic Networks	• It is simple and comprehensible • Efficient in space requirement • Easily clusters related knowled • It is flexible and easy to visualize • It is a natural representation of knowledge • Conveys meaning in a transparent manner	• Take more computational time at runtime • Inadequate as they do not have any equivalent quantifiers • Not intelligent and depend on the creator of the system • It's difficult to express some properties like negation, disjunction, etc. • Links on objects represent only binary options • Inheritance cause problems
Frame	• It makes the programming easier by grouping the related data. easy to understand and visualize • It is very easy to add slots for new attributes and relations • It is easy to include default data and search for missing values	• The mechanism cannot be easily processed • The inference mechanism cannot be smoothly proceeded by frame representation • It has a very generalized approach
Rules	• Easy to modify and maintain often allow certainty values to be associated with rules and facts • Easy to understand (natural form of knowledge) • Easy to derive inference and explanations • Easy to combine with uncertainty	• Tend to allow only relatively simple representations of the underlying facts in the domain • Complex knowledge requires many rules • Search limitations in systems with many rules
Logic	• Helps to perform logical reasoning • Is the basis for the programming languages	• Have some restrictions and are challenging to work with • This technique may not be very natural, and inference may not be very efficient

2.5 Decision-Making

Decision-making is the process of making choices by identifying a decision, gathering information, and evaluating alternative resolutions. An effective decision-making process that organizes relevant information and clearly explains the alternatives leads to the selection of the most acceptable option. Every decision-making process contributes to a final decision, which may or may not result in action. The two main categories for learning-based decision-making for autonomous vehicles are: classic methods and learning-based methods. Classical methods can be divided into three categories: rule-based methods, optimization methods and probabilistic methods. Learning-based methods, on the other hand, refer to the application of artificial intelligence to make decisions. Statistical learning methods, deep learning methods, and reinforcement learning methods are the three types of learning-based methodologies.

Rule-based methods require a rules database built from a variety of traffic laws, experiences, and driving knowledge. Logic rules with probability and/or fuzziness are used to represent and manage decision knowledge. Due to their transparency and understandability, rule-based techniques are promising solutions [5]. The Finite State Machine (FSM) method is the most well-known of rule-based approaches. In response to some inputs, FSM is a mathematical model of computation that can transition from one state to another. Some vehicles, like Junior [108], Bertha [183], Odin [130], and Talos [88], have adopted FSM methods. In the literature, one can also find some researches based on rule-based methods. Artuñedo, Godoy and Villagra [15] introduced a decision-making architecture that did not require detailed digital maps to relate the vehicle's location and identify various road elements. They suggest using a hybrid flow chart to generate decision results based on a sequence of observable environmental circumstances. In turn, Chang et al. [43] provides a decision framework for vehicle steering control called "multi-point turn". The research provides functions for determining the most viable turning manoeuvre based on steering width minimization rules.

Optimization methods usually rely on a reward function to generate decision results. One of the most widely used methods in the literature is Model Predictive Control (MPC). The study of Bey et al. [28] is an example of how this strategy can be applied. MPC was used to control the entire traffic situation, and the repercussions of the ego's actions for other vehicles are estimated by conscious prediction of the interaction. An optimal solution for the entire traffic scenario is defined using a cost function. The behaviour of the other vehicles was discovered to be influenced indirectly by the ego's behaviour. Game-Theory is another optimization method for solving decision-making problems. It is originally assumed that all agents will choose the "best strategy", and that behaviours will be created based on the strategies of other agents. Li et al. [90] presented a hierarchical reasoning method to driver behaviour in interactive traffic models based on game theory in their work. The method proposes a way for simulating the driver's interactive behaviour in traffic under the conditions in issue.

Based on mathematical probability theory, **probabilistic methods** generate behavioural results. In their study, Iberraken, Adouane and Denis [73] proposes a decision-making strategy to

deal with lane changing manoeuvres through a robust Two-Sequential Level Bayesian Decision Network. The latter is utilized to assess the driving situation, make decisions, and ensure that the manoeuvres executed are safe. To quantify risk and ensure the creation of safe probabilistic behaviours, a combination of Extended Time-To-Collision and Dynamic Predicted Inter-Distance Profile metrics were utilized.

Statistical learning-based methods give autonomous vehicles human-like decision-making capabilities. This strategy, which includes systems like Support Vector Machines (SVM), Ada Boost, and others, necessitates a significant amount of data. An SVM is trained by Vallon et al. [161] to allow the car to safely and autonomously change lanes without the need for driver assistance (e.g., activating turn signals). Subjective rules, functions of the positions and relative speeds of neighbouring vehicles, are used in the algorithm. They've also added a MPC (model predictive control) to produce a more customized autonomous lane change experience that meets safety and comfort requirements.

Deep learning-based methods use neural network structures to learn data features and generate classification or regression results. The main difference between deep learning methods and traditional machine learning methods is that the first one uses neural network structures to learn data features and generate classification or regression results. Because of the advantages of deep learning approaches in image processing, end-to-end systems have been built to achieve decision-making for autonomous cars. Sensor data is typically used as input, and trained neural networks are used to create low-level control commands.

The **reinforcement learning-based method** is currently one of the most popular decision-making learning methods. The purpose of reinforcement learning is to learn strategies for maximizing returns by experimenting with various actions. The agents' behaviours can then be modified based on the reward functions.

In Table 2.2 are gathered the characteristics of each method mentioned above.

Table 2.2: Characteristic of different methods for decision-making. Adapted from [93].

Methods		Pros	Cons
Classical Methods	Rule-based Methods	• Strong interpretability and adjustability • Strong feasibility of implementation since its low requirements for hardware • Good decision-making breadth	• Difficult to handle complex driving conditions since the lack of decision-making depth • Poor robustness for dynamic driving environment
	Optimization Methods	• Optimized decisions can be generated • Interaction between different traffic participants can be better modeled	• The assumption of "optimal strategy" for agents is often inconsistent with practical applications
	Probabilistic Methods	• Convenient to combine with other types of methods	• Low computational efficiency and difficult to generate optimal decision in complex environment
Learning-Based Methods	Statistic Learning-Based Methods	• Good versatility • Suitable for simple scenarios with sufficient environmental information	• Requirement for plenty of training datasets • Low decision-making accuracy
	Deep Learning-Based Method	• High decision-making accuracy for specific scenarios • End-to-end system ensures the full utilize of environmental information	• Poor universality of algorithms in dynamic scenario. • Requirement for plenty of training datasets thus quality of the datasets will greatly influence the effect of algorithm
	Reinforcement Learning-Based Methods	• Better modeling of uncertain and dynamic environments • Flexible framework of algorithms with high expandability	• Greatly depends on the establish of reward function • Poor stability, over-fitting in DRL methods

2.6 Feature Engineering

Feature engineering refers to the preprocessing step that transforms raw data into features that can be used in algorithms. The processes of creating, transforming, extracting, and selecting features, also known as variables, are involved. In short, feature engineering is the process of transforming raw observations into desired features using statistical or machine learning methods. By applying the feature engineering on the same model there is a chance to increase the performance from 70% to more [59].

In more detail, the processes involved in feature engineering are:

1. **Data Preparation:** is the pre-processing step that involves the manipulation and consolidation of raw data from different sources into a standardized format. It may result in data augmentation, cleaning, merging, ingesting, among others.
2. **Feature Creation:** Feature creation is a manual process where the variables that are most useful for the model are created.
3. **Transformation:** involves manipulating the variables to improve the model's performance. This process transforms the representation of the variables to ensure, for example, that they are at the same scale or within an acceptable range for the model.
4. **Feature Extraction:** is the process of automatically creating new variables from raw data. This procedure detects important data and decreases the data volume to a more manageable

size for the algorithm.

5. **Feature Selection/Exploratory Analysis:** Various features are analysed and ranked using feature selection algorithms. Identifying which features are unnecessary or redundant and should be deleted, as well as which should be emphasized. Data visualizations are frequently used to better comprehend how to alter data sources and identify mistakes.
6. **Benchmark:** is a trustworthy and simple-to-understand approach. It's frequently used to compare the results of different machine learning models. Benchmarking is used to minimize a model's error rate and increase predictability.

Below are some techniques used in feature engineering.

1. **Imputation:** missing values are frequently discovered during the data preparation stage. This omission could be due to human error, data flow interruptions, privacy concerns, and so on. Imputation's purpose is to deal with missing values and limit their impact on the learning models' performance. Numerical or categorical imputation is possible. When some information is missing, numerical imputation is used to fill in the gaps, such as replacing missing values with 0. To deal with categorical/non-numerical columns, categorical imputation is employed. Missing values can be filled in by using the greatest value in the column, computing the average, or labelling a category as 'Other'.
2. **Handling Outliers:** an outlier is an observation that differs significantly from the others in the series or is inconsistent. Outlier removal from a data collection is a common technique for obtaining a more accurate representation of the data. Outliers can easily be replaced or eliminated [68].
3. **Log Transform:** is a data transformation method in which each variable x is replaced with a $\log(x)$. When the original continuous data does not follow the bell curve, we can log convert it to make it as "normal" as feasible, increasing the validity of the statistical analysis results. The log transformation, in other words, decreases or eliminates the skewness of our original data.
4. **One-hot encoding:** this technique sets 0 or 1 to several flag columns based on the values in a column. The link between the grouped and encoded columns is represented by these binary values. This approach converts categorical data to a numerical representation, allowing categorical data to be grouped without losing any information.
5. **Scaling:** scaling of features is one of the most important problems to solve. After a scaling process, the continuous features become identical in terms of the range. Normalization and standardization are two different ways of scaling. Normalization scales all values in a fixed range between 0 and 1. Standardization typically means scaling data to have a mean of 0 and a standard deviation of 1 (unit variance). If the standard deviation of features is different, their range also would differ from each other. Both techniques reduce the effect of outliers in the features.

6. **Extracting Date:** date columns can provide useful information, but they are challenging to parse by algorithms. This could be related to the various forms they use (for example, 01-01-2021 or 1/1/2021). Extracting the date portions for different columns (month, year, day, etc.) or extracting the time interval between the current date and the columns in terms of years, months, days, etc. are possible solutions.

2.7 Evaluation Metrics for Classification

Evaluating a model is an essential part of ensuring the effectiveness of a machine learning model. Different metrics are used depending on the type of problem we want to solve. The process of building a model involves getting feedback from the metrics that evaluate its performance to subsequently make improvements to the model until it reaches the desirable accuracy. For classification tasks, among the many metrics used are the confusion matrix, F1 score and AUC-ROC. A confusion matrix is an $N \times N$ matrix, where N is the number of classes being predicted. This includes true positive values (TP), false negative values (FN), false positive values (FP), and true negative values (TN). The true positives correspond to the instances where the actual value and the predicted value are the same, i.e., the diagonal of the matrix. False negatives happen when the model makes a wrong prediction and considers the positive actual value as negative [66]. False positives mean that the model incorrectly predicts an outcome as positive. True negatives are the outcomes where the model correctly predicts the negative class. Metrics like accuracy, precision, recall, and f1-score were derived to carefully evaluate model performance. Accuracy (2.1) is a metric that generally describes how the model performs across all classes. In terms of the predicted positives, precision (2.2) shows how accurate/accurate the model is, or how many of them are actually positive. A well-known example for understanding precision is in the case of a model for classifying emails as spam. If precision has a low value, an email that is not spam (real negative) is frequently mistakenly identified as spam (false positive), which might cause the user to miss crucial emails. The recall (2.3) metric is known as the true positive rate or sensitivity, which calculates how many true positives the model captures by labelling as positive (true positive). The F1-score (2.4), in turn, combines the precision and recall of a classifier into a single metric by taking its harmonic mean. The 'macro F1' metric is frequently used to evaluate binary, multi-class and multi-label classification problems. Macro averaging is perhaps the most straightforward among the numerous averaging methods. The macro-averaged F1-score (or macro F1-score) (2.5) is computed using the arithmetic mean of all the per-class F1-scores [119]. Each metric is computed from the equations below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.3)$$

$$Precision = \frac{TP}{TP + FP} \quad (2.2) \quad F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2.4)$$

$$MacroF1 - score = 2 * \frac{MacroAveragePrecision * MacroAverageRecall}{MacroAveragePrecision + MacroAverageRecall} \quad (2.5)$$

Measuring the performance of the classification model can also be done from an Area Under the Curve (AUC) - Receiver Operating Characteristic (ROC) curve. This is considered one of the most important evaluation metrics to verify or visualize the performance of a multi-class classification problem. AUC-ROC curve is a measure in various threshold settings, where ROC is a probability curve and AUC represents the degree or measure of separability. The metric indicates how well the model is at distinguishing between classes. The larger the area under the curve, the better the model is at predicting 0 classes as 0 and 1 classes as 1. The ROC curve is plotted with TPR (True Positive Rate – 2.6) versus FPR (False Positive Rate – 2.8) where TPR is on the y-axis and FPR is on the x-axis. TPR is also known as Recall or Sensitivity and FPR as 1 – Specificity (True Negative Rate). An excellent model has an AUC value close to 1, which means that it has a good measure of separability. If AUC has a value of 0.5, it means that the model has no ability to separate classes. Sensitivity and Specificity are inversely proportional to each other. The corresponding formulas are entered below.

$$TPR/Recall/Sensitivity = \frac{TP}{TP + FN} \quad (2.6)$$

$$Specificity = \frac{TN}{TN + FP} \quad (2.7)$$

$$FPR = 1 - Specificity = \frac{FP}{TN + FP} \quad (2.8)$$

Chapter 3

Related Work

This chapter summarizes the current state of autonomous driving and brings together key research from renowned authors in the field. In a more concrete overview of decision-making, the perception process involved is defined, and some sensors used in the process are compared. The main datasets are also collected and compared, as well as simulators for generating synthetic data in the context of autonomous driving. Subsequently, the main factors influencing the scene's drivability collected by several authors in their researches are presented. Some studies are presented, along with the factors and methodologies utilized in each, with the focus now on the overtaking manoeuvre. Finally, methodologies, algorithms, and models to describe the knowledge, the inference, the reasoning, and support and evaluate decision-making provided in the literature are presented.

An overview of the mentioned sections is listed below:

- **Current Autonomous Driving Panorama:** Current state of autonomous driving, Section 3.1.
- **Inputs for Decision-making:**
 - **Perception:** Description of the steps of the environmental perception process, Subsection 3.2.1.
 - **Datasets and Simulators:** Collection and comparison of the main datasets and simulators used in autonomous driving, Subsection 3.2.2.
 - **Driveability Factors:** Survey of the factors that influence the scene's driveability, Subsection 3.2.3.
- **Decision-Making in Overtaking:** Overview of the actions involved during the overtaking manoeuvre and the factors considered in various researches, Section 3.3.
- **Knowledge Representation:** Presentation of methodologies, algorithms, and models for describing knowledge and support and evaluate decision-making provided in the literature, Section 3.4.

- **Inference, Reasoning and Learning:**

- **Inference Methods:** Description of some strategies based on inference algorithms used in several studies, Subsection 3.5.1.
- **Learning to Make Decisions:** Presentation of some systems with methods based on automatic learning of concepts and semantic representations that assist decision-making, Subsection 3.5.2.

Autonomous driving is a widely discussed topic nowadays. There are several papers in the literature that overview the myriad possibilities and problems that drive the impact of this field of research. Autonomous driving as we know it today is based on comfort, safety and promises to revolutionize transportation services. According to the World Health Organization, in the UN Global Road Safety Week 2021 report, every year more than 1.35 million people are killed in road accidents worldwide, which means almost 700 deaths on the roads every day [141]. In the recent past (2007–2014), of the total reported crashes, nearly 37% occurred on national highways, mainly two-lane two-way with mixed traffic environments [98]. In order to decrease these numbers, many countries develop their road safety plans based on the "Vision Zero" system. The term was conceived in Sweden in 1997, and it can be summarized in one sentence: No loss of life is acceptable [157]. The mistakes people make can be prevented by safety measures developed in projects such as *Visão Zero 2030* in Portugal by the *Autoridade Nacional de Segurança Rodoviária* [164].

3.1 Current State of Autonomous Driving

The current worldwide panorama of competitors in the autonomous driving market is vast and complete, especially involving companies such as: Waymo/Google/Alphabet, Cruise, Mobileye, Apollo, Baidu, Voyage, Aurora, Wayve, Tesla, Apple, Cisco, Aptiv, Alibaba, Drive. ai, Intel Corporation, Daimler/Mercedes-Benz, Audi, BMW, Ford Motors Corporation, General Motors Company, Honda, Hyundai Motor Company, Huawei, NIO, SenseTime, Uber ATG, Zoox, Samsung, Qualcomm, Jaguar Land Rover, PSA Group, Toyota Motor Corporation, AEye, Magna, AutoX, lyft, navya, Valeo, Continental, Denso, HERE, and others. There are many advances in this area thanks to the investments and several studies carried out by all these companies using different methodologies, strategies, and experiments. Some recent examples are: Amazon announced an investment of 700 million Euros in Rivian, a direct competitor of Tesla [7]; Audi established a partnership with NVIDIA [112]; BMW announced a partnership with Aurora and Apollo [31]; among others.

According to a McKinsey&Company's analysis, up to 15% of new cars sold in 2030 could be fully autonomous meaning that possibly 90 percentage of the accidents can be potentially decreased when autonomous vehicles are deployed [63][27]. However, nowadays, although there are no completely autonomous cars, there are many features present that help and alert the

driver and enhance the driving experience. While some systems already allow the driver to take their hands off the wheel, in certain situations, none yet allow them to safely take their eyes off the road. The systems that are currently implemented are described in Table 3.1.

Table 3.1: Currently systems implemented in autonomous vehicles [19] [33] [30]. Comfort and Safety are represented on a scale of 1 to 5: 1 is (-), 2 is (-), 3 is (+), 4 and 5 are (++).

System	Description	SAE Level	Comfort	Safety
Hill Start Assist	Makes the rear axle secure the vehicle for a short time when the driver disengages the brake.	0	++	-
Road Sign Recognition	Recognizes traffic signs, road lines, and other road elements, adapts driving while displaying all the information directly on the car's screen.	0	++	+
Emergency Brake Assist	Brings the vehicle to a complete stop before it results in a crash with an obstruction, a person or another vehicle.	0	-	++
Steering and Lane Guidance Assist	Can help the driver to drive the vehicle in the middle of the lane and therefore relieve him from steering.	1	++	++
Cruise Control	The system constantly maintains the driver chosen speed.	1	++	-
Adaptive Cruise Control	Adjusts the vehicle speed to match the traffic flow and/or maintain a pre-set distance to the vehicle in front.	1	++	-
Lane-change Warning	Alert the driver if there is any danger of overtaking monitoring the area beside and behind the car, and covering the notorious blind spot.	2	+	++
Parking Assist	Support the driver when manoeuvring or parking and help protect him from parking damage.	3	++	-
Driving Pilot	The vehicle should be able to take over driving completely so the driver can take their hands off the wheel for a certain amount of time. However, the driver must always be capable of resuming driving within a certain amount of time when requested to do so.	3	++	+

Table 3.1 continued from previous page

System	Description	SAE Level	Comfort	Safety
Valet Parking	Some companies have developed some systems like “Automated valet parking” from Bosch or “Intelligent Park Pilot” from Mercedes [105]. The system takes the hassle out of searching for a parking spot but also handles the actual task of parking the car in the parking garage.	4	++	+

But there are still many challenges to face before autonomous driving can gain the trust of road users and achieve the desired levels of safety and comfort. In their study, Liu et al. [92] conclude that while humans maintain an advantage in perceiving and sensing the environment, a combination of sensors can do a better job, especially in adverse weather or low lighting conditions. The following challenges are also mentioned: interaction of road users with machines; multisensory data synchronization – handle a variety of data sources and synchronize them; energy consumption – it’s a challenge due to the amount of sensors and computing devices implemented in the vehicle; data protection – the data has to be protected in order not to be vulnerable to cyberattacks; scarcity of labelled data – companies holding this data are the ones that are one step ahead in the race to autonomous driving; accumulate and hold massive amounts of driving data for application of Machine/Deep Learning technologies – only then can knowledge and algorithms be scaled with the necessary safety [92]. Due to all the challenges and issues that autonomous driving cannot yet address, user uncertainty in relying on this technology is still significant. The latest AAA annual survey on automated vehicles shows that just 14% of drivers would trust riding in a self-driving car. 54% claim to be afraid to drive an autonomous vehicle and 32% are not sure about the subject [1].

Many studies are conducted in order to solve the issues described above. Studies that focus on understanding driver behaviour and its influence on the road environment. According to Basu et al. [25], the authors realize that over 80% of users preferred the style that they thought was their own, but many times they were incorrect in identifying their own style. In the literature we can also find algorithms that adapt the autonomous car behaviour depending on what other drivers are doing [87] or a dynamic system between an autonomous car and a human driver [135]. In the latter, it is proved that autonomous car’s actions will actually affect what other cars will do in response. The authors model these consequences by approximating the human as an optimal planner, with a reward function that they acquire through Inverse Reinforcement Learning which is the problem of inferring the reward function of an agent, given its policy or observed behaviour [9]. Due to the danger present and the complexity of the driving scenarios, one of the solutions is to create middle-level input and output representations which allow easy mixing of real and simulated data [21]. In fact, the authors train a policy for autonomous driving via imitation learning and augment the imitation loss with additional

losses that penalize undesirable events and encourage progress, using the ChauffeurNet model (a Recurrent Neural Network (RNN)) to simulate complex traffic situations.

3.2 Inputs for Decision-Making

Autonomous vehicles are systems that combine 4 modules: perception, decision-making, planning, and control. The decision-making module is said to be the brain of self-driving cars. This procedure involves obtaining information from the perception module and then makes corresponding decisions for the planning and the control modules [151]. Due to the variety of situations that can occur while driving, the challenges of decision-making are many and complex. Proof of this is the critical difference between level 2 and levels 3/4/5 on the SAE scale of autonomy levels. The difference consists in the vehicle's ability to make decisions such as overtaking or changing lane on its own [94]. The autonomous vehicle's decision-making mechanism consists of the transition from perceiving world surrounding the vehicle to the motion planning system. In general, environmental cues and the status of the ego vehicle are inputs to a decision-making system, and the outputs are strategies that include behaviours and control commands that are fed into the motion planning system. Sensors placed in automobiles capture data from the surrounding environment, which is subsequently processed to provide perceptual outputs. The ego vehicle state is a representation of the ego's current location and movement. Information at the lane level can also be used to enhance the perception system. Outputs, in turn, are represented by behaviors such as overtaking, lane changing, among others, and control commands include acceleration, angular speed, etc [93].

3.2.1 Perception

Perception is the base and the vital part of all autonomous driving processes [162]. It mimics the human sensory system by analysing and understanding the surrounding environment. The outcomes of perception influence all the other areas. For instance, like it happened in 2018 when an Uber self-driving car killed a pedestrian holding a bike while crossing an urban street in the nighttime. According to a report by the National Highway Traffic Safety Administration (NHTSA), Uber's sensors were able to detect the pedestrian, but only identified them as an "unknown object" [107]. This accident exposes the need for more robust perception algorithms that are able to deal with multiple driving situations. Nowadays, perceptions algorithms are very complex and present good results in "ideal" street conditions, *i.e.*, streets with excellent road markings, flat and weakly curved road and perfect weather conditions. However, real-world situations are unpredictable, complex, and even dangerous. According to the literature, there are still a lot of challenges in this knowledge area to obtain an efficient and trusted perception, namely perception in adverse weather and poor light conditions, perception in complex environments and high velocities, real-time answers, security metrics in case of sensor failure, and others. Hazar Chaabani et al. focused their study on the problems caused by the lack of visibility in

foggy conditions. They present a neural network approach for estimating visibility distances and evaluate the solution using a diverse set of images under various fog density scenarios. The results suggested better performance compared to classical methods such as Linear Regression or Support Vector Machine [42]. Another highly respected work is the one conducted by Busch and Debes [35], where the authors used a wavelet transform (wave-like oscillation with an amplitude that begins at zero, increases or decreases, and then returns to zero one or more times) for estimating the distance of visibility in fog conditions. The physical and probabilistic models allowed them to detect the highest edge of the image with a contrast over 5%.

The perception of the environment can be split into 3 steps: collecting, filtering and processing the raw data obtained from the sensors with information that could come from the communications, the infrastructure and/or other vehicles. The perception consists of estimating the attributes of the 5 key components of the road scene. These 5 key components are:

- **obstacles:** objects that can be found in the surrounding area of an ego vehicle. The objects can be static (e.g. a hole) or dynamic (other vehicles).
- **road:** attributes that define the road surface (lanes, road markings, road states and types).
- **ego vehicle:** position and dynamic state.
- **environment:** weather conditions and other road-side components like vertical road signs or off-road objects.
- **driver:** current state of the driver (vigilance, drowsiness, etc.).

For driving automation, only the first four components are required. It is necessary to detect obstacles to avoid collisions and to have access to the entire road area surrounding the ego vehicle. By defining the road surface, we can obtain the number of lanes, identify which lane ego vehicle is in, and recognize the road markings [162]. The study of the impact of adverse weather conditions for autonomous driving has been getting more attention recently. One example is the study conducted by Zang et al. [176] that presents the effect of weather on different sensors used in autonomous vehicles and shows that the detection range of radar can be reduced by up to 45% under heavy precipitation conditions.

One of the most complicated tasks for perception is the detection and classification of occluded objects. Overtaking, for example, can be a challenging task when the view of the road is obstructed by a truck that makes it impossible to see through it. Olaverri Monreal et al. [117] and Barnum et al. [23] show different approaches to the topic. In the first, the authors propose an STS (See-Through System) system that is based on VANET (Ad hoc Networks) and video streaming technology. This system increases the driver's visibility and supports his decision to overtake in challenging situations. In the second, a 2D projective invariant is used to capture information about occluded objects (which may be moving). There have been other more recent studies with a Game Theory-Based approach such as the study conducted by Zhang and Fisac [179] which introduces a novel analytical approach that posits safe trajectory planning under

occlusions as a hybrid zero-sum dynamic game between the autonomous vehicle (evader) and an initially hidden traffic participant (pursuer). The analysis produces optimal strategies for both players that avoid collisions and support decision-making.

Autonomous driving is receiving increasing attention, with more and more resources becoming available to enable safe, reliable, and efficient automated mobility in complex and uncontrolled environments. Signal processing is a critical component of autonomous driving. Some required technologies include affordable sensing platforms that can acquire data under varying environmental conditions, reliable simultaneous localization and mapping, machine learning that can effectively handle real-world conditions and unforeseen events, complex algorithms to bring more effective classification and decision-making, efficient real-time performance, resilient and robust platforms that can withstand breaches and adversarial attacks, and end-to-end system integration of detection, Machine Learning signal processing, and control.

The data processing is structured in five levels. The first level corresponds to the raw data collected by the sensors (laser impacts, RADAR frames, distances, speeds, images, accelerations, angles). The second level focuses on filtering, spatial and temporal alignment, modelling of inaccuracy, uncertainty, and reliability. The third level corresponds to clustering, feature extraction, object detection and modelling. In the next level, more detailed information is obtained, such as the shape, colour and positioning of the object. At the last level, the interactions between objects are treated in order to build a more synthetic and enriched representation. The temporal relationships between objects make it possible to identify some behaviours and predict trajectories. The results of the last level can be used as inputs for the decision level (decision-making, risk assessment or trajectory generation) [162]. The detection of obstacles of various shapes, sizes, and orientations is still a challenge due to the lack of information in the literature and the number of labelled and tagged datasets is still scarce. The system for recognizing the environment around the vehicle must be done robustly, accurately and at a 360° angle by a combination of sensors. Several companies like Audi, Bosch Group, Uber, and Google/Waymo consider the LiDAR as a pillar sensor during the perception phase, while others like Tesla and Nissan do not consider it essential possibly because of cost-performance trade-off since the cost of placing a single LiDAR device on a car is somewhere around \$10,000 [173][144]. These values, however, are dependent on the sensor's robustness and range from \$1,000 to \$23,000 [131] [132]. Instead of LiDAR, they prefer to use radar sensors and cameras. On the other hand, the Radar sensor has its value recognized for being particularly useful in environments with adverse weather conditions (fog, rain, snow). However, to obtain 360° coverage would require many radar sensors, as demonstrated in Aptiv's nuScenes dataset which uses 5 Radar units in its setup and still fails to achieve 360° mapping at close range [37].

A comparison of some sensors is shown in Table 3.2.

Table 3.2: Comparison of sensors used for autonomous driving. Adapted from [69] and [92]. Legend: (++): perfectly adapted sensors; (+): sensors having good performances; (-): sensors potentially meeting the criteria but with possible disadvantages; (--) : sensors that can be used with adaptations and additional heavy treatments; (0): sensors unable to meet the criteria.

		Radar short range	Radar long range	LiDAR	Ultrasonic	Camera	Camera 3D	Camera IR
Range <2m (short range)		-	-	-	++	-	++	-
Range 2m - 30m (nominal range)		+	++	++	-	-	-	-
Range >30m - 150m (long range)		n.a	++	+	0	-	-	-
Field of View angle <10°		+	+	++	-	++	+	++
Field of View angle >30°		-	-	++	-	++	+	++
Angular resolution		-	-	+	-	++	+	++
Direct Speed Information		++	++	0	-	0	0	0
Night Vision		++	++	++	++	-	-	++
Sensitivity to Light		++	++	+	++	0	0	+
Cost		+	+	-	++	+	-	-
working:	under rain	++	+	-	-	-	-	-
	in fog and snow	++	++	-	+	-	-	-
	with dust	++	++	-	++	0	0	0
Application Scenarios		Object Detection Distance Estimation		Object Detection Distance Estimation Edge Detection	Object Detection Distance Estimation	Object Classification Edge Detection Lane Tracking		

3.2.2 Datasets and Simulators

To obtain robust and accurate perception algorithms, hundreds of millions of data are required. To train the algorithms, the stock of training data must be labeled ("ground truth") to ensure high training accuracy [79]. Ground truth refers to the actual values of the training set's classification, it is the manual verification that the virtual world corresponds to the real human-defined measurements [170]. The road environment as we know is quite unpredictable and some scenarios are too dangerous to be staged in the physical world (eg. a child behind the car during parking). Nevertheless, some companies like Uber, Google or Tesla usually test their vehicles in real traffic conditions. This brings disadvantages in that it is very difficult to reproduce dangerous and imprecise scenarios and makes it impossible to get the "ground truth" of obstacles, pedestrians, and other vehicles involved in the test [72]. For this reason, an alternative is to train and validate driving strategies in simulation with synthetic data. According to the Gartner 'Predict', published in the Wall Street Journal by analyst Svetlana Sicular, "By 2024, 60% of the data used for the development of AI and analytics projects will be synthetically generated" [41]. Some famous datasets are public and available for independent analysis. Public datasets have some associated disadvantages: they are either too generic for perception tasks or too task-specific [37]; they are extremely focused on the classes: car, pedestrian and cyclist [156]; they are "public" only for research purposes, and cannot be used in the industrialization process of Perception algorithms; most do not present data collected in different weather conditions; They are small, which prevents them from obtaining the minimum quantity of training data required to train neural networks to be competent in Perception tasks. Virtual KITTI, SYNTHIA, Synscapes,

Sintel and CARLA are examples of synthetic data datasets. On the other hand, some datasets that use real data are BDD, KITTI, ScanNet and NuScenes [113][104]. The Middlebury flow dataset contains both real and synthetic scenes [20]. Table 3.3 is a comparison of the features of the main datasets for autonomous driving.

Table 3.3: Comparison of main datasets’ features. Adapted from [37] and [86].

Legend: (-) indicates that no information is provided.

Dataset	Year	# Scenes	Size (h)	# RGB Images	# pc LiDaR	# pc Radar	# ann. frames	# 3D boxes	# Map Layers	# Classes	Night	Rain	Snow	Dawn	Locations
Kitti	2012	22	1.5	15k	15k	0	15k	200k	0	8	No	No	No	No	Karlsruhe
Cityscapes	2016	-	-	25k	0	0	25k	0	0	30	No	No	No	No	50 cities
CamVid	2008	4	0.4	18k	0	0	700	0	0	32	No	No	No	No	Cambridge
Vistas	2017	-	-	25k	0	0	25k	0	0	152	Yes	Yes	Yes	Yes	6 continents
Waymo Open	2019	1k	5.5	1M	200k	0	200k	12M	0	4	Yes	Yes	No	Yes	USA
Lyft Level 5	2019	366	2.5	323k	46k	0	46k	1.3M	7	9	No	No	No	No	Palo Alto
nuScenes	2019	1k	5.5	1.4M	400k	1.3M	40k	1.4M	11	23	Yes	Yes	No	Yes	Boston, Singapura
ApolloScape	2018	147k	100	144k	0	0	144k	70k	0	8-35	Yes	Yes	Yes	No	China
KAIST	2018	-	-	8.9k	8.9k	0	8.9k	0	0	3	Yes	No	No	Yes	Seoul
BDD100K	2017	100k	1k	100M	0	0	100k	0	0	10	Yes	Yes	Yes	Yes	NY, San Francisco
H3D	2019	160	0.77	83k	27k	0	27k	1.1M	0	8	No	No	No	No	San Francisco

	Sensors					Traffic					Annotation				
DATASET	Video	LiDaR	GPS	Thermal Camera	Radar	Urban	Highway	Residential	Rural	Campus	Pixel	Instance	Point Cloud	2D	3D
Kitti	x	x	x			x	x	x	x	x				x	x
Cityscapes	x		x			x					x	x			
CamVid	x					x					x				
Vistas	x					x	x	x	x		x	x			
Waymo Open	x	x				x	x	x	x					x	x
Lyft Level 5	x	x	x			x		x						x	x
nuScenes	x	x	x		x	x								x	x
ApolloScape	x	x	x			x	x	x	x		x	x	x		
KAIST	x	x	x	x		x		x		x				x	
BDD100K	x		x			x	x	x	x		x	x			
H3D	x	x	x			x									x

Most datasets focus on 2D annotations for RGB camera images. CamVid, Cityscapes, D2-City, BDD100k and Apolloscape have released datasets with segmentation masks (image processing method in which a small 'image piece' is defined and used to modify a larger image). Vistas, D2-City and BDD100k complement their datasets with images taken during different weather conditions and illumination. Multimodal datasets (consisting of images, sensor data, and GPS data) are expensive and difficult to collect due to the difficulty in synchronizing and calibrating sensors. KITTI was the pioneer in multimodal datasets combining dense point clouds provided by the LiDAR sensor with front-facing stereo images and GPS/IMU data. It was a great help in advancing 3D object detection. Thereafter, multimodal dataset KAIST uses colour and thermal cameras and a beam splitter to capture the aligned multispectral (RGB colour + Thermal) images.

Thus, the dataset is able to provide data during the night capturing various regular traffic scenes, but the annotations are in 2D [37]. Others notable datasets are: LiDAR-Video Driving benchmark dataset which is among the first attempts to utilize point clouds to help driving policy learning and provide driving behaviour labels [47]; two 3D outdoor datasets presented by Hojung Jung et al. for semantic place categorization labels: forest, coast, residential area, urban area and indoor/outdoor parking lot [78]; and Málaga Urban Dataset gathered entirely in urban scenarios providing raw data without semantic labels [29].

A lot of studies that discuss the pros and cons for synthetic and real data can be found in the literature. In 2018, Tremblay et al. [159] present a system for training deep neural networks for object detection using synthetic images. In order to force the network into learning only the essential features of the task, they use domain randomization for car detection. The idea was to effectively abandon photorealism in the creation of the synthetic dataset. This study also proved that using real images, the accuracy of the models is improved. In contrast, Alhaija et al. [6] propose an alternative paradigm combining real and synthetic data for learning semantic instance segmentation and object detection models. The authors conclude that cluttering the images with too many objects reduces the model performance, and models trained on augmented imagery generalize better than those trained on synthetic data. Grand Theft Auto (GTA) game is used in the article by Tremblay et al. [159] to propose a fast synthetic data generation approach. The authors demonstrate that a state-of-the-art architecture, which is trained only using synthetic annotations, performs better than the identical architecture trained on human annotated real-world data. In context of LiDAR sensor, Wu et al. [171] employ GTA combined with KITTI dataset in order to create a synthetic LiDAR dataset to train a deep model and synthesize large amounts of realistic training data. Thereafter, Fang et al. [58] propose a novel LiDAR simulator that augments real point cloud with synthetic obstacles. First, using a LiDAR sensor and a camera, was created a real background environment dataset and this data is then augmented with synthetic 3D objects. The authors conclude that mixing real and simulated data can achieve over 95% accuracy.

With a driving simulator, testing realistic scenarios in a virtual way is possible. Car models, sensors, roads, and traffic can be created and modelled. However, companies do not share the same simulation systems. Some leading companies like Alphabet's Waymo, GM Cruise, Tesla Autopilot, Aurora, BMW and Bosch are developing their own simulators. Waymo has developed a simulation platform called Carcraft. The platform recorded around 2.5 billion synthetic miles in 2016 which allowed it to test dangerous scenarios such as car crashes without human victims. Waymo's cars are equipped with LiDAR, radar and camera sensors that generate images that are often difficult to manually label due to the difficulty of the images interpretation [166]. The Matrix is the simulation platform of GM Cruise (subsidiary of General Motors). The platform is built on Unreal Engine and can collect 200,000 virtual miles per day. It contains full-3D graphics and it is estimated that each car can generate 300 terabytes of synthetic data and use 5,000 GPUs and 300,000 processor cores daily. The Matrix also handle LiDAR, radar, and video camera data from 30,000 virtual cars [65]. Tesla is known for relying heavily on video cameras and avoiding the LiDAR sensor by focusing heavily on photorealism. Tesla collects data from vehicle

owners to compare real driver actions with the projected trajectory of the autonomous car. With this, it becomes possible to train and improve its autonomous driving algorithms [18]. Aurora, backed by Amazon and Sequoia, built their platform called Offline Executor, focusing heavily on synthetic data. This platform was developed from the ground up and is dedicated to reproducing short, specific situations like making a dangerous right turn [74]. BMW has developed the BMW Motorsport simulator. The system has a processing power of 100 teraflops, i.e. 100 billion calculations are possible every second. BMW Motorsport is developing the simulator continuously in parallel for ongoing use, in addition to the hardware, primarily in the areas of software and modelling [85]. Most of these are proprietary tools, however, there are many open-source simulators available as well. There are a wide range of simulators for generating synthetic data for autonomous driving.

Some of popular simulators are: CarSim, PreScan, Gazebo, Carla and LGSVL. They differentiate themselves by their performance in features such as perception, localization, vehicle control, and the creation of dynamic 3D virtual environments. A simulator is expected to come as close as possible to reality. Consequently, this requires highly detailed 3D virtual environments and very precise but simple calculations about the physics of the car. In short, a trade-off is needed between the realism of a simulation, and the simplicity of its calculations. Despite the rapid advancement of autonomous driving, we are still a long way from having the perfect simulator. The more features and realistic they are, the more resources they will require to complete their computations in a suitable period of time [60]. The state-of-the-art simulators CARLA and LGSVL meet this trade-off. Furthermore, Gazebo is a popular robotic 3D simulator, it is inefficient in terms of the time it takes to create a 3D scene in the simulation environment [84]. Simulators like MATLAB/Simulink remain vital because they provide extensive analysis of the findings through their graphing tools. CarSim, on the other hand, is highly specialized in vehicle dynamic simulations since it is backed by more exact automobile models. Table 3.4 compares the performance of the simulators depending on their requirements.

Table 3.4: Comparison of popular simulators. Adapted from [84].

Legend: Y indicates that simulator has the corresponding requirement; N indicates the opposite; (-) indicates that no information is provided.

Requirements	CARLA	CarSim	Voyage Deepdrive	rFpro	AirSim	Deepdrive	Prescan	Gazebo	LGSVL	MATLAB (Simulink)
Perception: Sensor models supported	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Perception: Support different weather conditions	Y	N	Y	Y	Y	Y	Y	N	Y	N
Camera Calibration	Y	N	N	Y	N	N	Y	N	N	Y
Path Planning	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Vehicle Control: Support for proper vehicle dynamics	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
3D Virtual Environment	Y	Y	Y	Y	Y	Y	Y	Y	Y	-
Traffic Infrastructure	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Traffic Scenario simulation: Support of different types of Dynamic objects	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
2D/3D Ground Truth	Y	N	Y	Y	Y	Y	N	Y	Y	Y
Interfaces to other software	Y	Y	-	Y	Y	-	Y	Y	Y	Y
Scalability via a server multi-client architecture	Y	Y	-	Y	Y	-	-	Y	Y	-
Open Source	Y	N	Y	Y	Y	Y	N	Y	Y	N
Well-maintained/Stable	Y	Y	-	Y	Y	-	Y	Y	Y	Y
Portability	Y	Y	-	Y	Y	Y	Y	Y	Y	Y
Flexible API	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Simulink/MATLAB is designed for simpler scenarios that require efficient computations and plot functions. This system can be used in conjunction with other simulators to create a variety of vehicle models. However, when it comes to realistic depiction of traffic conditions, obstructions, and pedestrian models, MATLAB/Simulink falls short. MATLAB has multiple examples for simulating ADAS capabilities such as adaptive cruise control, automatic emergency braking, automatic parking assist, and so on [102]. CarSim includes over 200 dynamic and static objects with independent positions and motions. There are a variety of mathematical models available for vehicle control, and users can use built-in options like CarSim commands or external models to control motion. Although CarSim has extensive documentation and numerous sample simulations, it is not an open-source simulator [40]. PreScan does an excellent job at simulating real-world environments like weather. The modelling is more efficient because of the MATLAB/Simulink interface [127]. Despite the system's high flexibility in offering the user complete control over the simulation, it comes at a significant time and effort cost. Gazebo is a famous robotic simulator. The user must construct 3D models, specify their physics, and place them in the simulation world manually while creating a simulation environment. This is not the case, for example, with the CARLA and LGSVL simulators. The user can construct plugins using Gazebo to create/modify multiple sensor models. It also includes comprehensive documentation and a number of tutorials. It's usually used in conjunction with Robot Operating System (ROS) (a suite of software libraries and tools for creating robot applications), which allows it to model nearly any form of robot [120]. CARLA and LGSVL both offer high-end simulation environments that require a GPU computing unit to run at a reasonable frame rate. Both offer a flexible API, but they differ in some ways. One distinction is that CARLA allows you to record and store simulations, which LGSVL does not. CARLA uses the Unreal Engine to generate maps, which includes automatically placing stop signs

using OpenDRIVE technology [17]. LGSVL, on the other hand, uses Unitygame, which allows the user to create a new map by manually importing the various components.

The structure of data is an intriguing topic for data organization and representation. Choosing whether to employ structured, semi-structured, or unstructured data can have a significant impact on a project's success. Fischer, Sester and Schön [61] present a research data management system that includes a structured data storage for spatio-temporal experimental data in their study. Because data must be free, accessible, interoperable, and reusable, the usage of structured data is recommended. Complicated research initiatives necessitate large data sets, which necessitate a complex infrastructure to monitor data collection, store data, and enable structured and intuitive data access.

Another notable work by Sun et al. [152] suggests a Neuro-Symbolic Program Search to improve autonomous driving system design and validates it using the CARLA simulator. NSPS is a new automated search method that synthesizes Neuro-Symbolic Programs. The authors demonstrated that their method provides more stable and safer behaviours in complex driving scenarios than other neural network and rule-based driving methods, as well as an interpretable symbolic decision process, as compared to other neural network and rule-based driving methods. The human design of rules can be replaced with NSPS, which might speed up the automated programming process. To make the decision-making policy generalizable to multiple driving conditions, driving decisions are derived based on structured inputs. The 3D coordinates of the location, accelerations, speeds, and bounding boxes of the ego vehicle and the cars nearest to it within a 70m radius are among the structured inputs.

3.2.3 Driveability Factors

Understanding what factors influence a scene's drivability is critical for analysing the variables collected during a simulation. Environmental factors like as weather, traffic flow, road quality, and road obstructions, among others, are recognized to have a significant impact on the driving environment. However, these explicit characteristics, i.e. that can be immediately observed from the environment, are insufficient for evaluating the driving environment. The implicit information that must be inferred from observation must be taken into account. According to U.S. Department of Transportation reports, Guo, Kurup and Shah [70] considered important factors for driving based on studies of drivability in other fields of transportation systems research, U.S. Department of Transportation reports, and industry standards for estimating road risk. The explicit factors considered in this study were:

- **Weather:** bad weather conditions, such as fog, rain, and wind, can limit a driver's road visibility. As a result, detecting objects and barriers can be challenging. Deep Neural Network (DNN) models, for example, have a history of misbehaving in bad weather [177].
- **Illumination:** perception is challenged by fluctuations in brightness induced by the time of day, the landscape, and directed light sources. The nighttime environment presents

extra difficulties due to low illumination, changing contrast, and less colour information. As a result, studies on nighttime data are underrepresented [110]. At night, the vehicle's headlights and taillights help drivers recognize road objects. However, other illuminant sources such as traffic lights, street lamps, and road reflector plates on ground can cause many difficulties for detecting actual vehicles [46]. However, because pedestrians and other barriers lack their own light, it becomes difficult to identify them. Training in night situations reduces the accuracy of pedestrian detectors, according to a study conducted by Neumann et al. [110].

- **Road Geometry:** it is significantly easier to drive on free ways or straight roads. Because of the high number of accidents that occur in these places, road designs such as intersections and roundabouts are extensively examined. Shirazi and Morris's [145] research examines recent studies on vehicle, driver, and pedestrian behaviour at intersections, as well as their levels of safety.
- **Road Condition:** the road's condition may be affected by its uneven surface, road damage, potholes, or construction. Since these examples are not very common, there is a lack of labelled data in this field. Construction on the road can alter the driving environment by adding traffic signs, changing the geometry of the road, and workers on the road (pedestrians). In this regard, Seo et al. [142] provide a set of computer vision methods that recognize the limits of a road work zone as well as transitory changes in driving surroundings. This restriction is important in order to determine the available and safe area for driving while avoiding potential risks.
- **Lane Marking:** the detection of lanes or roadways is made feasible by lane markings. There are several roadways that have no lines or have irregular lines. This makes detecting and delineating highways a difficult process. Some research, such as that conducted by Li et al. [91] and Bao et al. [22], show how to discover unmarked roads. These experiments aim to combine numerous inputs from cameras, infrared sensors, and LiDAR sensors.
- **Traffic Condition:** there is a distinct contrast between driving in an urban area and driving outside of an urban area. Some of the elements that distinguish the two driving environments are speed limits, traffic flows, the number of lanes, and traffic rules.
- **Static and Dynamic Objects:** one of the most researched topics in autonomous driving is object perception and detection. Existing approaches still have high error rates when it comes to finding things that are occluded by others, are small, or are difficult to recognize. These things can make driving difficult and inhibit effective planning. Pinggera et al. [126] provide a dataset with small objects and a stereo vision algorithm for reliably detecting such impediments as lost cargo from a moving vehicle. Also, Zhang and Fisac [180], as well as Olaverri Monreal et al. [117], built a framework for decision-making in driving situations with hidden agents in their research.

The actions and intentions of road users constitute implicit elements. Road users communicate with the autonomous vehicle and other road users. There are three realms to consider: the

vehicle's interior, its surrounding environment, and the interiors of other vehicles [116]. Sadigh et al. [135] conducted a relevant study on this topic that reveals how autonomous vehicles' actions affect the responses of other vehicles. Some implied factors are: **vehicle behaviour** when overtaking, lane changes, speed-driving, non-compliance with traffic laws, and other harmful vehicle behaviours are evaluated; the **behaviour of pedestrians**, which represents the most vulnerable road users. Many of the accidents involving autonomous cars also involve pedestrians. Rasouli and Tsotsos' study outlines many methodologies for studying pedestrian behaviour, as well as two approaches for predicting pedestrian intent. The first is to approach it as a dynamic object tracking issue, calculating the future trajectory of pedestrians, while the second is to approach it as a classification problem, categorizing pedestrian behaviour as "crossing" or "not crossing". Methods that rely solely on pedestrian position are prone to errors. Other criteria, like age, gender, and speed, must also be considered in order to forecast pedestrian intent and thus minimize collisions and other incidents [129]; **driver behaviour**, since their intervention is still required in automatic driving when it fails or the car is unable to make trustworthy decisions. When investigating the causes of traffic accidents, elements such as skill, intention, driving style, distraction, and others are considered [53]. Various ways for identifying factors such as driver tiredness and distraction while driving are shown in studies such as those conducted by Kang [80] and Kaplan et al. [81]. Visual elements such as face expression and eye movement are used in traditional techniques. Autonomous vehicles currently have technologies in place to recognize when a driver is tired or preoccupied. Bosch's driver drowsiness detection, for example, monitors steering movements and urges drivers to have a rest when necessary [32].

The overtaking scenarios consider factors other than those mentioned. The research of Dixit et al. [52] considers the variations in number of overtaken vehicles, the duration of the overtake, the relative velocity between concerned vehicles and the distance between concerned vehicles. Features are classified as permanent (road and lane limits), slowly changing (speed limits, road works, traffic density, etc.), and fast changing (surrounding vehicle velocity, position, heading, etc.). According to the study, two crucial parts of high-speed overtaking trajectory planning are the integration of vehicle dynamics and environmental restrictions, as well as precise information of the surrounding environment and impediments. The research also proves that autonomous cars must have a precise understanding of the surrounding environment, which is not representative of real-world driving. Another good example is the study conducted by Schubert, Schulze and Wanielik [136]. Some different features considered were lane markings, velocity and yaw rate, position and heading of the vehicle, longitudinal acceleration, distance between vehicles, time to predicted collision and deceleration to safety time. They present a system that can perceive the vehicle's environment, assess the traffic situation, and give recommendations about lane-change manoeuvres to the driver.

3.3 Decision-Making in Overtaking

Several computer models are nowadays able to outperform humans in detecting and identifying objects, both in images and videos. However, autonomous vehicles, in addition to recognizing their external surroundings, have to make decisions, which has implications regarding safety, performance, ethics, and accountability. Mistakes during decision-making can result in severe accidents. Every year, road accidents result in about 20-50 million injuries and 1.25 million deaths. Many of these accidents are due to driver misinterpretation and untimely decisions, wrong speed chosen, not being able to see through an obstacle, sudden breaks, ignorance of road conditions, adverse weather conditions, etc [125].

The overtaking manoeuvre is a way for faster drivers to continue driving at the desired speed without lagging behind slower vehicles. It brings comfort to the driver and enhances their experience on the road. However, that manoeuvre is one of the most critical actions in driving. The process involves lane-changing, acceleration and deceleration actions, calculating the distance of the oncoming vehicle and speed of overtaking and overtaken vehicles. To reduce the impact of these manoeuvres and increase driver safety, the vehicles would have built-in intelligent algorithms that consider all important aspects during decision-making. These aspects can be: calculating the proximity of other vehicles to the ego vehicle, determining whether a lane change manoeuvre can be made, and designing optimal and safe paths for the manoeuvre. Jula, Kosmatopoulos and Ioannou [77] investigated on the minimum longitudinal distances required for lane changes or merging. For a vehicle overtaking another slower vehicle in front of it, Shamir [143] described the equations of motion employed and the ideal values of the variables. Pérez et al. [124] proposed a fuzzy logic decision control system that achieves two consecutive lane changes. Karaduman et al. [82] built a Bayesian belief network to calculate the probability of crashes in a driving environment with one car in front and one behind.

Certainly the greatest difficulty in overtaking is, during a fast flow of traffic, estimating the time available to perform the manoeuvre. Driver's decisions are unpredictability, especially when there is a speed difference between fast- and slow-moving vehicles. In their study, Asaithambi and Shravani [16] considered the flying (with no other vehicles nearby, the vehicle overtakes a slow moving vehicle without having to slow down) and accelerative (the vehicle approaching a slow moving vehicle reduces its speed until it has enough space in the other lane to overtake) overtaking scenarios. The features analysed were the acceleration characteristics, speeds of the overtaking vehicles, overtaking time, overtaking distances, safe opposing gap required for overtaking, flow rates, overtaking frequencies, types of overtaking strategy, and types of overtaking and overtaken vehicles. The authors proved that the majority of vehicles are travelling with their current speed without reducing the speed during overtaking (flying performed by 62% of drivers and accelerative by 38%).

Llorca and García [96] presented models with different Passing Sight Distance values depending on the proposed situation. They observed overtaking manoeuvres on highways in Spain using 6 video cameras installed at 4 different points. A sample of 234 manoeuvres was

recorded, and the new variables introduced in the model were helpful to compare data with others existing PSD models. The variables with the strongest influence were the type and speed of the passed vehicle and the passing zone length. The observed average speed difference between passing and impeding vehicles was significantly higher than that in any model.

Vlahogianni [165] applied a classical survival analysis with data provided by a driving simulator to model overtaking manoeuvres in two lane highways. They determined that factors influencing the duration of the manoeuvre could be the gender of the driver, speed difference of the overtaking vehicle, traffic flow and speed, spacing between cars in different lanes as well as whether the driver is engaged in multiple overtakes. The results conclude that the time to complete an overtaking manoeuvre in two lanes should be described by a logistic distribution. This function is suited for repeated events and suggests that the likelihood of an overtaking manoeuvre to end is high until a certain time period and then reduces (probably due to the willingness of the driver to conduct multiple overtakes).

In 2016, Llorca and Farah [95] realized that the studies done so far did not address the validity of simulators, as their results were rarely compared to real data. So, the main objective of their study was to compare drivers' passing behaviour as observed in the field with passing behaviour in a driving simulator. For this purpose, the authors needed data describing the overtaking performance to make passing gap acceptance decisions. The variables considered were driver's age and gender, type of impeding vehicle (passenger car or truck), passing time (s), passing distance (m), average speed of impeding vehicle during the passing time (km/h), average speed of passing vehicle during passing time (km/h), speed difference (km/h), clearance between impeding and passing vehicle before passing (m), clearance between impeding and passing vehicle after passing (m), speed difference at start (km/h), time until crossing with next opposing vehicle, or safety margin (s). A disadvantage of the driving simulator was that it provided an unlimited field of view. These scenarios do not correspond to reality, where many obstacles decrease the amount of observable information. The results showed similarities between passing time and passing distance of completed manoeuvre, but drivers passed faster in the driving simulator, keeping higher clearances.

Another work that goes towards decision-making in overtaking scenarios is the work conducted by Yang et al. [174]. The authors designed a Deep Neural Network (DNN) model to make overtaking decisions for stationary vehicles and analysed the significant decision factors for these cases. They demonstrated that the factors extracted during the process of evaluating the traffic scene helped to improve the model's learning performance. The factors considered were divided into 3 categories: A – preceding vehicle, B – surrounding vehicles and C – ego vehicle. The factors included in category A were lateral distance to the right/left boundary of the road; duration of time it has been detected as stationary; velocity; acceleration; yaw angle; yaw rate; lane occupancy rate and object width/length. Category B included position, velocity, and acceleration to the nearest vehicle and number of vehicles, free space rate, spatial gaps and time/space mean speed for the remaining vehicles. Category C included relative speed, relative distance, time-to-collision/time-headway, and waiting time duration. All relevant factors were chosen

based on the general characteristics of human drivers. Although all of these factors appear to be relevant in decision-making, in order for the model to be more efficient, the most crucial factors have to be chosen. The Sequential Backward Selection (SBS) approach was used to determine the relevance of each factor. This method evaluates the learning performance of a model that takes all candidate factors as input to estimate the importance of each factor. It assesses the decrease in learning performance by eliminating a candidate. The significance of the factor that has been eliminated is determined by the degree of decrease. The method iterates the process for all factors, removing the least important one each time. The deleted factor is regarded unimportant or redundant if the drop is almost zero or minor. In short, the SBS technique determines the dominant set by analysing the relevance of the features. As a result, the lateral position of the previous vehicle was found to be the most crucial component, followed by the vehicle's waiting time to begin the overtaking manoeuvre. This indicates that the driver's decision can change over time, even though the situation remains the same. In the end, factors like time/space mean speed and lane occupancy rate were found to have no discernible effect on performance. Figure 3.1 shows the outcome of the SBS method applied to the factors in this research.

Rank	Category	Feature Description	Accuracy
1	A	Lateral Position	65.1%
2	C	Waiting Time	79.8%
3	B	Time Mean Speed	83.0%
4	B	Number of vehicles	87.0%
5	C	Distance to Preceding Vehicle	88.4%
6	A	Moving Confidence	89.1%
7	A	Current Speed	89.4%
8	B	Speed of the Closest Vehicle	89.5%
9	B	Lane Occupancy Rate	89.5%
10	B	Space Mean Speed	89.5%

Figure 3.1: Outcome of the SBS method applied to the factors in the research by Yang et al. [174].

3.4 Knowledge Representation

The previous section exposed many challenges that need to be overcome in order to achieve a good level of quality in a self-guided vehicle. One important challenge is how to structure data in order to support decision-making. Various methodologies, algorithms, and models for describing knowledge and supporting and evaluating decision-making may be found in the literature. Many studies have been conducted on knowledge representation, reasoning, and learning. Several methods of knowledge representation are used in the proposed solutions. Patron et al. [123] present a semantic world model framework for hierarchical distributed knowledge representation in autonomous underwater systems in their study. The method given demonstrates how to abstract from raw real-world data using semantic technologies. The authors Zhao et al. [182] used ontologies to represent Knowledge regarding maps, paths and driving environments in the

development of ADAS systems. The Knowledge Base created by these authors contains 36.424 RDF triples, 147 classes, 84 attributes, and 121 entities. A similar but more recent study is the one conducted by Westhofen et al. [168]. The authors show how ontologies can be used to model, formalize, and reason about criticality issues in the context of automated cars. In the research conducted by Garcia-Duran and Niepert [64], on the other hand, knowledge is represented by a novel combination of neuronal representation learning and probabilistic approaches based on expert models. The authors proposed a framework for end-to-end learning of a Knowledge Base Representation from latent, relational, and numeric features, called KBLRN. Figure 3.2 shows a small part of a KB containing relations and numerical features.

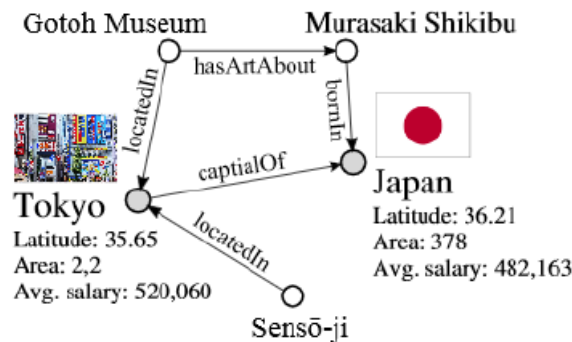


Figure 3.2: A small part of a knowledge base [64].

In turn, Zhang et al. [178] explore semantic networks and hypergraphs to model, describe and store knowledge. The construction of the hypergraph is divided into 6 steps. The first step is to determine the specific task and express the tasks and sub-task as nodes, as shown in Figure 3.3. The basic traffic behaviours can be: start, shift, vehicle safety distance, speed control, etc. The second step is to determine what solving problems each sub-task has. The vehicle safety distance of a sub-task, for example, requires the vehicle to know the speed of the current and preceding vehicles, as well as determining which safety distance level corresponds to the car and how far it will be kept. The third step is to add the rules. Using the previous example, the rule is to determine the safety distance to the preceding vehicle based on the current speed of the vehicle. Step 4 is to create metaknowledge and reference the corresponding rules. Metaknowledge is known as "knowledge about knowledge" [57]. The term is used to define elements that describe knowledge, including tags, models, and taxonomies. Step 5 is to link each task and sub-task to the associated knowledge super meta. The preliminary hypergraph build is the final phase. The hypergraph is saved, knowledge is assembled and transmitted, and knowledge reasoning is performed.

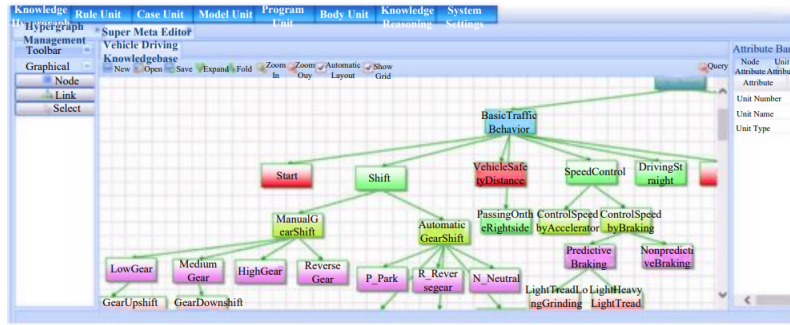


Figure 3.3: Example of hypergraph: vehicle driving knowledge base [178].

3.5 Inference, Reasoning and Learning

Knowledge representation languages such as Markov Logic Network and ProbLog have been developed to incorporate both reasoning and machine learning in a unified framework which allows a system to perform the reasoning using the language as a basis [3]. Inference methods are usually based on a sound mathematical formalism, be it deterministic or uncertain. Markov Logic Networks, for example, uses probabilities and potential calculation to score the strength of edges that connect nodes of a Markov Logic graph. ProbLog, on the other hand, performs two kinds of inference: logical and probabilistic based on a Bayesian modelling. Algorithms vary and in this section, we highlight some of them.

3.5.1 Inference Methods

The goal of reasoning is to create and improve inference and learning algorithms. Many types of reasoning are considered vital for advanced intelligent services, including spatio-temporal, causal, model-based, and uncertainty-based reasoning. To separate correlation from causation, inference methods and algorithms allow computing the effects of a certain input. As a result, they make it feasible to determine what can be extracted from the represented knowledge [49]. Karlsson [83] investigates how and what to expand Markov Logic networks (a set of logical assertions), which, when compared to Naive Bayes networks (a probabilistic classifier), might improve the resilience of an object identification system by incorporating context/contextual information. To reduce categorization mistakes over several scenes, the author introduced spatial relationships such as “isRightTo”, “isLeftTo”, and “isAbove”. Despite the fact that logic reasoning is not directly applicable to autonomous driving, research demonstrates that it can enhance categorization accuracy. Xue, Fang and Zhang [172] present a scene understanding flowchart that employs an event-based reasoning framework for autonomous driving and investigate high-level topological rules in the scene understanding problem. That flowchart is illustrated in Figure 3.4. Applying logical reasoning to structured knowledge databases is now the most extensively used and best performing method. This strategy is based on very efficient inference algorithms and

human-friendly representations. The subject of probabilistic Bayesian reasoning, and in particular the usage of graphical models, is regarded as the most flexible contemporary strategy in the case of knowledge extraction from unstructured data, where uncertainty is common. To summarize, reasoning encompasses a wide range of strategies, and it is typical to discover algorithms (and Knowledge Bases) tailored to a certain task or subject.

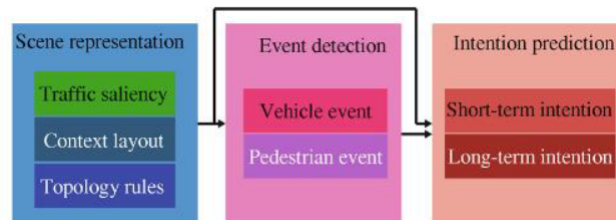


Figure 3.4: The scene understanding flowchart by event reasoning framework for autonomous driving [172].

More recently, machine learning approaches have been used to enable automatic learning of concepts and semantically meaningful representations from structured and unstructured data. Some of them have already been mentioned or described in the previous section. In particular, deep learning techniques have been used successfully on a variety of tasks to detect and predict sub-symbolic patterns using multilayer perceptrons or reinventing (and correlating) patterns through Convolutional Neural Networks (CNN). Once sub-symbolic objects and patterns are found, there is the need for putting them together and making sense of them.

3.5.2 Learning to Make Decisions

Machine Learning-based methods have recently been proposed for automatically learning semantically significant concepts and representations from structured and unstructured data. Deep Learning approaches, in particular, have been effectively applied to a range of applications, including detecting and predicting sub symbolic patterns with layers of perceptrons or rewriting patterns with CNN. IBM's Watson, which is promoted as a tool that collects multiple sources of knowledge and assists the decision-making process, is one of the most well-known commercial systems that extensively uses knowledge representations. Currently, the fields with the most use are medical and financial, both of which include structured data. For example, Google's Google Knowledge Graph Search API employs Description Logic, whereas IBM's Watson uses a Shallow Knowledge Base [153]. WordNet, Freebase, and YAGO (Knowledge graphs-based) are three other well-known systems [45]. Automatic knowledge representation extraction necessitates a large dataset capable of covering relevant examples; otherwise, the model risks being overfitted, i.e., rendering it unable to extrapolate to further use cases. Other examples are: the paradox of universal vs domain-oriented representations, in which it is unclear whether more generic representations function as well as those developed expressly for a certain purpose; and the enormous number of alternative behaviours. The lack of standards is particularly problematic,

because Knowledge Representations are most helpful when shared among different intelligent systems, yet research in this sector is still in its early stages, with no standards expected in the foreseeable future.

The various synthetic and real datasets presented in this chapter are used to obtain information needed for the label perception and categorization phase. However, there is a lack of datasets for decision-making, particularly related with decisions that involve overtaking manoeuvring. Moreover, the scenarios, reasoning, inference and learning methods found in the literature used to make decisions do not take into account overtaking. This work is a step towards this goal. In the next section we present a possible solution to fill this gap by (1) suggesting variables relevant to solve this decision problem, (2) generating synthetic data for overtaking scenarios, (3) formulating rules that represent knowledge using as a basis our synthetic data **falar do artigo**, (4) manually labelling the data, and (5) analysing the data and building prediction models that support overtaking manoeuvring.

Chapter 4

Synthetic Data Generation and Knowledge Base

This chapter covers the steps in the process leading up to data generation and knowledge representation and is divided between the explanation of the data generation design, the tools and materials employed, and the presentation and discussion of the findings. First, CARLA is introduced, and its limitations are discussed, as well as the strategies utilized to overcome them. This is followed by an overview of the scenarios architected in order to obtain successful, unsuccessful and neutral overtaking cases. All parameters used in the creation of the scenarios are explained, as well as the reason for choosing them. Following that, a description of the variables intended to be collected in each simulation is provided, along with justification for their selection. An Entity Relationship (ER) diagram is also employed to illustrate the data planning and structure and then the five classes considered for this work are presented, which classify the overtaking manoeuvres as: successful legal and illegal, non-successful with and without collisions and no attempt to overtake. The feature engineering process is initiated, and new features are created to complement the initial dataset. Finally, the Portuguese traffic laws are codified in logical rules and applied to the data in order to differentiate legal from illegal manoeuvres. A pipeline that illustrates the order and the output/intent of each task was constructed for a clearer understanding of the phases that constitute this chapter. The pipeline is shown in Figure 4.1.

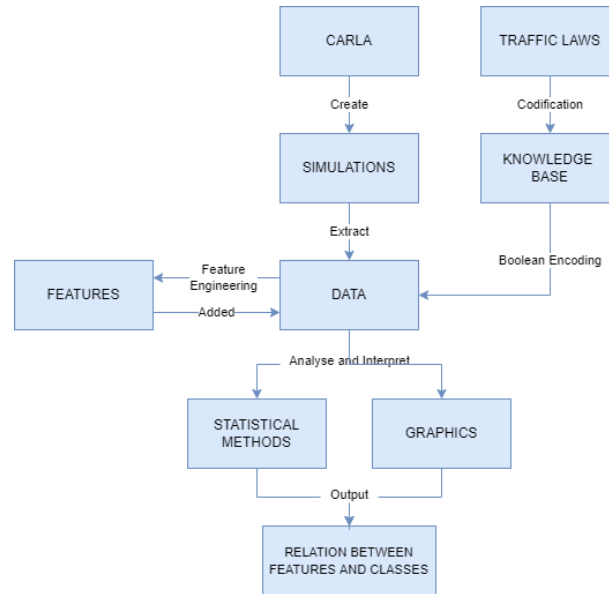


Figure 4.1: Pipeline of the first phase of the work.

4.1 Data Generation

Synthetic data is artificially generated information that can replace real data when it lacks quality, volume, or variety. Real data is also sometimes insufficient when it does not meet the needs of what is intended, or when its creation may cause danger or damage. Synthetic data is widely used in the field of artificial intelligence to, for example, train models when real data is lacking, to fill gaps in training data, to predict the future (old data may lose its value), to generate marketing images, etc. Generating synthetic datasets that are statistically significant and relevantly reflect real data can be challenging, since it is necessary to guarantee that the generated data is similar enough to the real data. Synthetic data is an added value for training neural networks, as they are more accurate when trained with a wider variety of data. However, gathering and labelling such massive datasets with thousands or even millions of objects is too expensive. In this regard, synthetic data can save money since simulators can reproduce scenarios that are challenging to reproduce in the real world. Apart from bringing the opportunity of collecting large amounts of data that in the real world may not be possible due to time limitation (there may not be time to collect the required amount of data), synthetic data also adds the advantage of being able to collect dangerous data that in the real world may cause harm/harm and/or put a living being at risk. A synthetic dataset should be sufficiently diverse, but it is necessary to control the random processes that generate the data in order to make it realistic. To generate synthetic data, models produce synthetic data based on the probability that certain data points will appear in the real dataset. Neural techniques like Variational Autoencoders and Generative Adversarial Networks are commonly used to generate synthetic data. In autonomous driving, synthetic data is often generated with the help of car simulators. The synthetic data is simulated data that generates photorealistic simulations that follow the laws of physics. Simulated data includes all necessary

annotations and dimensions, producing realistic 3D data.

The simulator chosen in this work was CARLA, an open-source tool for autonomous driving research developed to support development, training, and validation of autonomous urban driving systems. Following a thorough evaluation of numerous driving simulators, CARLA emerged as the most notable for being highly complex, having a variety of useful functionalities for this work, including the incorporation of weather conditions, and providing a thorough and understandable documentation. Carla has a flexible API that allows users to control all aspects related to the simulation, including traffic generation, pedestrian behaviors, weather, sensors, and much more [54]. The platform includes a large set of objects as well as static and dynamic actors that may be used to customize maps and scenarios. A wide range of sensors can be added to the car to collect information about all environment; weather can be customized in detail (Figure 4.2); simulations can be recorded and played and a blueprint library with a wide set of actors can be used.



Figure 4.2: Third-person view in four weather conditions [54].

The simulator made it feasible to create and manipulate every element of the road (something impossible in the real world), including the road layout, the flow of traffic, the types of vehicles present, assign speed values and perform manoeuvres such as lane changing. It was also possible to create risky circumstances using the simulator, such as collisions or running over. In order to collect as much information as possible and realistic, a planning and structuring phase of the scenarios and variables to be collected preceded the implementation of the simulations.

4.1.1 Limitations of the Simulator

Carla simulator was found to be the most suitable for simulating the scenarios in this work. It was chosen due to its complexity, variety, and popularity. However, this simulator exhibited some limitations in the construction of scenarios during the project. When attempting to complete

two lane changes in a row (in considerably close frames), the first limitation was encountered. The responsibility for the lane change process lies with the Traffic Manager. When the method "force_lane_change(self, actor, direction)" is called, a lane change is done to the vehicle described in actor to the right if direction is True and to the left if it is False [158]. This method has an associated timeout that prevents the execution of a new job before the old one has completed. To tackle this issue, two TrafficManagers were employed. Upon the first lane change of the ego vehicle, the first TrafficManager was turned off, thus ending its tasks. To make the second lane change and return to its original lane, a second TrafficManager was turned on and paired with the vehicle. Often an overtake can occur at high speeds and consequently take a few seconds between the first and the third stage (explained in Subsection 4.1.3). Without this solution, these circumstances, and data could not have been represented or collected.

Another crucial limitation of the system was the impossibility to force a lane change with the Traffic Manager when the vehicle is at a speed higher than 40 km/h. There were different solutions to this problem proposed. One approach was to reduce the speed to below 40 km/h the moment the car started the overtaking manoeuvre, and then return to the previous speed. This strategy was quickly shown to be impossible, since it manipulated/changed the simulation far too much. The values recorded were far from a realistic scenario. Because the vehicle was travelling at such a slower speed than the other vehicles, there was no risk of crashes or infractions of different traffic laws. It was realized that the speed difference was important for the preservation of the simulation environment and could not be changed. With this, another approach was taken where a conversion of the randomly generated speeds from a simple rule of 3 was performed. The maximum speed considered in the simulations was 120 km/h, so this value would correspond to 40 km/h in the simulator (maximum speed to be able to overtake). Once the speed of all the vehicles present in the simulation is converted, the timestamp is also converted and the difference between the vehicle speeds is preserved.

Finally, the weather conditions. The simulator provides a visual representation of various weather conditions: cloudy, rainy, day/night, sunset, etc [39]. However, the physics of the car during the simulation does not change depending on the level of precipitation, wind or fog. The same happens with the time of the day. The simulation does not differentiate between day and night. There is no reduction in road visibility and no activation of vehicle lights. The system has been updated to reflect these two modifications. The influence of weather conditions was considered an asset to the work. Therefore, since it was not implemented in the system, it was necessary to apply forces to the car depending on the level of rain and wind. Depending on the wind direction, a force is applied to the car to move it to the right or left. Forces are applied at longer time intervals for a lower average wind and rain values, and vice versa. When a high level of cloudiness (above 60%) is registered, or when the sun is at the driver's eye level (horizon line), his field of vision is reduced.

4.1.2 Scenarios

The first step in generating the data corresponding to overtaking situations was to create the scenarios that, by common knowledge, are considered possible to happen. In order to obtain positive, negative, and neutral cases for model training, successful, unsuccessful, and non-overtaking scenarios were considered. All environments were created randomly, but ensuring that all cases were represented. First, a random set of cars are placed in a portion of the world. The location, type, colour, and speed of the vehicle as well as weather conditions are randomly generated for each simulation.

The number of cars was between 2 and 6. The minimum 2 to guarantee always an ego vehicle and the vehicle it intends to overtake, and the maximum 6 considering the available space on the chosen portion of the street. Each vehicle is assigned initial values of location x , y and z and speed ranging from 50 to 120 kilometres per hour. The values were chosen because they represent the lowest and highest speeds permitted on Portugal's highways in article 27th of the Portuguese Highway Code (*Código da Estrada*) [12]. During the simulation, these values may fluctuate and may surpass 120 km/h or fall below 50 km/h. Various vehicle types were considered in the simulations, including small, medium, and large light passenger vehicles, vans, trucks, bicycles, and motorcycles. The ego vehicle does not depict bicycles or motorcycles because this effort is intended at autonomous light passenger vehicles. Another distinguishing characteristic of the cars is their colour. Yellow, blue, red, green, orange, grey, white, and brown are all options.

Finally, each simulation has its own set of meteorological conditions. Cloudiness, precipitation, wind intensity, and sun height angle are among the characteristics that can be changed in the simulator. For convenience, CARLA also provides a list of predefined weather presets that can be directly applied to the world. As a result, this work's simulations can select among the following weather conditions: *SoftRainNight*, *MidRainyNight*, or *HardRainNight*, which depict a nighttime scenario with low, mid, and high percentages of precipitation and fog, respectively, as well as low wind intensity. *HardRainNoon* indicates a daytime scenario with strong precipitation, and low wind and fog intensity; *ClearNoon* represents a daytime scenario with no precipitation, and low wind and fog intensity; *ClearSunset* represents a daytime scenario with a horizon line (sunlight at the driver's eye line) without precipitation and low wind and fog intensity; *ClearNight* represents a scenario with the same aspects as *ClearNoon* but at night; and *CloudyNight* represents a nighttime scenario with high intensity of fog and low precipitation and wind intensity [38]. An example of simulations performed in the *ClearNoon*, *HardRainNoon* and *ClearNight* scenarios can be seen in Figures 4.3a, 4.3b and 4.3c respectively.

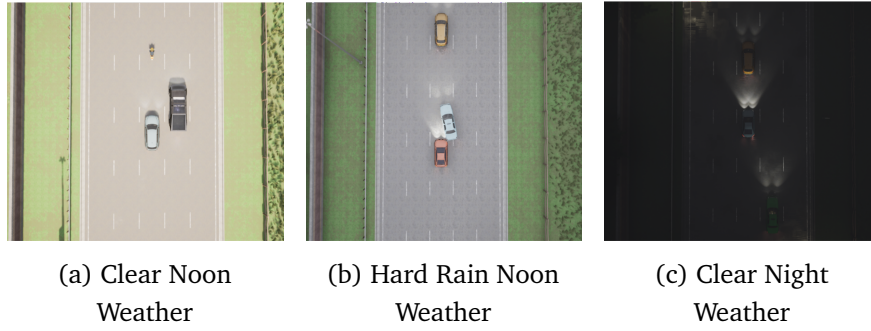


Figure 4.3: Overtaking scenarios with multiple weather conditions.

4.1.3 Collected Data

At each frame of the simulation, relevant information regarding each object present is collected. The data is entered into a data table, and Table 4.1 displays the description of the stored variables.

Table 4.1: Stored variables en each simulation.

Legend: idv represents the vehicle id and id_lane represents the id of the lane the vehicle is in.

Name	Description	Type	Unit of Measurement
S	Number of simulation	int	int
F	Frame	int	int
TS	Timestamp	float	seconds
IDego	Id of ego vehicle	int	int
Dim	Vehicle Dimension	(int, float, float, float)	(idv, x, y, z)
L	Vehicle location	(int, float, float, int)	(idv, x, y, id_lane)
V	Vehicle velocity	(int, int)	(idv, km/h)
D	Vehicle direction	(int, float, float)	(idv, x, y)
A	Vehicle acceleration	(int, float)	(idv, m/s ²)
MV	Max velocity corresponding to the ego vehicle lane	int	km/h
RT	Type of right line of the ego vehicle	string	"Solid", "Broken", etc
LT	Type of left line of the ego vehicle	string	"Solid", "Broken", etc
LW	Width of ego vehicle current lane	float	m
LWR	Width of right lane	(float, int)	(m, id_lane)
LWL	Width of left lane	(float, int)	(m, id_lane)

Table 4.1 continued from previous page

Name	Description	Type	Unit of Measurement
C	Collision between ego vehicle and other object	int	idv
Prec	Precipitation	int	%
Fog	Fog	int	%
Wind	Wind	int	%
DN	Day or night	string	Day or Night
HL	Horizon line	string	Yes or No
OV	Time when the overtake occurs (1 if true, 0 if false)	int	0 or 1

The variables to be collected were chosen based on literature research and the Portuguese road traffic, the document that regulates the traffic of people and vehicles since 1901 in Portugal [2]. It has been proven, as mentioned in Section 3.2.3, that factors such as weather, illumination, and road geometry, among others, influence driving. These variables are still essential in overtaking scenarios. This work has included additional factors in this study to determine if an overtake is successful or not. The variables collected in each simulation can have qualitative/categorical and quantitative/numeric values. As seen in the previous table, categorical variables are of type “string”, whereas numerical variables are of type “int” for whole numbers or “float” for decimal values.

Each moment of overtaking is represented by the variables S, F, and TS. S stands for the simulation’s unique identifier, F for each frame’s unique identifier, and TS indicates the seconds passed from the beginning of the simulation to the moment it is recorded. Information such as each vehicle’s unique identity (IDego for the ego vehicle and idv for the others) and dimension (Dim) are saved to identify the cars present in the simulation. To monitor all the activity of each vehicle, the location (L), recorded speed (V), wheel direction (D) and acceleration (A) are stored for each vehicle in each frame. The vehicle’s location is given by a point x,y together with the unique identification of the lane the vehicle is in. To account for the traffic condition factor, the maximum speed value (MV) for the lane in which the ego vehicle is located is recorded. It’s possible to tell if it is exceeding the maximum speed by comparing its current speed (V) to its maximum speed (MV). Considering other factors such as road geometry and lane marking, the types of lines on the right (RT) and left (LT) of the ego vehicle were recorded, as well as the width of the lanes that surround it (LW for the lane it is in, LWR for the lane on its right and LWL for the lane on its left). These variables are crucial to check, for example, whether the vehicle crosses a solid line (which is prohibited by Portuguese Highway Code in article 146th) [10]. In turn, by comparing the variables car size (D) and road width (LWR and LWL), it is possible to determine if the lane in which the automobile is heading has sufficient width. Overtaking without

endangering other road users is one of the conditions for successful overtaking. Since collisions between vehicles constitute a danger scene, it is necessary to store this information (variable C). Weather and lighting were also taken into account. These restrict the driver's view and make it harder to interpret the surroundings. As a result, rain (Prec), fog (Fog), and wind (Wind) percentages were measured. The time of day was stored in the variable DN which can register whether it is day or night. HL records whether the horizon light is dazzling the driver, a factor that was also considered relevant in limiting driver visibility. Finally, the variable OV simply stores the moment when the vehicle starts the lane change manoeuvre. This variable can be used to determine how many overtaking phases the vehicle has completed. Considering *v* as the car that the ego vehicle intends to overtake, three stages are considered in a successful overtake: the first is before the ego vehicle executes the lane change (ego vehicle is behind *v* in the same lane), the second is when it has already changed lanes and tries to reach a position ahead of *v*; and the third is when ego returns to the initial lane, leaving *v* behind. So, if there are two 1s in OV in a simulation, the vehicle has moved from the first to the second stage and recorded a 1, and then went to the third stage and recorded another 1. If it just gets one value 1 in OV, it has only progressed from the first to the second stage and is unable to back to its initial lane. There was no lane change if it didn't register any value 1 in OV. The overtake in the first two circumstances is considered a non-overtake in the first place.

4.1.4 Planning and Structuring

In Section 2.2 it is mentioned that most of the data held by businesses is unstructured. However, it is well known that this data requires structuring in order to be used in decision-making processes. For a successful comprehension and utilization of data, a planning phase is essential. Many artificial intelligence researchers agree that many tasks, including reasoning, planning, and decision-making, rely on a combination of three mental mechanisms: neural, symbolic, and probabilistic [115] [134] [62]. The symbolic component is used to represent and reason about abstract knowledge. The probabilistic inference model aids in the establishment of causal relationships between objects, the reasoning of counterfactual or never-before-seen events, and the handling of uncertainty. Finally, pattern recognition is used by the neural component to link real-world sensory data to knowledge [51]. Data is a representation of facts, a simple observation of the world. Data can be categorized into qualitative when it attributes a quality (e.g. colour of a ball) and quantitative when it can be measured (e.g. price of a book). However, it is necessary to interpret and contextualize the data in order to turn it into information. Structuring data is a crucial step in knowledge representation. This is how data is organized to be interpreted by machines and humans. There are several data structures used in programming like lists, stacks, trees, graphs, hash tables, among others. Data models, on the other hand, are tools that allow you to demonstrate how the data structures that will later support the decision processes will be built. They explain how the data will be organized and what relationships are intended to be established between them.

In this work, were applied an Entity Relationship diagram. ER is a type of flow diagram that

illustrates the relationships between different system components, including items, people, and concepts. They are frequently used to create and model relational databases, as well as to identify and fix logic or implementation issues. These models, also known as ERD or ER Models, express the interconnections of entities, relationships, and their qualities using a predetermined set of symbols, such as rectangles, diamonds, ovals, and connecting lines. Entities are represented by nouns and relationships by verbs. ER diagrams are constituted of entities, relationships, and attributes. They also indicate cardinality, which is a numerical definition of relationships. Entities, represented by rectangles, define people, objects, concepts, or events that can have data saved about it. Rhombuses, which represent relationships, show how different entities interact or are connected. Attributes are properties or characteristics of an entity and are represented by circles or ovals. Finally, cardinality defines the numerical attributes of the relationship between two entities or sets of entities. The three main cardinal relationships are one-to-one, one-to-many and many-to-many [169]. Relationship (ER) diagrams are used to display relationships, and they can only be used with structured data. However, the diagram was chosen because it is a very well-known and understandable platform and because of the possibility that it provides a preview of how tables should link to one another and which properties should be emphasized or eliminated. Since in this work there are several entities (e.g. Vehicle) with several attributes (e.g. colour) that relate to each other, the ER diagram was considered the most suitable to represent. The diagram reproduced in this work is shown in Figure 4.4.

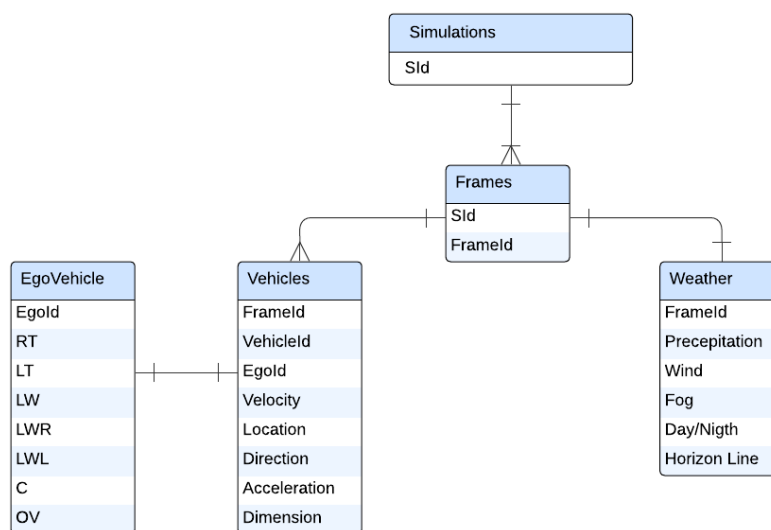


Figure 4.4: ER diagram developed.

In the diagram, the different entities are represented: Simulations, Frames, EgoVehicle, Vehicles, and Weather. Each entity has its own attributes, for example, each vehicle (belonging to the Vehicles entity) has an associated speed. Here the Ego Vehicle attribute is differentiated from the Vehicles attribute, which in turn stores information about all cars (including the ego vehicle), because the values of the variables RT, LT, LW, LWR, LWL, C and OV are only stored for the ego

vehicle. The relations are also represented as well as their cardinality. An example is the relation between the entity Simulations and the entity Frames that have a one-to-many relation, that is, each simulation has one or more frames, but each frame only belongs to one simulation. This graphic was used to help organize the ideas and future actions in the project. It was expected to visually express the specific needs to the overtaking manoeuvre and highlight the necessary code to be programmed/implemented at a later stage.

4.1.5 Classification

As mentioned in Subsection 4.1.2, successful and unsuccessful overtaking scenarios were considered as well as non overtaking attempts. Let v be the vehicle to be overtaken by ego vehicle. Scenarios in which the ego vehicle can achieve the following steps in order are classified as successful overtaking: visualize v (which is in front of it), ego vehicle change to the lane on its left; reach a location on the x-axis higher than the v ; and return to its initial lane. Figure 4.5a illustrates an example of a successful overtaking. In this case, the question is whether returning to one's initial lane is sufficient to be considered successful overtaking. A system that implements an overtaking decision method must additionally consider whether this presents a danger to other road users. According to article 38th of the Portuguese Highway Code (*Código da Estrada*), a vehicle can only overtake another vehicle if the manoeuvre does not represent a danger to those passing on the road [13]. As a result, various traffic rules must be considered in order to assure the manoeuvre's legality as well as the driver's and other vehicles' safety. These rules will be presented in detail later. In light of this, successful overtaking was divided into legal and illegal successful overtakes. In turn, overtaking considered unsuccessful are those where the ego vehicle begins the overtaking manoeuvre but does not finish. This can happen due to a collision or due to some event or obstacle that did not allow the manoeuvre to be completed, as illustrated in Figure 4.5c. In these cases, we might notice a decrease of speed after the car has started to overtake, which makes the car unable to get enough speed to overtake. Collisions, represented in Figure 4.5b, can have many causes, among them: poor visibility that prevents the driver from seeing the car ahead in time to brake safely; not keeping a necessary safety distance from the car ahead; adverse weather conditions that increase braking time; among others. Finally, the scenarios considered as neutral (no attempt to overtake) are those in which no overtaking manoeuvre is performed.

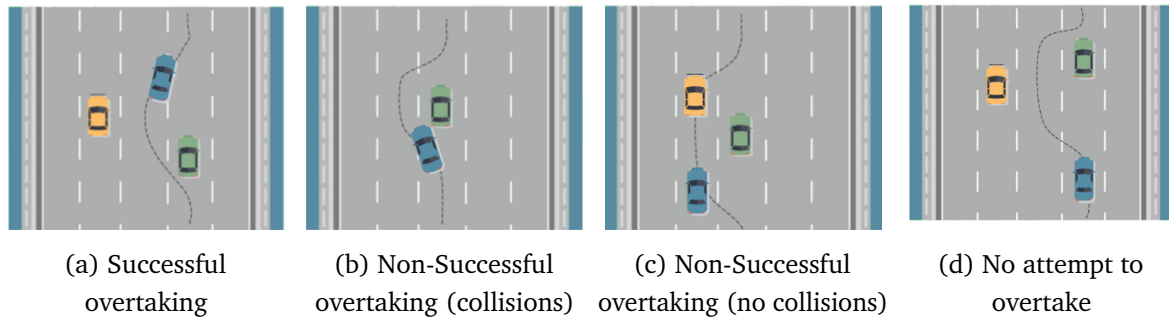


Figure 4.5: Classes chosen to represent the different overtaking scenarios.

Legend: The blue colour represents the ego vehicle; the green colour represents the vehicle that the ego vehicle wants to overtake; the yellow colour represents simple cars; the dashed black line represents the car's intended path.

Following that, a data analysis was performed to determine which elements had the greatest impact on the simulation's outcome. For each factor related to each class, the minimum, average, and maximum values, as well as the standard deviation, were gathered. These values were measured at the moment the driver initializes the overtaking manoeuvre. As indicated in Subsection 4.1.3, a successful overtaking is one in which the vehicle reaches 3 stages. Figure 4.6 displays these three stages, as well as the transition points between them. The first step, represented by letter A, occurs when the ego vehicle remains behind the other it intends to overtake. The second stage begins when the ego vehicle starts the overtaking manoeuvre, which is symbolized by transition a in Figure 4.6. The second stage continues with the vehicle in the lane it has moved into until it reaches an x-axis location larger than the other car. b represents the instant it executes the second lane change to return to its original lane. Thus moving on to stage 3, where the ego vehicle is in its initial lane, leaving the overtaken vehicle behind, represented by the letter C. As a result, all measures were taken at time a for successful overtakes, legal and illegal, as well as unsuccessful overtakes. For the non-overtake situations, the measured values in each frame of the simulation were averaged.

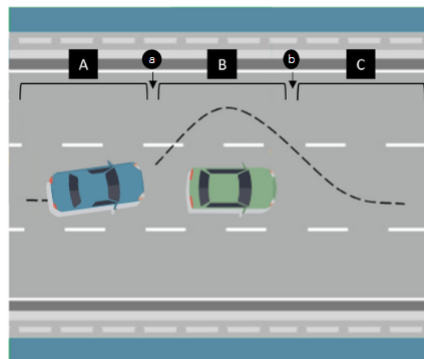


Figure 4.6: Representation of the 3 stages of a successful overtaking.

4.1.6 Feature Engineering

Feature engineering is the process of transforming raw data into useful features that are later used by the algorithm. This includes the process of creation, transformation, extraction, and selection. During this stage, features that add relevant information to the study and that can support the classification of the overtaking manoeuvre, were created and included to the dataset. Because all data is acquired using the CARLA simulator, no information is gathered from any other source. Using the imputation technique, missing values were removed. In frames when no collisions are recorded, missing values are discovered. Since the values are numeric, the missing values are set to 0. Following the creation process, each feature's statistics were analysed by computing the minimum, average, and maximum observed value for each class, as well as the standard deviation. The results of this analysis are presented in detail later.

In the process of creating the features, features created in different important studies in the literature were taken into account. The features were divided into two categories: static (values that don't change over time) and dynamic (values that change in each frame). Let E be the ego vehicle and P the vehicle that E intends to overtake. The static features considered were: time of the day; presence of horizon line; type of E; type of P; waiting time (time that E waited to start the overtaking manoeuvre); overtaking time (total time the overtaking manoeuvre takes); number of vehicles present in the simulation. The dynamic features considered were: current speed of E; current speed of P; speed difference between these two vehicles; distance between E and P; occupancy rate of the E's left lane (the lane E intends to head towards); weather, percentage of precipitation, wind and fog. Tables 4.6 and 4.5 illustrate the static and dynamic features created, respectively, as well as the measurement units of each. Letters S, M, L, V, T, B and MC represent the possible types of vehicles present in the simulation. S, M and L represent the small, medium and large light passenger vehicles, respectively; vans, trucks, bicycles, and motorbikes are represented by V, T, B and MC, respectively.

Table 4.2: Static Features Created.

Name	Static Features	Measurement
DN	Time of the Day	Day or Night
HL	Horizon Line	Yes or No
TE	Type of E	S, M, L, V or T
TP	Type of P	S, M, L, V, T, B or MC
WT	Waiting Time	seconds
OT	Overtaking Time	seconds
NV	Number of Vehicles	int number

Table 4.3: Dynamic Features Created.

Name	Dynamic Features	Measurement
SE	Current Speed of E	kilometers per hour
SP	Current Speed of P	kilometers per hour
DSEP	Speed Difference Between E and P	kilometers per hour
D	Distance Between E and P	metters
OLR	Occupancy Rate of the E's Left Lane	percentage
PREC	Precipitation	percentage
WIND	Wind	percentage
FOG	Fog	percentage

The features were created based on generally known overtaking scenarios along with information gleaned from study in the field. The selected factors are those considered to be relevant in the overtaking decision at the moment the driver initiates the manoeuvre. Time of day and horizon line were chosen as features because they contribute to a diminished perception

of the road. As referenced in Section 3.2.3, researches show that night sceneries make it difficult to perceive road objects and reduce the driver's understanding of the surrounding environment. The horizon line was an added feature because it represents a decrease in the field of view by being a light directed at the driver's eyes. In turn, the type of vehicles that overtake as well as those that are overtaken are also taken into account. The physics of a heavy vehicle differ from those of a light vehicle. Because it is heavier, it slows down and may take longer to complete the driving manoeuvre. A light vehicle that wants to pass a heavy vehicle, on the other hand, will need more time to reach a greater x-axis point than the other vehicle. The waiting time value is also essential, since a driver may be enticed to overtake if he has been driving behind that automobile for an extended period of time. If the contrary is true, the same thing can happen. The vehicle may have had enough time to adjust its speed to that of the vehicle in front of it and no longer sees the need to overtake. The overtaking time value represents the time it takes to manoeuvre from the moment the vehicle changes to the left lane until it returns to its initial lane, overtaking the desired vehicle. This aspect might also impact a driver's decision to overtake or not, if he knows how long the overtaking will take ahead of time. With a big safety distance between automobiles, a successful overtake will undoubtedly take longer. The number of vehicles in the simulation has an impact on traffic flow, which can play a role in the driver's decision to overtake. The motorist will have more confidence to overtake in a less congested environment, and the dangers of a failed overtake will be minimized. In terms of dynamic characteristics, both the speed of the overtaking vehicle and the speed of the one being overtaken is important. A car driving at a high speed will act differently than one driving at a low pace. The values of reaction, braking, and stopping distance are affected by the speed values. These distances are affected not only by the vehicle's speed, but also by the condition of the tyres, the efficiency of the brakes, and the road surface (wet, slippery, or sandy) [100]. The condition of the car is not taken into account in this work. These values, as well as overtaking time and waiting time, are also influenced by the speed of the car being overtaken. Speed values may not be sufficient alone, so the difference between the speed of the overtaking car and the speed of the overtaken car was also considered. A greater difference in speeds can be the cause for overtaking to happen. In theory, an automobile travelling at a very high speed behind a car travelling at a very low speed will need to overtake. Because the two automobiles are moving in the same direction and with the same orientation, the difference in speeds is computed by subtracting the quicker car's speed from the slower car's speed. The distance between the two vehicles must also be taken into account. An important practice in driving is to increase the safety distance between cars as a way to prevent accidents [100]. Another factor chosen was the occupancy rate of the lane to the left of the vehicle (the one to which the ego vehicle wants to head to overtake the other car). The traffic flow must be considered once more, but this time in a specified lane. Finally, weather conditions were taken into account because, as is well known, they have an impact on the state of the pavement and the driver's visibility. In different weather circumstances, the identical overtaking scenario produces vastly diverse results. As previously stated, a scenario with heavy rain or fog can increase reaction time, braking distance, and stopping distance, resulting in crashes. Wind, in turn, might cause the vehicle's trajectory to be disrupted.

4.1.7 Traffic Rules

Figuring out if an overtaking is successful or unsuccessful might be a challenging task. It's difficult to tell what motivated a driver to do a particular action just by looking at a driving circumstance. Internal and external vehicle elements, as well as driver-related issues, must all be considered. The factors relating to the driver, such as tiredness, exhaustion, or inebriation, were not been examined in this work. It was determined that in order for overtaking to be successful, it must be safe for all road users [14]. The safety of the overtaking manoeuvre will then be dictated by traffic rules that must be obeyed during, before, and after the overtaking manoeuvre. It is possible to extract information about the rule infraction from the data collected during the simulation. The overtaking manoeuvre was subjected to both general highway traffic rules and specific overtaking manoeuvre laws. The general traffic rules were as follows:

1. It is forbidden to exceed the maximum speed limits up to 30 km/h, outside the towns [12];
2. Crossing a continuous/solid line is prohibited [10];
3. The driver of a moving vehicle should maintain sufficient distance between his vehicle and the vehicle ahead of him [11];

The traffic rules considered focusing on overtaking manoeuvre were as follows:

4. It is prohibited to overtake if a driver has initiated a manoeuvre to overtake you [13];
5. It is prohibited to overtake a car that is overtaking a third car [14];
6. It is prohibited whenever the lane width is insufficient [14];
7. It is prohibited in all low-visibility areas [14];
8. When the driver has completed the manoeuvre and is able to do so safely, he must turn right again [13].

Taking these rules into account, first order logic was subsequently used to formalize them. All rules were applied to the ego vehicle of each simulation, since it is the car that will do the overtaking and will make the decision. To determine if the 1st rule has been satisfied, information about the ego's speed in each frame of the simulation is needed, as well as the maximum speed assigned to the stretch of road it drives on. The first rule is fulfilled if its speed does not exceed the maximum limit by more than 30 km/h throughout the simulation. For the 2nd rule, two moments of the manoeuvre are checked. When the ego vehicle passes into the lane to his left at an early point of the manoeuvre (a point of Figure 4.6), the type of line to the left of the vehicle that separates the two lanes is checked. In a second moment, when it returns to the initial lane after being in front of the overtaken car (b point of Figure 4.6), the type of line to the right of the vehicle is checked. If one of these lines is of the "Solid" type, i.e., continuous, the rule is broken. The 3rd rule can be very ambiguous, since the term 'sufficient' does not translate to the

precise amount of meters needed between vehicles. The two-second rule is one strategy that is frequently utilized in this situation. This technique advocates for leaving adequate distance between the vehicle and the vehicle in front of it to safely stop if the car abruptly stops. The rule entails picking an object that is close to or above the road, such as a sign or a tree. The driver behind you counts down 2 seconds as the vehicle in front of you overtakes you. If you approach the same object before the timer runs out, the gap between the two automobiles is too small. In bad weather conditions or when following large trucks, it is advised to increase the count to 3 or 4 seconds for additional space [111] [56]. Figure 4.7 illustrates an example of the two-second rule approach.

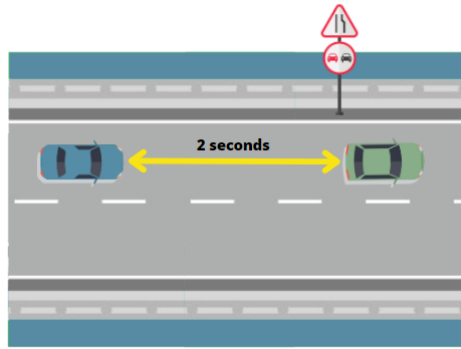


Figure 4.7: Example of the two-second rule approach.

In this work's case, the safe distance was calculated mathematically using the following script:

```

1 min_dist = v_ego * 1000 / 3600 * 2
2 min_dist = min_dist * 40 / 120
3
4 if loc_p > loc_e :
5     loc_p = loc_p - (dimx_p / 2)
6     loc_e = loc_e + (dimx_e / 2)
7     if loc_p - loc_e < min_dist:
8         print("no safe distance")
9
10 if loc_p < loc_e:
11     loc_p = loc_p + (dimx_p / 2)
12     loc_e = loc_e - (dimx_e / 2)
13     if loc_e - loc_p < min_dist:
14         print("no safe distance")

```

Listing 4.1: Code sample for calculating the minimum distance for safe overtaking.

The first line multiplies the current speed of the ego vehicle by 1000 and divides by 3600 in order to convert the speed from km/h to m/s. The minimum distance to ensure safety during the manoeuvre (**min_dist**) is given by the two-second rule, so this distance is multiplied by 2. Due to the adjustment explained in Section 4.1.1, the conversion of distances is done in row 2. At the first moment of the overtaking, the overtaken vehicle is in front of the overtaking vehicle (ego), line 4. Because the current location corresponds to the vehicle's centre point, it was

essential to add the vehicle's dimension on the x-axis (**dimx_p** and **dimx_e**). Since the dimension corresponds to the entire vehicle, we divide by 2 because we only want the value from the centre to the back of the car (in the case of the overtaken vehicle), line 5, or to the front of the car (in the case of the ego vehicle), line 6. Figure 4.8 represents the logic behind the calculation of the distance between two vehicles during the first moment of overtaking. The value obtained in row 5 (**loc_p**) is represented by the yellow dot, and the value obtained in row 6 (**loc_e**) is represented by the purple dot. Thus, the rule is broken if the distance between the two vehicles is less than the minimum space required for a safe distance, line 8. Starting at line 10 refers to the second moment of the manoeuvre, when the car aims to return to the original lane and so has a higher x-axis position than the overtaken car. The mathematical logic is inverse to lines 4-7 and the rule has been infringed once again if the distance between the cars is less than the safety requirement.

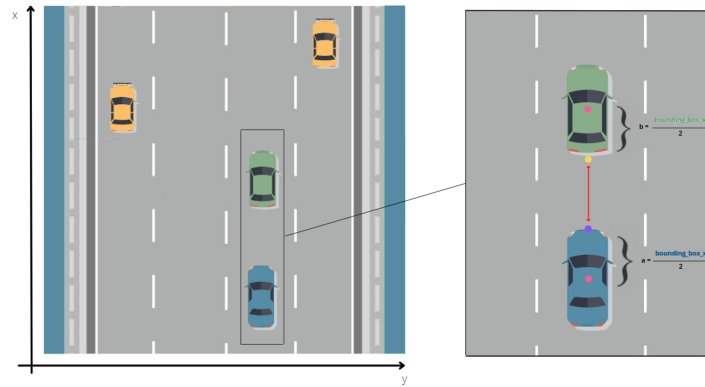


Figure 4.8: Logic for calculating the distance between two cars from the data collected by the simulator.

Legend: pink point: centre point to vehicle location (x, y, z); purple point: $\text{loc_e} + a$; yellow point: $\text{loc_p} - b$; red arrow: distance between the two vehicles = $b - a$

When it comes to overtaking manoeuvre traffic rules, the 4th rule prohibits ego's driver from overtaking when a third vehicle is overtaking him. It was feasible to verify the violation of this rule using the data acquired from the simulations by looking at the direction of the wheels of the car trailing behind the ego vehicle. If, at the time the ego vehicle initiates the overtaking manoeuvre, the automobile following behind has a sufficient degree of wheel spin to the left, then the rule has been broken. To check rule number 5, we must first determine whether the vehicle we want to pass is passing a third party. Using the same logic as the preceding rule, if the value of the overtaken car's left-hand wheel rotation is substantial at the time the ego vehicle begins its overtaking manoeuvre, the rule is violated. Rule number 6, on the other hand, is violated when the driver performs an overtaking move in a lane that is narrower than the width of his automobile. To test this rule, all that is required is to access the vehicle width, which is represented by the y value of the Dim variable, and the LWL variable, which indicates the width of the lane to the vehicle's left. The condition is broken if the value of LWL is less than the y value of Dim. 7th rule is broken if the driver's visibility is poor when the vehicle is doing the overtaking action. The simulator does not provide a value for visibility. Some aspects present

in the simulation scenario that would reduce driver visibility in a real-world setting were taken into account. Low visibility is exacerbated by high amounts of fog, nighttime settings, and the existence of the horizon line (light shining in the driver's eyes). A weighted average of these criteria was determined, with a score of more than 60% indicating inadequate visibility. Finally, rule 8 is satisfied if the vehicle returns to its starting lane as soon as it has travelled a distance greater than the minimum necessary.

4.2 Materials and Methods

For this work, scenarios were created and then simulated in the CARLA simulator. The data was extracted from the simulator thanks to 'GET' requests from the Python API and stored using the Microsoft Excel platform. All parameters used during the simulation were chosen after considering a number of factors that are discussed in more depth later. Although the data are artificially simulated, it is desirable that they represent the real world. In order to achieve this, a validation was conducted by contrasting the results from this study with studies from the literature that make use of real data. The final dataset consists of 276 simulations resulting in 22 columns (variables) and 111242 rows of data. The number of frames differs from simulation to simulation, since this is interrupted when the information being collected is no longer relevant for the study. These cases arise when: the manoeuvre is completed reaching the 3 states defined in Subsection 4.1.3; the ego vehicle collides with another vehicle; or when the simulation time exceeds 60 seconds, avoiding infinite simulations. This procedure prevents memory expenditure and excessive density of the dataset. The presence of all simulation cases (successful, unsuccessful and neutral) was ensured, as well as the presence of the five classes defined in Section 4.1.5. For data processing, application of the methods and visualization of graphs, Python code and some of its libraries were used such as Pandas [122] to create and manipulate dataframes, matplotlib [103] and sns [8] to plot the graphs and sklearn [137] to implement the classification and prediction algorithms. Other libraries like numpy [114], scipy [138], dython [55] and yellowbrick [175] were also used.

4.2.1 Carla Simulator

CARLA is designed to be highly extensible, which allows users to create custom functionality or content to suit specialized applications or specific needs. Some started system features will be demonstrated below followed by the code samples used to obtain each one.

- **Launching CARLA and connecting the client:**

CARLA can be launched from the command line, either a Windows executable or a Linux shell script.

```
1 cd /carla/root
```

```
2 ./CarlaUE4.sh
```

To use the Python API to manipulate CARLA, we must connect the Python client to the server through an open port. Through client and world objects, the client can control the simulator.

```
1 import carla
2 import random
3
4 client = carla.Client('localhost', 2000)
5 world = client.get_world()
```

- **Loading a map:**

The user has access to all elements of the simulation, including the map, the objects within the map such as buildings, traffic lights and vehicles. The server normally loads a default map (Town10) but it can be changed in the config.py script:

```
1 ./config.py --map Town06
```

or loading a map from the client.

```
1 world.load_world('Town06')
```

- **Adding NPCs:**

In order to simulate a real environment with traffic and other road users, we need to find some appropriate spots in the map to spawn vehicles. First, we select the vehicles we want from the blueprint library.

```
1 vehicle_blueprints = world.get_blueprint_library().filter('*vehicle*')
```

Now, we can add 10 vehicles to random pre-defined spawn points provided by the map.

```
1 spawn_points = world.get_map().get_spawn_points()
2
3 for i in range(0,10):
4     world.try_spawn_actor(random.choice(vehicle_blueprints), random.
5                           choice(spawn_points))
```

Or we can spawn one vehicle at a specific point x,y,z on the map (e.g., 90,-16,0).

```
1 spawnPoint = carla.Transform(carla.Location(x=90, y=-16, z=0), carla.
2   Rotation(pitch=0, yaw=0, roll=0))
3 vehicle_1 = world.spawn_actor(vehicle_1, spawnPoint)
```

Usually, an ego vehicle is included to serve as the simulation's centerpoint. It's the vehicle that the autonomous agent will control.

```
1 ego_vehicle = world.spawn_actor(random.choice(vehicle_blueprints),  
    random.choice(spawn_points))
```

- **Adding Sensors:**

Sensors are attached to the ego vehicle to collect information (e.g., a collision detector sensor).

```
1 collision_bp = world.get_blueprint_library().find('sensor.other.  
collision')  
2 collision_sensor = world.spawn_actor(collision_bp, attach_to=  
    ego_vehicle)
```

We need to set the camera to record using the `listen()` method once it has been launched. The `listen` method takes a callback as an argument that specifies what should be done with the data (e.g., call “`store_collision`” function).

```
1 collision_sensor.listen(lambda event: store_collision(event))
```

Later, it is possible to save to disk in image or video format (e.g., store the point cloud from LiDAR sensor in each frame).

```
1 LiDAR_sensor.listen(lambda point_cloud: point_cloud.save_to_disk('out  
/%06d.png' % point_cloud.frame))
```

- **Traffic Manager:**

The Traffic manager controls vehicles to autonomously move around the roads of the map, following the road conventions and behaving like real road users. After finding all the vehicles in the simulation using the `world.get_actors()` method, we can set the autopilot mode to hand over control of each vehicle to the Traffic Manager.

```
1 for vehicle in world.get_actors().filter('vehicle'):  
2     vehicle.set_autopilot(True)
```

Variables and methods from the Python API allow the environment customization to make it as realistic as possible and to store all necessary information. Getters and Setters methods can be used to give and obtain information, respectively, about objects. Some classes and their variables and methods applied at work will be demonstrated below.

- **Actors:** CARLA defines actors as anything that plays a role in the simulation or can be moved around. That includes: pedestrians, vehicles, sensors, and traffic signs.

- **Instance Variables:**

- * `id` (int): unique identifier for this actor.

- * `is_alive (bool)`: returns whether this object was destroyed.
- * `type_id (str)`: the identifier of the blueprint this actor was based on.

– **Methods:**

- * `destroy(self)`: tells the simulator to destroy this actor and returns True if it was successful.
- * `add_force(self, force)`: applies a force over a certain period of time at the centre of mass of the actor.
- * `add_impulse(self, impulse)`: applies instantaneous forces at the centre of mass of the actor, usually applied once.

– **Getters:**

- * `get_acceleration(self)`: returns the actor's 3D acceleration vector at the moment.
- * `get_velocity(self)`: returns the actor's velocity vector at the moment.
- * `get_transform(self)`: returns the actor's transform (location and rotation) at the moment.

– **Setters:**

- * `set_target_velocity(self, velocity)`: velocity is a vector measured in m/s.
- * `enable_constant_velocity(self, velocity)`: sets a vehicle's velocity vector to a constant value over time.
- * `set_transform(self, transform)`: teleports the actor to a given transform (location and rotation).

- **Client:** the Client establishes a connection between CARLA and the simulation server. The CARLA library (`libcarla`) is present on both the server and the client, with some peculiarities that allow communication between them.

– **Methods:**

- * `load_world(self, map_name, reset_settings=True)`: creates a new world with default settings using `map_name` map and destroy all actors in the current world. `reset_settings` is an option to reset the episode setting to default values.
- * `reload_world(self, reset_settings=True)`: reload the current world with default settings and the same map. All world's actors are destroyed, but traffic manager instances stay alive.
- * `start_recorder(self, filename, additional_data=False)`: enables the recording feature, which will saving all information about the simulation in file `filename`. `additional_data`, when True, enables recording non-essential data such as bounding box location, physics control parameters, etc.
- * `stop_recorder(self)`: stops the recording in progress.
- * `replay_file(self, filename, start, duration, follow_id, replay_sensors)`: playback a file `filename`. We can choose at what second to start (`start`), the id of the car to follow (`follow_id`), the duration of the playback (`duration`) and whether to disable the sensors (`replay_sensors`).

- * `stop_replayer(self, keep_actors)`: stops the replaying in progress.
- * `show_recorder_collisions(self, filename, category1, category2)`: show the collisions between two vehicles (category1 and category2) registered in recorder file filename.
- **LandMark**: defines any type of traffic landmark or sign affecting a road.
 - **Instance Variables**:
 - * `road_id(int)`: unique identifier for this road.
 - * `value(float)`: value printed in the signal (e.g. speed limit)
 - * `waypoint(carla.Waypoint)`: a point of a specific lane.
- **LaneChange**: defines the permission to turn either left, right, both or none.
 - **Instance Variables**:
 - * `NONE`: only going straight is allowed.
 - * `Right`: turning right is allowed.
 - * `Left`: turning left is allowed.
 - * `Both`: turning either right or left is allowed.
- **LaneMarking**: gathers all the information regarding a lane marking.
 - **Instance Variables**:
 - * `color(carla.LaneMarkingColor)`: color of the marking (White, Blue, Green, Red, Yellow or Other)
 - * `lane_change (carla.LaneChange)`: permissions for said lane marking to be crossed.
 - * `type (carla.LaneMarkingType)`: lane marking type (Solid, Broken, Curb, etc.)
 - * `width (float)`: horizontal lane marking thickness.

4.2.2 Scenarios and Data Collected

After planning and defining the scenarios to be represented, parameters were assigned at the beginning of each simulation. The number of vehicles as well as their type, colour, and speed were varied in the intervals already mentioned in Subsection 4.1.2. The chosen portion of the world represents a one-way highway consisting of 5 traffic lanes. This portion has an x-axis between 320 and 450 and a y-axis between 238 and 258, as shown in the Figure 4.9, where ego vehicle is represented by colour blue and yellow vehicles are the surrounding ones. In the simulator environment, waypoints are represented by a xyz axis. Because the z-points always have the value 0 in the measurements, they are ignored at the point where the cars have been added to the scenario (the cars are always on the ground). The value of z subsequently assumes significance, as the height of the cars can play a significant role in overtaking. The initial placement of the vehicles can be anywhere between 320 and 450 on the x-axis and on the y-axis each lane is

represented by a value. That is, vehicles with a y-location between 238 and 242 are in the first lane, those with a y-location between 242 and 246 are in the second lane, those with a y-location between 246 and 250 are in the third lane, those with a y-location between 250 and 254 are in the fourth lane, and those with a y-location between 254 and 258 are in the fifth lane. Since the ego vehicle always wants to overtake, it never started on the fifth lane, since a free lane on its right is needed for it to perform the manoeuvre.

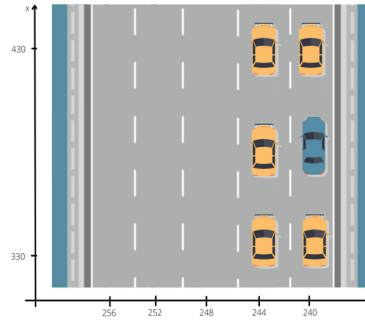


Figure 4.9: Street Portion Layout.

Once the parameters were assigned to each simulation, the output of each was recorded and organized in text format in a data table. As a crucial component for the later analysis and interpretation of the data obtained in each simulation, a mp4 video was saved for each one. A screenshot of the video can be seen in the Figure 4.12.

4.2.3 Data Exploration

Humans are significantly more perceptive of visual data than numerical data. Without any visual component, data scientists and analysts find it difficult to attribute meaning to rows and columns of data and explain it. Data exploration allows for a better understanding and detection of anomalies and linkages that could otherwise go missed. At this stage, graphs like histograms, box plots, and swarm plots were used to graphically display and interpret the collected results. The dataset corresponding to the data collected in each simulation was read and stored in a variable called “df” and the S column (unique identification number of the simulation) because it is just a column informing about the simulation number to which the data refers (it is not a feature).

The number of cases in the dataset corresponding to each class was calculated and illustrated with a histogram, a graphic representation of a data distribution made up of columns and bars. Each column’s base represents a class, and its height represents the quantity, or absolute frequency, with each class’s value appearing in the dataset if the classes are uniform or the density of frequency of the classes is not uniform. The histogram and the frequency associated to each class were reproduced with the Python code below.

```
1 import seaborn as sn
2
```

```
3 sns.catplot(x='CLASS', data=df, kind='count')  
4 df.groupby('CLASS')['CLASS'].count()
```

Listing 4.2: Code sample to display the histogram and the frequencies of each class.

The first line represents the import of the seaborn package from Python [139], a Python data visualization library based on matplotlib which, in turn, is a comprehensive library for creating static, animated, and interactive visualizations in Python [103]. The histogram is the output of line 3 and the last line displays the frequency of each class for a better understanding of the graph. For a better understanding of line 3: the dataset that had already been defined (`data=df`) was used, and a count (`kind='count'`) of the instances in which a certain class was checked (`x='CLASS'`) was made. Therefore, to understand the distribution of each feature, a histogram for each one was also plotted and analysed.

Another representation for interpreting the data was the box plot, a tool that allows the visualization of outliers and the distribution of the data. The diagram consists of a minimum and a maximum value and the first, second and third quartile. The median or second quartile is the mean value of the data; the first quartile is the mean value between the lowest number (not the minimum) and the median of the data; the third quartile is the mean value between the median and the highest value (not the maximum) of the data. Outliers are points of observation far away from other observations. Since the variables were not all at the same scale, before computing next plots it was necessary to use the `MinMaxScaler` function provided by the `sklearn.preprocessing` package to transform the features by scaling each feature to a given range [0-1] by default. The dispersion of the data can be represented by the difference between the third and first quartile, i.e. the size of the box. The amplitude, in turn, is obtained from the difference between the maximum and minimum value. The dispersion is a more robust measure because it does not consider outliers [89]. A swarm-plot was also implemented where only the points are adjusted so it won't get overlap to each other as it helps to represent the better representation of the distribution of values [140].

An interesting next step is to understand the relationship between two variables. For example, it would be desirable to know if the type of vehicle being overtaken has an impact on the ego vehicle's safety distance. A correlation coefficient is one technique to quantify this relationship. The degree to which two variables are linearly related is known as correlation. Correlation does not always imply causation, two variables can have a strong correlation due to a random exogenous occurrence. For example, a clothing store increases the number of sales of fur coats during the winter. As a result, there is a strong correlation between coat unit sales. It can be seen that there is a causal relationship in this example, as extreme winters enhance coat sales. Coat sales, on the other hand, are strongly correlated to the Olympic events. Here it is very clear that the Olympic Games are definitely not caused because of the coats, so there is no causation here. The correlation coefficient is a statistical measure of how strong a relationship exists between two variables' relative movements, with the range of values -1 to 1. A positive correlation corresponds to a directly proportional relationship and when it reaches the value of 1 it shows a perfect positive correlation. A negative correlation denotes an inversely proportionate relationship, and

when it reaches the value -1, it is said to be a perfect negative correlation. A correlation of 0.0 indicates that two variables do not have a linear correlation. Pearson and Spearman are the two most prominent and well-known correlation coefficients. The main distinction between the two coefficients is that Pearson deals with linear relationships between two variables, whereas Spearman deals with monotonic relationships as well. Another distinction is that Pearson uses raw data values for variables, while Spearman uses rank-ordered variables. A monotonic relationship is a relationship where as the value of one variable increases, the value of another variable also increases or decreases, but not exactly at a constant rate. The rate of increase/decrease in a linear relationship is constant. Following the graphing of pairwise correlations in the dataset (as shown in Figure 4.10a), it was discovered that some variables are linearly related, such as DSEP and SE (Figure 4.10c), while others are monotonically related, such as SE and SP (Figure 4.10b).

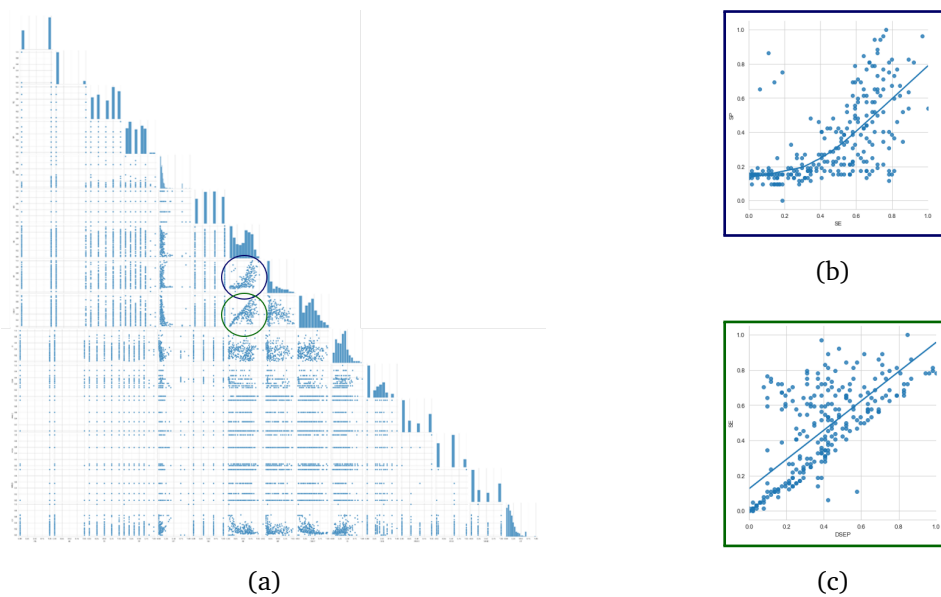


Figure 4.10: Features Relation.

Legend: (a) corresponds to pair plot output for all features; (b) corresponds to correlation points between SE and SP; (c) corresponds to correlation points between DSEP and SE.

As a result, the spearman coefficient was used to calculate the correlation matrix between the dataset variables. The code used for this calculation is below.

```
1 from dython import nominal
2
3 nominal.associations(df, num_num_assoc='spearman', clustering=True)
```

Listing 4.3: Code sample to compute the correlation matrix.

The dython library [55] (set of data analysis tools in python) was used to compute the correlation matrix. Since the data includes the variable CLASS which is categorical, the module nominal.associations [184] was used to compute the correlation of association of the features in the data set with categorical and continuous features. 'clustering=True' causes the computed

associations to be sorted into groups by similar correlations. There is a p-value associated with each association, which indicates the chance that the null hypothesis is true. P-value is the measure of the probability that an observed difference could have occurred just by random chance. A p-value determines whether there is evidence to reject a null hypothesis. The greater the difference between two observed values, the less likely it is that the difference is due to random chance, and this is reflected by a lower p-value [121]. A p-value of 0.05, for example, indicates that there is only a 5 percent possibility that the sample results happened by chance. This means that the outcome is 95 percent guaranteed. In order to uncover patterns and connections between variables as well as between variables and class, the correlation matrix was carefully examined. For a better comprehension of the data, several high positive and negative values, as well as the null values of correlation, were highlighted and analysed. Once a 95% confidence level was employed, the p-value was used to determine the linear relationship between two variables by comparing the p-value to the significance level with $\alpha = 0.05$. Thus, there is a significant relationship between the variables if p-value is less than or equal 0.05.

4.2.4 Traffic Rules

Following the definition of each traffic infringement presented in Subsection 4.2.4, a study was performed to determine under what conditions each violation occurs and how it may affect the driving environment. The convention used was as follows: constants are introduced using upper case letters, while variables and predicates are introduced with lower case letters. Initially, first order logic was used to formalize the rules that were presented in natural language. To define the environment, the following variables were defined:

- Type of lane divider: $line = \{Continuous, Dashed\}$. Represents the line which is to the left of the car. Unlike *Dashed*, if it is *Continuous*, overtaking is not allowed.
- Speed: $speed = int$. Represents the speed of the ego vehicle.
- Speed limit: $speedlimit = int$. Represents the maximum velocity allowed in the ego vehicle lane.
- Ego vehicle location: $locego = float$. Location, on x-axis, of the ego vehicle.
- Front vehicle location: $locfront = float$. Location, on x-axis, of the car ahead of the ego vehicle that is going to be overtaken.
- Overtaking status: $overtaking = \{FrontOvertaking, BehindOvertaking, EgoOvertaking\}$. *FrontOvertaking* represents that the car in front of the ego vehicle is overtaking a third vehicle. *BehindOvertaking*, on the other hand, means that the vehicle behind the ego vehicle is overtaking it. *EgoOvertaking* represents that the ego vehicle is overtaking.
- Ego vehicle dimension: $dim = int$. Width of the ego vehicle.
- Lane width: $width = int$. Width of the road to the left of the ego vehicle.

- Fog percentage: $fog = int$. Fog percentage.
- Time of the day : $day = \{Night, Day\}$. Represents the time of the day when the simulation takes place.
- Collision: $col = \{Yes, No\}$. *Yes* represents a collision during the simulation.
- Rule: $rule = Line, MaxSpeed, Distance, OtherOvertaking, LaneWidth, Visibility, Collisions$. Represents the violated rule.

Using these variables, the following predicates were defined. They all represent facts about the ego vehicle. The variable *sim* represents the simulation number identifier.

- $dividerLine(sim, frame, line, overtaking)$: identification of the divider line.
- $maxSpeed(sim, speed, mspeed)$: during the simulation, the speed was exceeded.
- $minDistance(sim, locego, locfront, speed)$: the distance is sufficient when it is greater than the distance calculated using the 2-second rule. Conversions are explained in Subsection , and fraction values have been simplified.
- $otherOvertaking(sim, frame, overtaking1, overtaking2)$: when the ego vehicle opted to overtake, another vehicle performed the same manoeuvre.
- $laneWidth(sim, frame, width, dim, overtaking)$: lane width is sufficient when the lane to the left of the ego vehicle has width greater than 1 (chosen value) + the width of the car.
- $lowVisibility(sim, day, fog, i, overtaking)$: since it is impractical to compute the driver's percentage of vision in simulations, the quantity of fog or the time of day, which is regarded in the literature as reducing the field of vision, determine this value. Therefore, when the FOG variable is below 60 or, when it is nighttime, when a random value (represented by *I* between 0 and 10 is below 5, visibility is regarded to be high.
- $collisions(sim, col)$: collisions occurred during the simulation involving the ego vehicle.
- $prohibitedManoeuvre(sim, rule)$: ego vehicle is doing a prohibited manoeuvre during the simulation.

With these predicates, MLN formulae were constructed to encode traffic rules and the knowledge about the environment in first-order logical sentences, as shown in the list below. In order to reduce the length of the rule, predicates and constants were abbreviated as shown in Table 4.4.

Table 4.4: Abbreviation of predicates and constants.

s	<i>sim</i>	ms	<i>maxSpeed</i>	OvE	<i>EgoOvertaking</i>	dl	<i>dividerLine</i>
w	<i>width</i>	sp	<i>speed</i>	OvB	<i>BehindOvertaking</i>	C	<i>Continuous</i>
d	<i>dim</i>	sl	<i>speedlimit</i>	OvF	<i>FrontOvertaking</i>	D	<i>Dashed</i>
oov	<i>otherOvertaking</i>	md	<i>minDistance</i>	lw	<i>laneWidth</i>	le	<i>locego</i>
col	<i>collisions</i>	lf	<i>locfront</i>	pM	<i>prohibitedManoeuvre</i>	Y	<i>Yes</i>

1. It is forbidden to exceed the maximum speed limits up to 30 km/h, outside the towns.
 $ms(s, sp, sl) \wedge sp \geqslant sl + 30 \Rightarrow pM(s, MaxSpeed)$
2. Crossing a continuous/solid line is prohibited.
 $dl(s, C, OvE) \Rightarrow pM(s, Line)$
3. The driver of a moving vehicle should maintain sufficient distance between his vehicle and the vehicle ahead of him.
 $md(s, le, lf, sp) \wedge lf - le \leqslant sp \times 10 \div 27 \Rightarrow pM(s, Distance)$
4. It is prohibited to overtake if a driver has initiated a manoeuvre to overtake you.
 $oov(s, OvE, OvB) \Rightarrow pM(s, OtherOvertaking)$
5. It is prohibited to overtake a car that is overtaking a third car.
 $oov(s, OvE, OvF) \Rightarrow pM(s, OtherOvertaking)$
6. It is prohibited whenever the lane width is insufficient.
 $lw(s, w, d, OvE) \wedge w \leqslant d + 1 \Rightarrow pM(s, LaneWidth)$
7. It is prohibited in all low-visibility areas.
 $lv(s, day, fog, i, OvE) \wedge (fog \geqslant 60 \vee (day = Night \wedge i > 5)) \Rightarrow pM(s, Visibility)$
8. When the driver has completed the manoeuvre and is able to do so “safely”, he must turn right again. “Safely” means, among other things, that there is no danger of collision.
 $col(s, Y) \Rightarrow pM(s, Collisions)$

Since all rules are considered to classify an overtaking manoeuvre as successful and legal (Success_L), the final rule is the combination of all the above. The next step was to represent the previous rules in Python code and add a new R (Rule) column to the initial dataset. The column contains all the rules violated during this simulation, separated by the symbol ;. The abbreviations VI, D, C, MV, LT, and BO represent overtaking in low-visibility areas, lack of safety distance, collisions, exceeding the speed limit, crossing the continuous line and overtaking while a third car is overtaking, respectively. When none of these rules are violated, the column value is None. An excerpt of the dataset is shown in Figure 4.11.

S	DN	HL	TE	TP	WT	NV	SE	SP	DSEP	D	OLR	PREC	FOG	WIND	OT	CLASS	R
1	Night	No	T	L	3.35	4	90	54	36	36.84	0.00	60	60	60	7.00	Success_I	VI;D
2	Night	No	S	M	3.20	3	90	57	33	16.77	0.00	100	100	100	3.50	Unsuccess_col	NR;VI;D;C
3	Day	Yes	S	M	2.90	5	74	55	19	27.24	20.00	0	2	10	6.00	Success_I	D
4	Day	Yes	V	S	1.95	4	95	68	27	57.51	25.00	0	2	10	16.00	Success_L	None
5	Night	No	T	L	0.70	3	83	70	13	51.21	0.00	30	60	30	20.00	Success_I	VI
6	Night	No	V	M	51.15	3	53	53	0	39.27	31.93	30	60	30	0.00	No_attempt	NO
7	Night	No	T	V	3.65	5	106	85	21	23.82	0.00	100	100	100	4.45	Unsuccess_col	MV;VI;D;C

Figure 4.11: Portion of dataset with R column added

A preliminary investigation was conducted to better understand the additional column introduced to the data using histograms. By grouping the simulations by classes, it was possible to see how the rule infractions are distributed among the classes and which are predominant. It becomes important to understand whether the violation of one traffic rule precludes the violation of other rules and how certain features can be at the origin of rule violations. For this purpose, the correlation matrix for these values was once more calculated. The categorical values of the variable CLASS were transformed into numerical values using the OneHotEncoder function from Python's sklearn package. This creates a binary column for each category and returns a sparse matrix or dense array. Thus, new columns were created, one for each class with the value 0 if the class is matching and 1 otherwise, just to calculate the correlation matrix.

4.3 Results

4.3.1 Features Statistics

The input parameters in the simulation scenario are variably chosen and have an impact on the physics and duration of the manoeuvres. Because the driver always wants to overtake, his decision will never be influenced by any factor other than the simulation's outcome. However, in the non-simulated real world, one can speculate what circumstances create these results. Following that, some important or anomalous data from each characteristic will be studied and interpreted. Reasons for these outcomes will be studied given the simulator's characteristics and limitations. A last step will consist of speculations on real-world events.

Tables 4.6 and 4.5 show the measurement results for the static and dynamic features, respectively. The abbreviations Min and Max represent, respectively, the minimum and maximum value of each feature, and the symbol σ represents the standard deviation. Also in the second table, the abbreviations af and rf represent the absolute and relative frequency, respectively.

Table 4.5: Statistics of Dynamic Features.

DYNAMIC FEATURES		STATISTICS				OUTCOME
		Min	Mean	Max	σ	
Current Speed		58.00	77.93	99.00	11.89	Successful (legal)
		56.00	84.32	107.00	14.44	Successful (illegal)
		53.00	75.50	103.00	16.71	Non-successful (no collision)
		53.00	87.75	117.00	17.66	Non-successful (collision)
		53.00	55.10	65.00	2.64	No attempt to overtake
Speed of Preceding vehicle		50.00	57.38	78.00	7.49	Successful (legal)
		50.00	63.72	97.00	12.61	Successful (illegal)
		45.00	61.16	87.00	10.87	Non-successful (no collision)
		50.00	64.01	95.00	12.20	Non-successful (collision)
		52.00	57.03	90.00	10.67	No attempt to overtake
Speed Difference		4.00	20.55	34.00	7.45	Successful (legal)
		3.00	20.60	52.00	11.62	Successful (illegal)
		0.00	14.34	35.00	9.41	Non-successful (no collision)
		1.00	23.74	51.00	13.07	Non-successful (collision)
		0.00	4.07	30.00	7.56	No attempt to overtake
Distance between C and P		40.26	55.77	75.60	9.21	Successful (legal)
		20.76	39.06	85.29	10.19	Successful (illegal)
		21.12	39.40	74.01	11.13	Non-successful (no collision)
		16.77	31.50	68.49	11.54	Non-successful (collision)
		23.91	43.78	72.78	13.11	No attempt to overtake
Occupancy Lane Rate (%)		0.00	11.95	50.00	15.84	Successful (legal)
		0.00	16.95	50.00	15.58	Successful (illegal)
		0.00	18.85	50.00	15.28	Non-successful (no collision)
		0.00	16.96	60.00	16.60	Non-successful (collision)
		27.59	31.54	36.88	3.05	No attempt to overtake
Weather	Rain	0.00	37.93	100.00	49.38	Successful (legal)
		0.00	38.80	100.00	40.30	Successful (illegal)
		0.00	42.50	100.00	43.40	Non-successful (no collision)
		0.00	37.79	100.00	41.46	Non-successful (collision)
		0.00	33.67	100.00	37.28	No attempt to overtake
	Wind	10.00	44.14	100.00	44.44	Successful (legal)
		10.00	42.99	100.00	36.35	Successful (illegal)
		10.00	46.56	100.00	31.57	Non-successful (no collision)
		10.00	42.35	100.00	37.34	Non-successful (collision)
		10.00	38.00	100.00	33.36	No attempt to overtake
	Fog	2.00	3.90	7.00	2.47	Successful (legal)
		2.00	49.53	100.00	33.52	Successful (illegal)
		2.00	38.38	100.00	39.40	Non-successful (no collision)
		2.00	47.97	100.00	32.50	Non-successful (collision)
		2.00	58.93	100.00	27.18	No attempt to overtake

Table 4.6: Statistics of Static Features.

STATIC FEATURES	STATICS										OUTCOME						
Night and Day	Day					Night											
	af		rf			af		rf									
	20		0.21			9		0.05			Successful (legal)						
	35		0.37			82		0.45			Successful (illegal)						
	21		0.22			47		0.26			Non-successful (no collision)						
	14		0.15			18		0.10			Non-successful (collision)						
	4		0.04			26		0.14			No attempt to overtake						
	94					182					Total of simulations						
Horizon Line	Yes					No											
	af		rf			af		rf									
	7		0.27			22		0.09			Successful (legal)						
	10		0.38			107		0.43			Successful (illegal)						
	2		0.08			30		0.12			Non-successful (no collision)						
	4		0.15			64		0.26			Non-successful (collision)						
	3		0.12			27		0.11			No attempt to overtake						
26					250					Total of simulations							
Type of ego vehicle	S		M		L		V		T								
	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf							
	3	0.06	7	0.13	4	0.10	5	0.07	10	0.16	Successful (legal)						
	23	0.48	23	0.43	13	0.32	33	0.46	25	0.40	Successful (illegal)						
	4	0.08	7	0.13	6	0.15	9	0.13	6	0.10	Non-successful (no collision)						
	10	0.21	13	0.24	10	0.24	19	0.27	16	0.26	Non-successful (collision)						
	8	0.17	4	0.07	8	0.20	5	0.07	5	0.08	No attempt to overtake						
	48		54		41		71		62		Total of simulations						
Type of Preceding vehicle	S		M		L		V		T		MC		B				
	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf			
	5	0.10	6	0.09	5	0.10	7	0.13	6	0.12	0	0	0	0.00	Successful (legal)		
	21	0.41	29	0.45	18	0.37	21	0.38	26	0.52	2	0.4	0	0.00	Successful (illegal)		
	7	0.14	6	0.09	8	0.16	6	0.11	2	0.04	2	0.4	1	1.00	Non-successful (no collision)		
	16	0.31	15	0.23	11	0.22	17	0.30	9	0.18	0	0	0	0.00	Non-successful (collision)		
	2	0.04	8	0.13	7	0.14	5	0.09	7	0.14	1	0.2	0	0.00	No attempt to overtake		
	51		64		49		56		50		5		1		Total of simulations		
Waiting Time	Min				Mean			Max			σ						
	0.05				0.44			2.63			0.63						
	0.05				1.30			5.03			0.98				Successful (illegal)		
	0.05				2.38			23.17			4.19				Non-successful (no collision)		
	0.05				1.79			9.33			1.86				Non-successful (collision)		
	1.12				8.00			20.05			7.75				No attempt to overtake		
	0.05				2.18			23.17			3.74				Total of simulations		
Overtaking Time	Min				Mean			Max			σ						
	3.33				7.68			15.00			3.10						
	1.67				4.36			16.00			2.37				Successful (illegal)		
	4.58				11.16			26.72			5.16				Non-successful (no collision)		
	0.75				2.80			14.78			2.81				Non-successful (collision)		
	0.00				0.00			0.00			0.00				No attempt to overtake		
	0.00				4.64			26.72			4.17				Total of simulations		
Number of Vehicles	3				4			5			6						
	ra		rf		ra		rf		ra		rf		ra			rf	
	9		0.15		6		0.08		11		0.14		3		0.05		Successful (legal)
	29		0.49		35		0.46		34		0.44		19		0.30		Successful (illegal)
	8		0.14		11		0.14		8		0.10		5		0.08		Non-successful (no collision)
	9		0.15		15		0.20		19		0.25		25		0.39		Non-successful (collision)
	4		0.07		9		0.12		5		0.06		12		0.19		No attempt to overtake
	59				76			77			64				Total of simulations		

Table 4.6 can be found in figure format in Figure A.1 of Appendix A.

The static features correspond to the total of the observed values according to the different features DN, HL, TE, TP and NV. It can be noted that the data is unbalanced, which is a key characteristic of categorization methods. Balancing the data provides the same amount of information for each feature, allowing us to forecast each class more accurately. Now, in this synthetic generated dataset, one can see that there is a significant difference between daytime and nighttime scenes (94 versus 182), as well as scenes with and without the horizon line (250 versus 26). Later on, a data balancing will be considered. It is currently feasible to compare the recorded values for characteristics that calculate the lowest, maximum, and average. When there is no attempt to overtake, the waiting time, which indicates the expected time to overtake, has a high mean value (8.00). It assumes this high value since, for these cases, the vehicle waits until the end of the simulation to overtake. That is, the value is the same as the simulation's duration. In scenarios where the overtaking is successful and legal, the ego vehicle's driver waits an average of 0.44 seconds before initiating the manoeuvre. In a real-world circumstance, the reverse would be expected: a longer time spent perceiving the surrounding components prior to the manoeuvre would indicate their safety. The overtaking time feature is also quite intuitive. It is not possible to compute how long it takes in cases when there is no overtaking, hence the number is set to 0. Except for this class, the smallest number relates to an unsuccessful overtaking with collision (2.80). Because the manoeuvre ends when both vehicles collide and ego vehicle does not return to the original lane, this number is also expected. For unsuccessful overtakes without crashes, the highest rating, 11.16, is recorded. These situations occur when, for some reason, the car does not return to the original lane and instead stays in the left lane until the simulation is completed, which takes longer.

When it comes to dynamic features, for cases of unsuccessful overtaking with collisions, the average value of the ego vehicle speed is high (87.75). When reported for unsuccessful overtaking without crashes, the value is much lower (75.50). This fact may indicate that the overtaking vehicle's speed may have an impact on the occurrence of collisions. On the other hand, the value for the Successful (illegal) class is also high (84.32), so it is not possible to distinguish a successful overtaking from an unsuccessful one by the speed of the ego vehicle alone. When there is no attempt to overtake, the average ego vehicle speed is recorded at its lowest (55.10). It can be assumed that in real-world settings, when a vehicle's speed is low, it will not attempt to pass another vehicle because the latter will most probably drive at a higher speed.

In turn, the average value of the overtaken vehicle speed measurements is consistent, ranging between 57 and 64. The lowest recorded values relate to no overtaking attempt (57.03) and successful and legal overtaking (57.38). The highest value is assigned to the class of unsuccessful overtaking with collisions (64.01). Because they are in such a tiny range, it is clear that the feature Speed of the preceding car does not indicate the class by itself. The speed difference is an important feature to consider when studying the classes. The highest reported value corresponds to unsuccessful overtaking situations involving crashes (23.74). Because this value is so near to the values associated with successful overtaking (20.55 and 20.60), it's assumed that the speed difference feature isn't a factor in determining whether an overtaking attempt is successful or not. The value achieved in circumstances where no overtaking attempt, on the other hand, is much

lower than the others (4.07). This could imply that in situations where the speed differential between the ego vehicle and the other vehicle is insignificant, the driver does not feel the need to overtake.

Distance between C and P is another important factor in feature analysis. It can be expected that the greater the safety distance between vehicles, the more likely that overtaking will be successful. The distances of successful and legal overtaking events have an average value of 55.77. This score is substantially higher than the others, implying that it could be a deciding factor in whether an overtaking is lawful. The lowest rating denotes an unsuccessful overtaking attempt that resulted in a collision (31.50). As a result, it is reasonable to predict that collisions are likely at low safety distances. The high value of 43.78 in non-overtaking scenarios could also indicate that if a car keeps a large distance from the automobile ahead, it has no desire to overtake and will always follow behind it.

The values of the lane occupancy rate to the left of the vehicle that wants to overtake are very similar to each other. The highest and most discordant value is 31.54, which corresponds to no overtaking situations. The lane occupancy does not influence the driver's decision to overtake in the simulation scenario, but it can be a significant effect in real-world settings. Because the lane is more constrained, the driver may be afraid of the manoeuvre due to the risk of colliding.

The observed weather conditions appear to have an impact on class selection. In terms of precipitation, all values are similar, with the exception of the smallest value (33.67), which corresponds to circumstances where there is no overtaking. In the actual world, the driver may be concerned about his overtaking manoeuvre in adverse weather conditions. Because the simulation does not bring into question the driver's judgement to overtake, the data does not support the statement. The same is true for the wind values. The fog values, on the other hand, are distinct. Cases with successful and lawful overtaking are assigned a lower value of 3.90, whereas cases without overtaking are assigned a lower value of 58.93. These values are due to the fact that fog represents a decrease in visibility which, when overtaking, leads the driver to violate traffic rules, and overtaking is considered illegal. It is expected that in real-world conditions with high fog content, the driver will choose not to overtake for this reason.

4.3.2 Data Visualization

The data collected during each simulation is composed of 22 variables, clearly detailed in Section 4.1.3. Table 4.7 shows the data recorded in frame 70 (F) of the first simulation (S) at 3.5 seconds (TS). Three vehicles with ids 488, 489 and 490 are involved, where 488 is the id of the ego vehicle (IDego). These numbers can be checked in the columns Dim, L, V, D and A. Considering only the ego vehicle, its dimension is then 6.27m long, 2.39m wide and 2.1m high (D). It is located in lane -7 having its x and y coordinates equal to 351.35 and 251.58 values, respectively (L). It has a speed of 66 km/h (V) and an acceleration of 76.28 m/s² (A). The direction of the wheels registers -0.2 on the x-axis and -3.62 on the y-axis (D). In this frame, lane -7 indicates a maximum speed of 90 km/h (MV). The lane to the right of the ego vehicle

has the solid type (RT) and to the left the broken type (LT). Considering lane width, the lane in which the ego vehicle is travelling has 3.5 meters wide (LW), the lane to its right has 0.5m (LWR) and the lane to its left has 3.5m (LWL). A collision was recorded in this frame between the ego vehicle and the vehicle with ID 489 (C). As for the weather conditions, the percentage of precipitation, fog and wind registered the same value of 60% (Prec, Fog, Wind). It is nighttime (DN) and there is no horizon line (HL). Finally, the OV column shows a 0 value, which means that in this frame there was no overtake attempt.

S	F	TS	IDego	Dim	L	V	D	A	MV	RT	LT	LW	LWR	LWL	C	Prec	Fog	Wind	DN	HL	OV
1	70	3.5	488	(488,6.27,2.39,2.1); (489,5.37,1.80,1.57) (490,4.86,2.03,1.65)	(488,351.35,251.58,-7); (489,355.82,251.68,-7) (490,318.8,251.61,-7)	(488,66); (489,79) (490,113)	(488,-0.2,-3.62); (489,0.35,-0.43) (490,0.1,0.23)	(488,76.28); (489,40.06) (490,0.55)	90	Solid	Broken	3.5	0.5	3.5	489	60	60	60	Night	No	0

Table 4.7: Information about frame 70 of simulation 1 extracted from dataset.

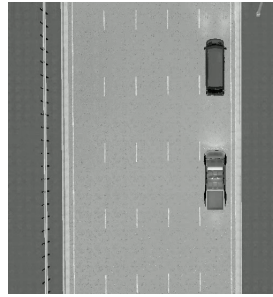


Figure 4.12: Screenshot of simulation 1 video at frame 70.

The dataset corresponding to the static and dynamic features collected in each simulation was used. The dataset is shown using the python code below, and Figure 4.13 shows its output.

```

1 import pandas as pd
2
3 df = pd.read_csv("features.csv")
4 df = df.drop('S', axis=1)
5 df

```

Listing 4.4: Code sample to display a dataset portion.

S	DN	HL	TE	TP	WT	NV	SE	SP	DSEP	D	OLR	PREC	FOG	WIND	OT	CLASS
1	Night	No	T	L	3.35	4	98	54	36	36.84	0.00	60	60	60	7.00	Success_I
2	Night	No	S	M	3.20	3	90	57	33	16.77	0.00	100	100	100	3.50	Unsuccess_col
3	Day	Yes	S	M	2.90	5	74	55	19	27.24	20.00	0	2	10	6.00	Success_I
4	Day	Yes	V	S	1.95	4	95	68	27	57.51	25.00	0	2	10	16.00	Success_L
5	Night	No	T	L	0.70	3	83	70	13	51.21	0.00	30	60	30	20.00	Success_I
6	Night	No	V	M	51.15	3	53	53	0	39.27	31.93	30	60	30	0.00	No_attempt
7	Night	No	T	V	3.65	5	106	85	21	23.82	0.00	100	100	100	4.45	Unsuccess_col

Figure 4.13: Features Dataset Portion.

'CLASS' column shows the classifications detailed in Section 4.1.2. The abbreviation Success_L corresponds to legal successful overtaking, Success_I corresponds to illegal successful overtaking,

Unsuccess_col corresponds to unsuccessful overtaking with collisions, Unsuccess_ncol corresponds to unsuccessful overtaking without collisions, and No_attempt corresponds to cases where there is no attempt to overtake. In order to standardize the data, the categorical values DN, HL, TP, and TE, were mapped to numerical values. For the DN (Day/Night) feature, the code assigned numerical values so that Day maps to 0 and Night maps to 1. The same logic was followed for the HL (Horizon Line) variable, where Yes maps to 1 and No maps to 0. For the values of the TP and TE columns, vehicles S (small), M (medium), L (large), V (vans), T (truck), MC (motorcycle) and B (bicycle) map respectively the values from 0 to 6.

To begin data visualization, the histogram and the frequencies of each class were displayed as shown in Figure 4.14.

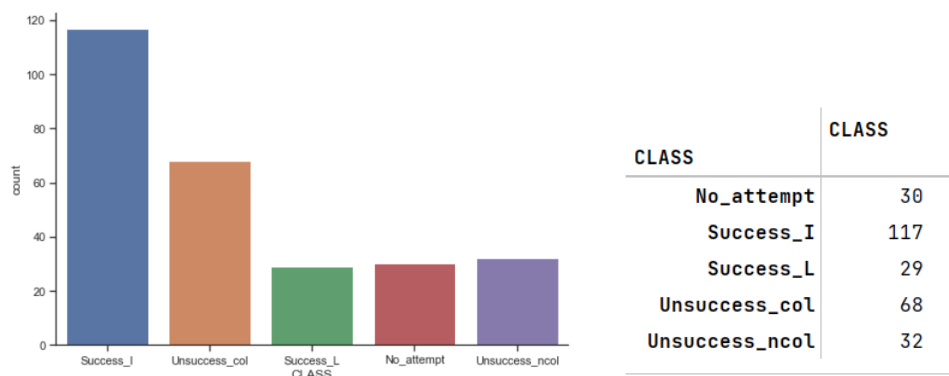


Figure 4.14: Frequency of classes.

Out of 276 simulations, 30 are classified as No_attempt, 117 as Success_I, 29 as Success_L, 68 as Unsuccess_col and 32 as Unsuccess_ncol. It can be seen that the class with the highest frequency is Success_I and the one with the lowest frequency is Success_L. The total numbers of occurrences for each class would not be as dissimilar if some overruns considered Success_I were considered Success_L. This might be because many factors have to happen for the manoeuvre to be considered legal. A histogram for each feature, illustrated in Figure 4.15, was created to better understand how the data was distributed.

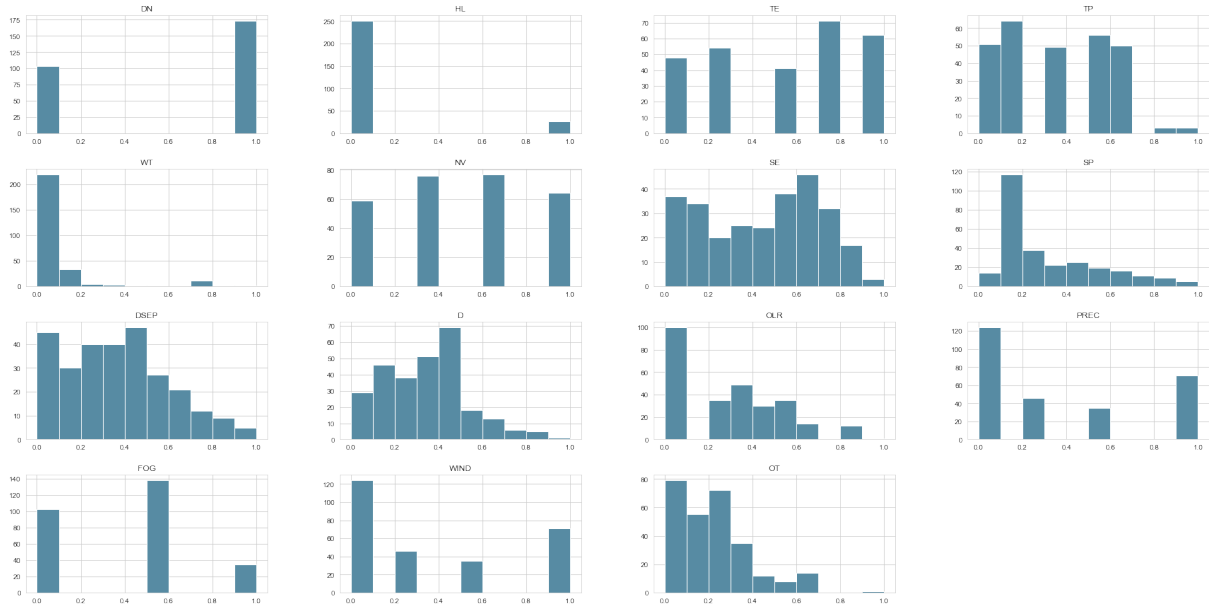


Figure 4.15: Histogram for each feature.

The histogram corresponding to the WT and OT features represents a right-skewed distribution. The data distribution suggests that high values occur with a low frequency. In other words, in the case of the WT variable, the majority of the cases represented in the dataset have a very short time to overtake. The histogram in the OT feature, on the other hand, demonstrates that the overall time of the overtaking manoeuvre is quite low in the majority of situations. This is because the OT value registers a low value in 47 percent of circumstances $((30+68+32)/276*100)$: in unsuccessful overtaking and when there is no desire to overtake, the value is 0. The histogram corresponding to the NV (number of vehicles) feature represents a normal distribution. This means that points on one side of the average are as likely to occur as on the other side of the average. About half of the cases have 4 or 5 vehicles present, and another half have 3 or 6 vehicles, the remaining features have a random distribution. In a random distribution histogram, it can be the case that different data properties were combined. In the histogram corresponding to the OLR feature, it is possible to verify that in 36% $(100/276*100)$ of the cases the average occupancy rate during the simulation is between 0% and 5%. In the histogram corresponding to the SP feature (current speed of ego vehicle), about 43% $(120/276*100)$ of the cases, the ego vehicle drives at a speed between 50 and 55 km/h. As seen in the histograms of the DN and HL features, there are many more simulations in nighttime scenario than daytime scenario (about 66% versus 34%) and many more simulations without the presence of horizon line than with the presence (about 91% versus 9%). This is owing to the fact that the horizon line is only present in one of the weather situations considered in the simulations, ClearSunset. In terms of nighttime circumstances, they account for 5 of the 9 presets used, or around 56%. When the scenarios in each simulation are picked at random, the chances of the features with the biggest number of instances being represented are higher.

Another way of visualizing and interpreting the data was through a box-plot. Figure 4.16

shows the box plot diagram for each feature.

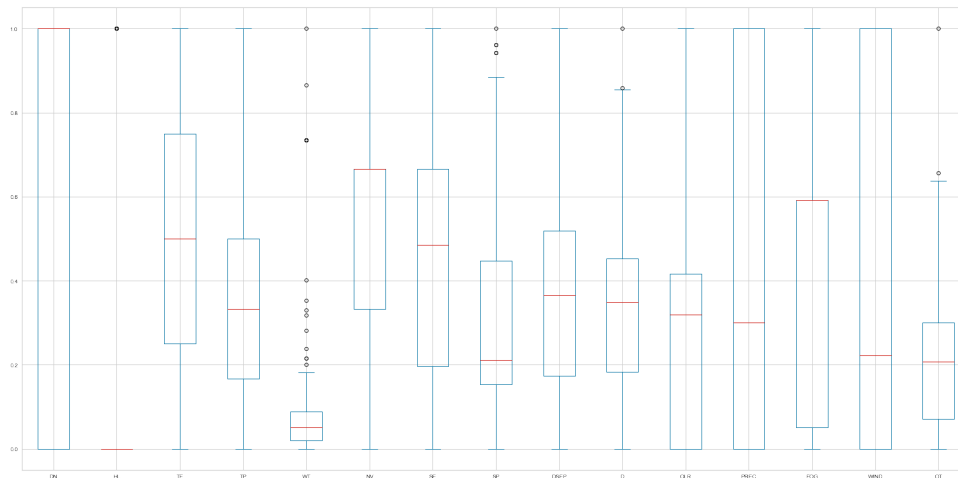


Figure 4.16: Box plot diagram for each feature.

It can be observed that, for example, the values of waiting time (WT) are not dispersed, unlike the variables PREC or WIND. In the case of feature OT and SP, the median is closer to the bottom quartile. When this happens, then the distribution is not symmetric (normally distributed) but positively skewed (skewed right). The mean is greater than the median, so the data constitute higher frequency of high valued scores. Many outliers are found in features WT, SP, D and OT. These data should be discarded when they are known to have been entered/measured incorrectly or when they affect assumptions or create significant associations.

A swarm plot is another way of plotting the distribution of an attribute or the joint distribution of a couple of attributes. The box plot and swarm plot for the SE feature grouped by classes were compared. The code below corresponds to the creation of the box plot (line 1) and swarm plot (line 2). Their outputs are illustrated by Figures 4.17a and 4.17b, respectively.

```
1 sns.boxplot(x="CLASS", y="SE", data=df)
2 sns.catplot(x='CLASS', y='SE', hue='CLASS', kind='swarm', data=df)
```

Listing 4.5: Code sample to display box and swarm plots.

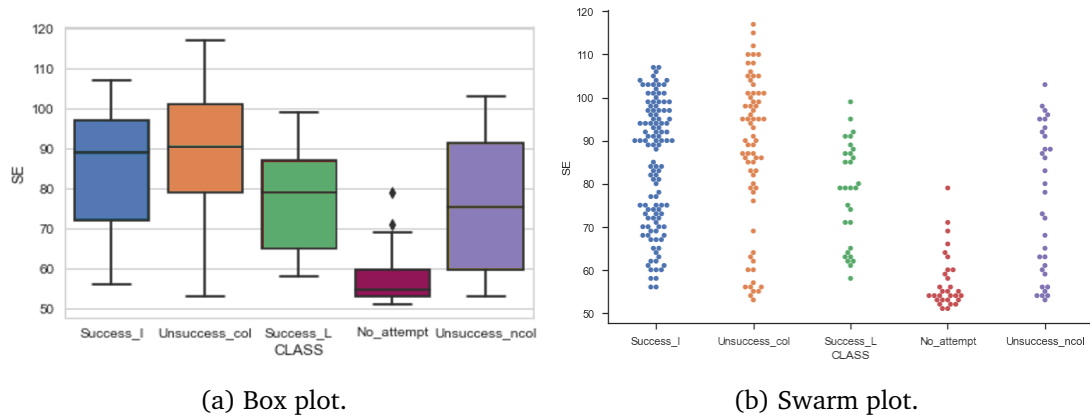


Figure 4.17: Box and swarm plots for SE feature grouped by classes.

It can be seen that the swarm plot supports the box plot by highlighting the scattering and clustering zones of the points. It is interesting to see that the box plot of the No_attempt class is comparatively short. This suggests that, in general, vehicles that do not attempt to overtake have a speed between 55 and 60km/h.

The correlation matrix was then analysed to uncover relationships among variables and between variables and class. Figure 4.18 shows the matrix, whereas Figure 4.19 shows the corresponding p-values, which will be evaluated alongside the matrix. The null hypothesis in this case is a statement that there is no relation between the two variables being compared. Because diagonal elements indicate each variable's association with itself, they will always equal 1.

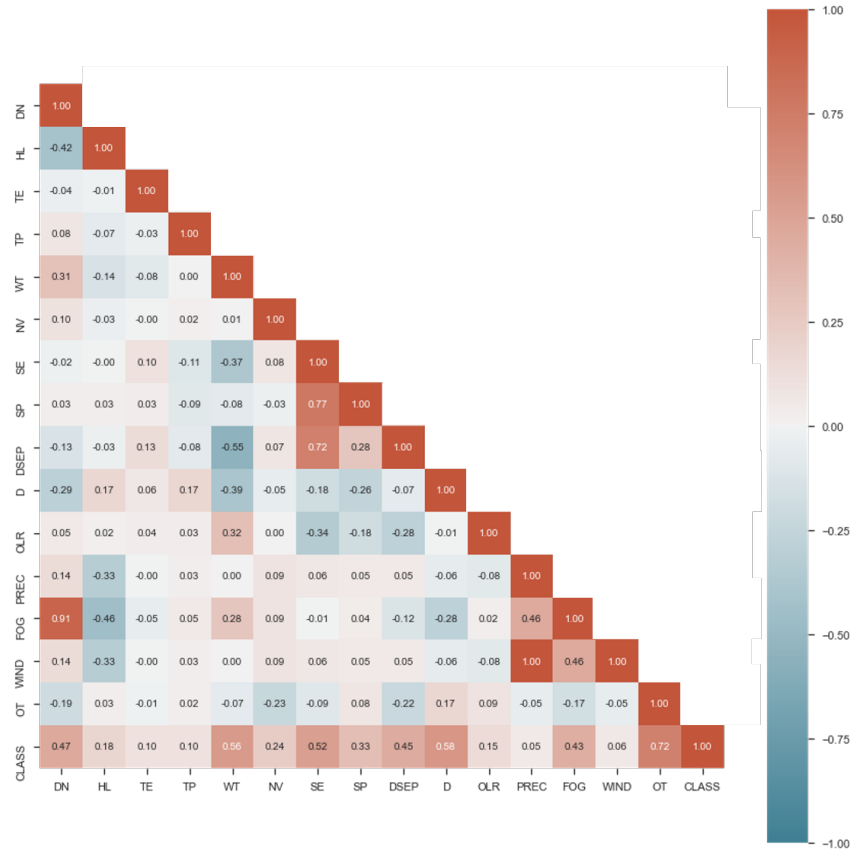


Figure 4.18: Correlation matrix.

	DN	HL	TE	TP	WT	NV	SE	SP	DSEP	D	OLR	PREC	FOG	WIND	OT	CLASS
DN	0.0	0.0	0.5614	0.2008	0.0	0.0995	0.6006	0.2558	0.0322	0.0	0.1094	0.0199	0.0	0.0199	0.0001	0.0058
HL	0.0	0.0	0.8259	0.2713	0.0238	0.5769	0.9969	0.7293	0.6027	0.0039	0.7923	0.0	0.0	0.0	0.6152	0.6502
TE	0.5614	0.8259	0.0	0.6509	0.164	0.9918	0.088	0.9171	0.0419	0.3537	0.7593	0.9351	0.4061	0.9351	0.6994	0.2259
TP	0.2008	0.2713	0.6509	0.0	0.9551	0.721	0.0635	0.2076	0.1751	0.01	0.2394	0.6395	0.3701	0.6395	0.5402	0.1463
WT	0.0	0.0238	0.164	0.9551	0.0	0.9314	0.0	0.6118	0.0	0.0	0.0	0.9973	0.0	0.9973	0.0174	0.0059
NV	0.0995	0.5769	0.9918	0.721	0.9314	0.0	0.284	0.9257	0.3293	0.1555	0.2025	0.1401	0.1454	0.1401	0.0003	0.6896
SE	0.6006	0.9969	0.088	0.0635	0.0	0.284	0.0	0.0	0.0	0.0746	0.0	0.2273	0.8353	0.2273	0.0972	0.0001
SP	0.2558	0.7293	0.9171	0.2076	0.6118	0.9257	0.0	0.0	0.0	0.0063	0.0	0.239	0.1542	0.239	0.2457	0.1576
DSEP	0.0322	0.6027	0.0419	0.1751	0.0	0.3293	0.0	0.0	0.0	0.9686	0.0	0.1674	0.054	0.1674	0.0019	0.0002
D	0.0	0.0039	0.3537	0.01	0.0	0.1555	0.0746	0.0063	0.9686	0.0	0.767	0.4249	0.0	0.4249	0.0	0.0101
OLR	0.1094	0.7923	0.7593	0.2394	0.0	0.2025	0.0	0.0	0.0	0.767	0.0	0.1342	0.3282	0.1342	0.061	0.0019
PREC	0.0199	0.0	0.9351	0.6395	0.9973	0.1401	0.2273	0.239	0.1674	0.4249	0.1342	0.0	0.0	0.0	0.5302	0.7737
FOG	0.0	0.0	0.4061	0.3701	0.0	0.1454	0.8353	0.1542	0.054	0.0	0.3282	0.0	0.0	0.0	0.0007	0.0181
WIND	0.0199	0.0	0.9351	0.6395	0.9973	0.1401	0.2273	0.239	0.1674	0.4249	0.1342	0.0	0.0	0.0	0.5302	0.7737
OT	0.0001	0.6152	0.6994	0.5402	0.0174	0.0003	0.0972	0.2457	0.0019	0.0	0.061	0.5302	0.0007	0.5302	0.0	0.0
CLASS	0.0058	0.6502	0.2259	0.1463	0.0059	0.6896	0.0001	0.1576	0.0002	0.0101	0.0019	0.7737	0.0181	0.7737	0.0	0.0

Figure 4.19: P-values.

The correlation between the variables WIND and PREC has a value of 1 indicating a perfect positive correlation. The corresponding p-value of 0 indicates that the null hypothesis (there is no significant relationship between the two variables) can be rejected. This occurs because each precipitation value correlates to the same wind value in the presets specified for the meteorological circumstances of the simulations. Another example of positive correlation is the 0.91 correlation between the FOG and DN variables, with a p-value of 0. It means that higher fog intensity is significantly linked to nighttime scenarios. Since it is expected that the simulated

data would accurately reflect reality, it is important to discuss potential relationships that are discovered. As such, the two cases presented above are distinct in that the strong relationship of the first example is expected as opposed to the strong relationship of the second case. In the non-simulated real world, strong wind is often observed on days of heavy rain. In the second case, it is believed that in the real world this relationship would not be so evident since clear nights without fog can often be found.

The SE and DSEP variables, on the other hand, have a positive correlation of 0.72. Then, because the p-value is 0, it can be concluded with a high level of confidence that for high ego vehicle speeds, a large difference in speeds between the ego vehicle and the overtaken vehicle is observed. As for the negative correlations, there is no perfectly negative one. With a value of -0.55, the correlation between the DSEP and WT variables is the most significant. The relevance of this relation is confirmed by the p-value of 0. In this case, it turns out that the waiting time to overtake is greater when two cars are travelling at similar speeds, i.e., the speed difference is low. In a real-life scenario, this statement makes sense since for an overtaking to occur and the wait time to be reduced, the vehicle must accelerate and move closer to the car in front, increasing the DSEP value. For near-zero correlation values, such as the case of the 0.075 value for the NV and SE variables, indicates that they're basically not correlated. The p-value of 0.284 confirms this value, indicating that there is about 28 percent possibility that the results were obtained by chance. As a result, the number of vehicles in the simulation and the ego vehicle speed have almost no relationship. Analyzing this example might be interesting. In the simulated situation, the number of cars in the space under consideration for simulation has no impact on the ego vehicle's speed. In a real-world scenario, it would be reasonable to assume that the ego vehicle speed would be lower with a heavier traffic flow. However, when comparing only the two variables - number of vehicles and ego vehicle speed - there is no information about the speeds of the other vehicles. Assuming that all vehicles are moving at constant and similar speeds, the speed of the ego vehicle would not need to be changed. This value of correlation could be increased if the simulation window was reduced and the number of possible cars was increased.

4.3.3 Traffic Rules

Figure 4.20 depicts a histogram of the total number of simulations divided by the classes that violated each traffic rule.

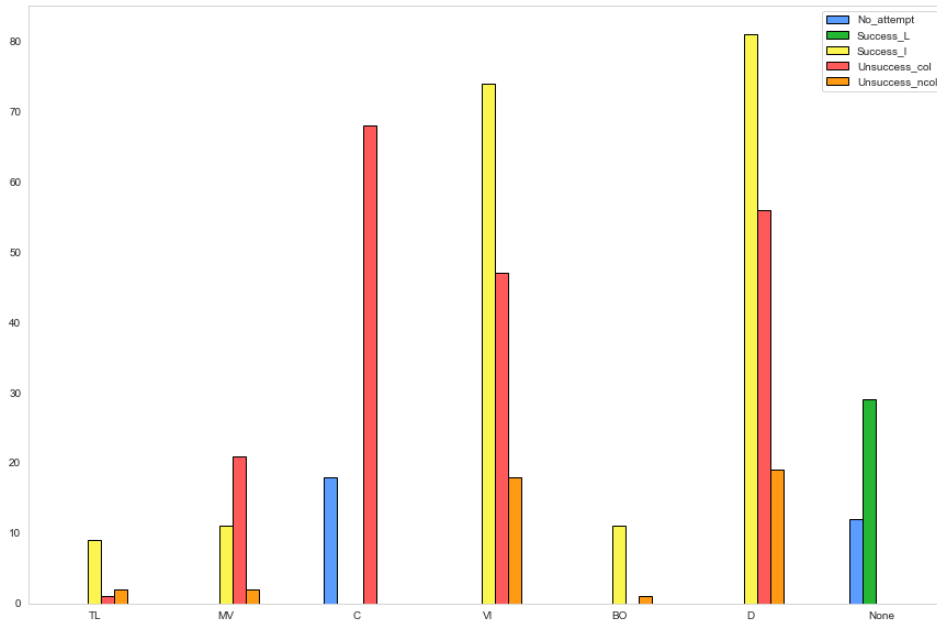
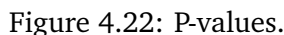


Figure 4.20: Histogram for violated traffic rules grouped by class.

It can be observed that there were no rules violations in the Success_L class, thus it is considered legal. In the cases where there is no overtaking attempt there are 18 cases of collisions and 12 cases where no rule is broken. The rest of the collisions occur in the Unsuccess_col class. For the Success_I class all rules are found to be infringed, with the disrespect of the safety distance being the most observed. The correlation matrix calculated for the new dataset is represented in Figure 4.21. The observations were supported by the corresponding p-values present in Figure 4.22.



With a value of 0.78, B_VI and DN have a high positive correlation. This means that high B_VI values correspond to high DN values. Because the variables have binary values, 1 denotes a violation of the rule in question, whereas 0 denotes the absence of such a violation. As a result, it can be demonstrated that nighttime scenarios have a high possibility of causing limited visibility. B_VI and FOG have a 0.69 correlation coefficient. This demonstrates that conditions with high fog content result in poor visibility. The variables B_C and OT, as expected, have a negative relationship with the value of -0.61. High B_C values, such as 1 (indicating the occurrence of collisions), result in a shorter total simulation duration. This is because each simulation ends when the vehicle returns to its initial lane after overtaking the vehicle, but in the presence of a

collision, the simulation ends upon collision. Another obvious relationship is the one between the variables B_D and D , which corresponds to tiny distance values for big values of B_D (the safety distance is too small). The SE and SP variables have a positive connection with the B_D variable. This means that in situations where both the ego vehicle and overtaken vehicles are travelling at high speeds, there is a good chance that the ego vehicle will not maintain a safe distance from the automobile in front. The variables D and B_C have a negative correlation of -0.23 . That is, for high values of B_C (presence of crashes), the distance between the ego vehicle and the vehicle in front is low.

Moving on to an analysis of the correlation of the rules with each other and with the class, the correlation matrix shows that the categorization of an overtaking can be done with high accuracy just by looking at the rules B_C or B_None . This is because the variable B_C when equal to 1, 78% relates to the `Unsuccess_col` category. The same is true for B_None , which when equal to 1, 64% concerns the `Success_L` category. According to the relationship between the variables B_VI and B_D , which has a value of 0.46, the lack of visibility causing the violation of the overtaking without sufficient vision rule ($B_VI = 1$) leads in many cases to overtaking without minimum safe distance. This can be explained because the driver's small field of vision can force him to get too close to the vehicle in front in order for it to enter his field of vision. The variables B_MV and B_C have a positive association, although having a low correlation coefficient of 0.25. That is, high B_MV values (maximum speed limit disrespected) equate to high B_C values. It's easy to see how disobeying speed restrictions increases the likelihood of a collision. The low positive correlation (0.12) between variables B_C and B_D is an interesting fact that is not expected. The fact that B_C correlates negatively with variable D , means that in cases where there are collisions, the distance between vehicles tends to be low even if it does not violate the two-second rule. One can conclude that even respecting the distance rule, a collision can happen, and therefore some authorities regard two-second rule as inadequate, and recommend a three-second rule [109]. To be more precise, of the cases where collisions are recorded (86 in total), 35% are two-second rule (30 in total). Another significant correlation is the negative relationship of -0.50 between the variables B_None and B_D . This means that for high values of B_None , i.e, there is no infraction of any traffic rule considered in this work, the value of B_D is low, i.e, the safety distance is respected. It then confirms what was observed in the study of important features, that distance is an important factor for overtaking safety. Additionally, it can be seen that the variable B_None displays a negative correlation with all the other traffic infractions, since the other variables have low values of 0 for high values of 1 in B_None (no rule is broken). The negative correlation of -0.46 between the `No_attempt` class and the variable SE shows that for high values of `No_attempt`, i.e., when there is no overtaking at all, the ego vehicle presents a low speed. Another significant value is the correlation between the class `Success_L` and the variable D . This demonstrates that in cases where the overtaking is successful and legal, the distance between the vehicles tends to be large. The almost zero coefficients of the correlation between $WIND$ and all classes shows that the variable $WIND$ has almost no influence on the classification of an overtaking.

For a better understanding of the participation of each feature in the infraction of the rules, the histogram of the average values of each feature was analysed when rules B_C and B_VI are

infringed, value equal to 1, and when there is no infraction ($B_None = 1$). These histograms are shown in Figure 4.23, below.

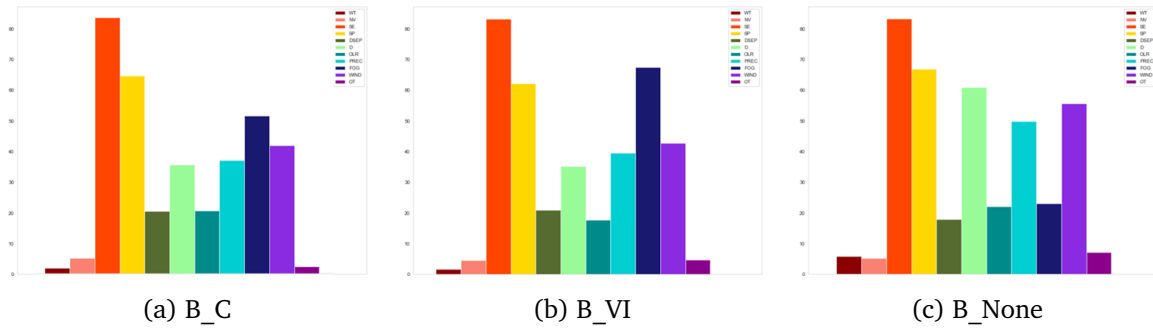


Figure 4.23: Histograms of the mean values of each continuous feature for variables B_C , B_V , and B_None of value 1.

The variable having the highest value in all the histograms is SE. In the instance where no rule is broken, Figure 4.23c, the ego vehicle's average speed is around 70 km/h. The average value of SE is around 80 km/h for infractions of overtaking without visibility, Figure 4.23b, and the presence of crashes, Figure 4.23a. The average SP value for cases with violations is around 62 km/h, while the average SP value for cases without violations is around 55 km/h. Here, one can see that small increases in the speed of the intervening cars, can result in traffic rule infractions or even crashes. The fog values, in turn, reveal high levels for the two infractions, around 50% for B_C and 67% for B_VI , whereas the value reduces to 20% in the cases without infraction. As previously stated, the value of the space between cars has an impact on the safety of overtaking. When there are no infractions, the value is higher, averaging 51 metres, whereas when there are infractions, the value reduces to around 35 metres. The figures also show that variables like WIND, for example, have no effect on rule infraction because its average value is between 40 and 45 percent in all three scenarios.

4.3.4 Data Validation

In data science, data validation is the process of ensuring that data is correct, useful, and accurate. It is critical to confirm the quality and accuracy of data before entering it into a system. Validation rules must be used on a regular basis to ensure that data is correct, meaningful, and secure. Because of data integration systems, data validation can now be a relatively quick procedure. Validation processes can be incorporated and automated. It is dangerous to make decisions based on data that are flawed and do not fully reflect the scenario at hand by skipping data validation phase. The artificial intelligence sector suffers from a scarcity of high-quality data. Only 3% of companies' data fulfils basic quality criteria, according to a report published by the Harvard Business Review in 2017 [118]. Comparisons with real data are used to partially validate the synthetic data obtained by the simulator. Two studies will be used in this follow-up to compare their data obtained in the real world to the synthetic data collected in simulation.

This comparison will aid in data validation and reliability.

The first study is conducted by Asaithambi and Shravani [16], a prestigious study in the field of highway overtaking. The primary goal of this research was to look into the overtaking behaviours of automobiles on undivided highways in mixed traffic. The observations were done on Indian roadways, which are very diverse in nature with substantial differences in vehicle static and dynamic properties. Vehicles do not adhere to rigorous lane discipline and will take up any available space on the road. The overtaking characteristics were observed and the data extracted and analysed were acceleration characteristics, overtaking vehicle speeds, overtaking time, overtaking distances, safe distance required for overtaking, flow rates, overtaking frequencies, overtaking strategy types, and types of overtaking vehicles and overtaking vehicles. The difference in vehicle speeds and total overtaking time were found to have a negative correlation in the study by Asaithambi and Shravani [16]. As can be seen in Figure 4.18, the same is true in this investigation, with the value -0.19. Asaithambi and Shravani [16] estimated the total time spent during overtaking, just like is done in this study. The results can be compared (shown in Figure 4.24a) with this work (shown in Figure 4.24b) by distinguishing between different types of cars. Two-wheeler, car, LCV (light commercial vehicle), heavy vehicle, and bus were the study's categories. The results can be compared by associating with this work's vehicles: Car with medium car, LCV with small car, and Heavy vehicle with truck. In Figure 4.24b, the number 1 represents medium cars, the number 0 represents small cars, and the number 4 represents trucks.

Vehicle type	Total overtaking time (s)				TE	OT mean	min	max	std
	Mean	Min	Max	Std. Dev.					
Two-wheeler	7.5	3.4	17.4	2.3	0	5.127308	2.00	10.67	2.786451
Car	7.5	3.6	17.4	2.3	1	4.877667	2.00	11.33	2.701469
LCV	8.9	5.4	16.0	2.7	4	5.009143	1.67	15.00	3.105728
Heavy vehicle	10.3	5.3	24.7	3.4					
Bus	9.1	4.9	20.5	3.4					

(a) Values obtained in the study of Asaithambi and Shravani [16].

(b) This work's values.

Figure 4.24: Total overtaking time for different types of vehicles.

Despite the fact that the values are not similar, the synthetic values obtained are not completely unrealistic. One apparent similarity is that the average total overtaking time for small vehicles (0 in this study and LCV in the other) is higher than for medium automobiles (1 in this study and Car in the other) in both cases. The same cannot be said for the truck and heavy vehicle categories, which do not have a greater mean value than the other categories in this analysis, as it happens in the study in comparison. The differences in the results of the two studies could be due to a variety of factors. The cars in the two experiments are not the same dimension, and the variables surrounding the car are different. The heavy traffic flow on Indian streets will a priori influence the time it takes for a vehicle to overtake, which will be higher.

The study of Mahmud et al. [99] looked at the dangers of overtaking movements on a bidirectional, undivided two-lane highway in a diverse traffic environment in Bangladesh. To

better comprehend the major aspects related with the chance of conflict and its severity during the overtaking manoeuvre, a risk severity model was constructed. This paper is a part of his PhD dissertation, where Mahmud [97] analysed the speed of the vehicles during overtaking by representing their minimum, maximum, and average speed. The value was presented according to the type of vehicle when overtaking and when being overtaken. The outcomes are shown in Figure 4.25a. Because the simulations don't have a Bus category, only Truck and Car categories were compared. In this work's case, Car was considered as medium vehicle (1) and truck (4). Figure 4.25b depicts this work's values.

Type of Vehicle	When overtaking				When overtaken			
	Min	max	Average	St. Dev.	Min	max	Average	St. Dev.
Bus	50	102	68	11	40	85	64	16
Truck	45	95	62	10	40	75	48	6
Car	55	90	71	11	55	80	68	18

(a) Values obtained in the study of Mahmud [97].

TE	SE mean	min	max	std
1	83.000000	54	106	16.249238
4	81.209677	53	110	16.692137

TP	SP mean	min	max	std
1	62.000000	50	95	12.351981
4	61.880000	50	87	10.925798

(b) This work's values.

Figure 4.25: Speed profile (km/h) of the overtaking and overtaken vehicles.

Car category always exhibits the highest speed in the findings of the study under comparison, whether overtaking or being overtaken. This is also true in this work's case. The speed values recorded by the simulator are not very different from the values observed in this study. Therefore, it can be considered that the speed values are close to values recorded on real roads. This study, like the previous one, documented the overtaking time values for each type of vehicle. Figure 4.26a shows the minimum, maximum, and average times for two types of overtaking: flying and accelerative. As in the study by Asaithambi and Shravani [16], flying overtaking refers to a situation in which there are no other vehicles in the area and the vehicle overtaking another slow moving vehicle does so without having to slow down, and accelerative overtaking refers to a situation in which the overtaking vehicle reduces its speed until it has enough space in the other lane to overtake. In this case, the study's categories of Large Truck, Microbus, and Vehicle were considered to be equivalent to truck (4), van (3), and medium car (1), respectively. The correspondence was established entirely on the basis of the vehicles' dimensions. As expected, in the study of Mahmud [97], heavy vehicles, such as large truck and large bus, show longer overtaking times. In this work's results, it can be observed that this is also true. The values of the two studies are also similar, so there are no absurd values that compel disagreement with the real world.

Vehicle type	Minimum	Maximum	Average	Std.Dev.
Flying overtaking				
Large Bus	5.6	20.1	12.1	3.5
Large Truck	7.7	13.8	11.1	1.9
Microbus	6.4	12.1	9.3	4.0
Car	6.9	17.4	10.6	4.1
Accelerative overtaking				
Large Bus	6.8	19.9	11.4	3.4
Large Truck	5.9	14.5	11.5	2.8
Microbus	9.4	16.3	11.6	3.2
Car	5.5	12.3	8.0	1.7
Pickup	10.4	11.3	10.9	0.7

(a) Values obtained in the study of Mahmud [97].

TE	OT mean	min	max	std
1	4.877667	2.00	11.33	2.701469
3	4.639737	1.67	16.00	2.620028
4	5.009143	1.67	15.00	3.105728

(b) This work's values.

Figure 4.26: Overtaking time for different vehicle types (seconds).

4.3.5 Discussion of the Results

To determine if the work's goals were achieved, it is necessary to critically analyse and interpret the results. The design of data generation was divided between: the creation of scenarios in the CARLA simulator to represent motorway overtaking; the choice of the variables to be collected during the simulation; the strategies for the manoeuvre classification; the creation of additional features to the collected dataset and the collection of traffic rules regarding the overtaking manoeuvre. Afterwards, statistical methods were applied to explore and interpret the dataset and the road code was coded into first order logic rules to attribute legality and safety to the overtaking manoeuvre. Finally, the data were validated using published studies that make use of real data.

The information collected by the simulator was chosen based on the literature reviewed and the Portuguese Highway Code. All scenarios were designed taking into account Portuguese roads ignoring factors such as road surface quality, car brand and style, presence of obstacles on the road and objects other than vehicles. The simplicity of the scenarios is therefore a factor that reduces the similarity of the simulated world with the real world. However, the simple scenarios are able to provide relevant information and their complexity can be increased in future work.

Due to the limitations of the simulator, it was not possible to collect information considered essential to the study of overtaking. Therefore, questions such as “does the weight of a vehicle influence overtaking?” and “is a driver's field of vision reduced in nighttime environments?” remained unanswered. Even so, in most cases it was found that the relationship between the variables collected is close to what was expected.

One of the major limitations of this work is to explain the results based on the real world. Although it is possible to resemble some results of the class distribution to reality, such as the fact that there are more successful overtaking than those with collisions, most of these would only be possible to validate with tests in the real world ensuring similar conditions to those of the simulated world. This limitation is due to the hybrid state that currently exists, i.e. the junction of autonomous and non-autonomous cars on the roads that causes the uncertainty and

unpredictability of manoeuvres. Considering a perfect world where all cars are autonomous and have the same algorithm, it would be possible to eliminate all classes except Success_L.

The logic rules applied resulted from the codification of Portuguese traffic laws to ensure that, throughout the manoeuvre, the driver does not break the law. If for an overtaking to be successful, it implies the infraction of some rule, then it should not be considered. Because of the hybrid system that exists and the unpredictability of other drivers' actions, the rules alone ensure the legality of the manoeuvre but do not guarantee its safety. Even so, it can be expected that their implementation will prevent accidents and improve the overall driving landscape.

Data validation was a necessary step to add confidence to the results obtained in this work. Approximating the values of this work with the real data compared allowed to find errors in the calculations and discard nonsense values. However, the works in the literature used as a means of comparison use real data collected in different circumstances to this work. Factors such as the number of lanes of the road, the number of directions, the country in which the data is collected, the traffic flow, the civility of drivers, among others, are decisive and can drastically change the values of the variables.

Chapter 5

Prediction of Overtaking Scenarios

The second phase of the work is the prediction of overtaking scenarios presenting the materials and methods used and the results obtained. First, a thorough study of the features created in the previous chapter is performed and the importance of each one for the classification of the manoeuvre is calculated using a prediction model. Finally, moments prior to the overtaking are considered in order to obtain a sequence of actions that provides the success of the manoeuvre. For a clearer understanding of the phases that constitute this chapter, a pipeline is shown in Figure 5.1 illustrating the order and the output/intent of each task.

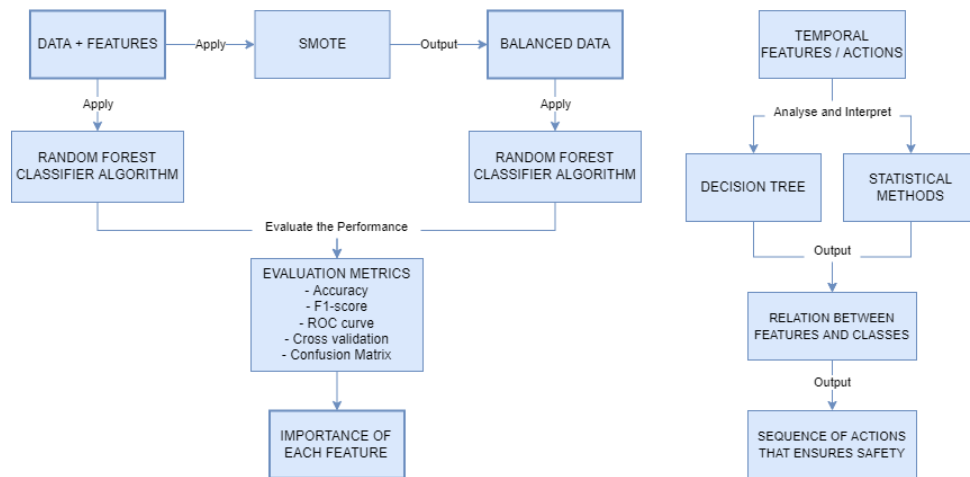


Figure 5.1: Pipeline of the second phase of the work.

5.1 Materials and Methods

5.1.1 Prediction and Features Importance

This phase is dedicated to predicting the final class of overtaking using the Random Forest Classifier model. This model was chosen because it effectively handles large datasets and

produces good predictions (with a higher level of accuracy) that can be understood easily. The importance of each feature included in the prediction can be determined once the model has proven great performance. Features importance refers to techniques that assign a score for all input features for a given model. The scores indicate how significant and pertinent each element is. A higher score here indicates that this feature will have a bigger impact on the model that predicts the final class. For this, the Random Forest Classifier algorithm was applied, and its performance was evaluated through a confusion matrix and some metrics and evaluation models such as accuracy, cross-validation, precision, recall, f1-score and AUC-ROC curve.

Random Forest Classifier is a supervised learning method that uses many decision trees to make predictions that can be used for classification tasks such as classifying an overtaking manoeuvre successful or unsuccessful based on the types of vehicles involved. When performing a classification task, each decision tree in Random Forest votes on one of the classes to which the entry belongs. Once all trees have come to a conclusion, the random forest counts which class has the most populous vote, and that class will be what the random forest produces as a prediction. In the case of regression, instead of determining the most populous vote, the random forest will average the results of each decision tree [34].

The following is a description of the random forest algorithm:

1. N is the assumed number of observations. These N observations will be sampled at random with replacement.
2. Take into account M input variables. At each node, a number m , where $m < M$, will be chosen at random from the total number of features, M . To split the node, the best split of these m variables is utilized, and this value remains constant as the forest grows.
3. Each decision tree in the forest is grown to its largest extent.
4. The forest will generate a prediction based on the combined predictions of the forest's trees (either a majority or an average vote).

Random forest is a very popular algorithm that produces, most of the time, an excellent result. The motivation for using this algorithm is the use of the “bagging” method, which argues that combining learning models improves the end result overall. It is called “random” because its technique employs two random processes: bootstrapping and random feature selection. Bootstrapping ensures that we are not using the same data for every tree, so in a way it helps our model to be less sensitive to the original training data. The random feature selection helps to reduce the correlation between the trees. The random forest approach also makes it very simple to measure the relative importance of each feature in the prediction. The `RandomForestClassifier` function [146] provided by the Sklearn library measures the importance of a feature by looking at how much the tree nodes that use that feature reduce impurity across all trees in the forest. After training, it automatically determines this score for each feature and scales the results so that the sum of all importances is equal to 1. By looking at feature importance, it can be decided which features should be discarded since they do not contribute enough, or anything, to the prediction

process. As a result, a random forest method was used to determine the importance of each feature for overtaking manoeuvre classification and make classification prediction by studying historical data and patterns. Thanks to the sklearn's function `train_test_split`, the data was divided into 2 groups: training and testing (75% for training and 25% for testing). The `stratify` parameter makes a split so that the proportion of values in the sample equals the proportion of values in the given set. Therefore, if 40% of the data are classified as `Success_L`, *stratify = y* will ensure that the random split has 40% of the data classified as `Success_L`. The OT feature was not taken into account in this study because it is an output after overtaking and not an input to the decision at the time of overtaking. Accordingly, X represents all columns except S, CLASS and OT, while Y represents only the classification column (CLASS). The random forest classifier was trained on the training set, while the test set was used to evaluate the performance of the model.

```
1 X = df_overtakes.iloc[:,1:-1].values
2 y = df_overtakes["CLASS"].values
3
4 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .25,
    stratify=y)
```

Listing 5.1: Code sample to split the dataset into test and training.

The data was then standardized, putting all the features on the same scale. This ensures that attributes with higher values do not have a significant impact on the model. To calculate the score for each feature and guarantee that each has a mean of 0 and variance of 1 in the dataset, the sklearn function `StandardScaler` was utilized.

```
1 sc = StandardScaler()
2 X_train_std = sc.fit_transform(X_train)
3 X_test_std = sc.transform(X_test)
```

Listing 5.2: Code sample to scale the data.

After preparing the data, the random forest classifier method was applied.

```
1 clf=RandomForestClassifier()
2 clf.fit(X_train, y_train)
```

Listing 5.3: Code sample to apply `RandomForestClassifier` method.

The model accuracy was initially evaluated using the split between training and testing to evaluate the model's performance. Since there are many model parameters that are set manually, these can adjust until the model performs optimally and cause risk of overfitting the test set. This way, knowledge about the test set can “leak” into the model and evaluation metrics no longer report on generalization performance. To address this issue, a portion of the data set can be used as a “validation set”, and after the evaluation done on this set is successful, the final evaluation can be done on the test set. The amount of samples that can be used to learn the model is substantially reduced when the available data is split into three sets, and the outcomes may vary

depending on the random selection made for the pair of sets (train, validation). As a result, the cross-validation method was employed, which disregards the requirement of the validation set. This procedure consists of dividing the data into k sets. A model is trained using $k-1$ of the folds as training data, and the resulting model is validated on the remaining part of the data. The performance measure reported by k -fold cross-validation is then the average of the values computed in the loop. The advantage of using this validation model is that it avoids the waste of data that occurs when fixing an arbitrary validation set [50]. The two procedures: split between training and testing, and cross-validation, were computed 30 times (the minimum number to ensure that the findings are significant) and the accuracy values of each were averaged.

The function “score” computes the count of accurate predictions and is equivalent to the “accuracy_score” function from the Python’s package called metrics. The subset accuracy is 1.0 if the complete collection of predicted labels for a sample strictly matches the true set of labels; otherwise, it is 0.0 [106]. A metric from the Metrics package called balanced accuracy score was used to determine whether the model was prioritizing a particular class [147]. Balanced accuracy is the average between the metrics sensitivity (total of predicted positives) and specificity (total of predicted negatives). As a result, imbalanced datasets can be accounted, as a model will receive a worse balanced accuracy score if it only predicts accurately for the majority class in the dataset.

Another way to evaluate the model performance is from a confusion matrix. The matrix shows the combination of the real and predicted classes. Each row of the matrix corresponds to the predicted classes and the columns correspond to the real classes, or vice versa, and it is possible to find out which classes are more likely to be mixed up. To deal with the imbalanced dataset, the Synthetic Minority Oversampling TEchnique (SMOTE) technique was applied. SMOTE is a method of oversampling that enables the creation of artificial samples for the minority classes. The K-Nearest Neighbours algorithm is the basis of the concept. Thus, the difference between a sample and one of its k nearest neighbours is obtained and multiplied by some random value in the range of (0, 1). The value obtained in the preceding procedure is then added to create a new synthetic sample [44]. Image augmentation is a popular method for enhancing the training samples in image classification. In the context of SMOTE, a synthetic sample is equivalent to an augmented image. After applying the technique, a new dataset was generated, now balanced, and the model accuracy and a new confusion matrix were calculated.

To complement the previous remarks, an AUC-ROC curve was plotted for the multiclass model using the One Vs All methodology. Since there are 5 classes, the result will be one ROC curve for Success_L ranked against all others, another ROC curve for Success_I against all others, and so on. Other prediction models such as Support Vector Classification and Linear Support Vector Classification were also tested, and their results were compared. The goal of the support vector machine algorithm is to find a hyperplane in an N -dimensional space (N – the number of features) that distinctly classifies the data points. Support Vector Classification implements the “one-versus-one” approach for multiclass classification and Linear Support Vector Classification is another implementation of Support Vector Classification for the case of a linear kernel and implements the “one-vs-the-rest” multiclass strategy.

After the evaluation metrics that supported the reliability of the model were applied, the features were organized by their degree of importance in the classification task.

The results of this step-by-step experiment are shown in Section 5.2.1.

5.1.2 Temporal Data

After defining the traffic rules to be considered in this work, it was realized that rules 4 to 8 need information over time. Therefore, it is essential to investigate moments before the overtaking. Up until now, measurements of each feature were obtained when the car made the decision to overtake. The final classification of the manoeuvre, however, can be influenced by the moments prior to this one. Therefore, this phase involved collecting the ego vehicle's actions throughout the simulation and dividing them into the two key moments of the manoeuvre: the first is before the ego vehicle changes lanes and the second is when it has already changed lanes and is attempting to get ahead of the second vehicle. The second moment, when the ego vehicle changes lanes a second time to return to the initial lane, is not considered since already in the other lane the simulation ends because the overtaking manoeuvre is already complete. Thus, there is not enough rich data to study during this moment. The first moment is considered for all classes and the second moment is considered for all except the No_attempt class, because there is no first lane change. The actions taken into consideration included varying the speed in 2km/h and the distance from the ego vehicle wanted to overtake. From these, the following features were developed:

- Speed Increased (SPEED_I): the total speed increased throughout the simulation (km/h)
- Speed Decreased (SPEED_D): the total speed reduced along the simulation (km/h)
- Times Speed Increased (Δ SPEED_I): total number of times the ego vehicle increased speed (int)
- Times Speed Decreased (Δ SPEED_D): total number of times the ego vehicle reduced speed (int)
- Distance Increased (DISTANCE_I): : total increase of the distance to the car in front (m)
- Distance Decreased (DISTANCE_D): : total decrease of the distance to the car in front (m)
- Times Distance Increased (Δ DISTANCE_I): total number of times the ego vehicle increased distance to the car in front (int)
- Times Distance Decreased (Δ DISTANCE_D): total number of times the ego vehicle decrease distance to the car in front (int)

In order to understand how features and classes are related at the first moment of the manoeuvre, a decision tree was deployed and later, a correlation matrix was once more applied.

A decision tree constantly divides the data into points that belong to each class by using the features in the data set to generate yes/no questions. The dataset is divided and a new node is added to the tree each time a question is asked. One branch of the tree has all the data points corresponding to the "yes" answer to the question that the rule at the previous node implied. The remaining data points are located at a node on the other branch. The goal is to keep dividing the dataset and applying rules until there are no more rules to apply or data points left. Finally, the last nodes in the tree, leaf nodes, are assigned a class. The process of class assignment is done by an algorithm that attempts to completely separate the data set so that all leaf nodes belong to a single class. The majority of the time, leaf nodes are mixed, meaning that not all data points have the same class, and the algorithm assigns them the most common class among the corresponding data [26].

5.2 Results

5.2.1 Prediction and Features Importance

In this phase, the RandomForestClassifier algorithm was applied to predict the final class of overtaking and to obtain the most important features in the overtaking manoeuvre classification task. At the beginning of the experiment, the model was applied to the initial dataset excluding some non-relevant columns and divided into training and test groups. After then, predictions were produced and their accuracy was evaluated. The score method calculates the random forest's average accuracy on the given data. The training data is used to evaluate performance in the first phase, and then the test data is used to evaluate performance in the second phase. The model achieves a 100% accuracy on the training data and an accuracy of about 62% on the test data. Since the accuracy of the training data is so high and different from the accuracy of the test data, it can be said that the model is modelling the training data too well and is not generalizing what it is learning.

```
1 preds = clf.predict(X_test)
2 print (clf.score(X_train, y_train))
3 print (clf.score(X_test, y_test))
4
5 Out:
6 1,0
7 0.6231884057971014
```

Listing 5.4: Code sample to obtain the average accuracy of the random forest.

The cross-validation technique was also applied in order to estimate how accurate this model is in practice, that is, its performance for a new data set. In the code below (Listing 5.5), we can see that the technique was applied by splitting the data, fitting a model and computing the score 10 consecutive times (with different splits each time). For this work's case, the obtained value was approximately 0.66, which means that for new data, the model is able to make predictions

with 66% accuracy, which is a very low value which means that the data is not good for this model.

```
1 scores = cross_val_score(clf, X, y, cv=10)
2 print(scores.mean())
3
4 Out:
5 0.6589947089947089
```

Listing 5.5: Code sample to apply cross-validation technique.

The two procedures: split between training and testing and the cross-validation, were computed 30 times and the mean and standard deviation between the accuracy values of each was calculated. The values are similar, the first being 0.64 ± 0.04 and the second 0.66 ± 0.09 .

Another way to evaluate the performance of the model is through a confusion matrix. Table 5.1 shows the confusion matrix calculated for the applied model.

Table 5.1: Confusion Matrix.

		Predicted				
		Success_L	Success_I	Unsuccess_col	Unsuccess_ncol	No_attempt
Actual	Success_L	7	0	0	0	0
	Success_I	3	24	2	0	0
	Unsuccess_col	0	8	8	1	0
	Unsuccess_ncol	2	6	0	0	0
	No_attempt	1	3	0	0	4

A confusion matrix includes true positive values (TP), false negative values (FN), false positive values (FP), and true negative values (TN). The number of overtakes correctly classified by the model is represented by the diagonal values of the matrix: 7, 24, 8, 0 and 4. The values that aren't on the diagonal represent data that has been improperly categorized. Looking more closely at the confusion matrix and taking into account the low accuracy value, it can be realized that the method used does not produce reliable results. For example, the model does not correctly predict any cases corresponding to the Unsuccess_ncol class.

When a dataset is imbalanced, it provides misleading classification accuracy. It was possible to verify that the model was accurately predicting the majority class by obtaining the low value of 0.48 for balanced accuracy score. As a result of the distribution of classes being so unequal due to the large number of simulations classed as Success_I, the algorithm is biased in favour of that dominant class. To address this issue, the SMOTE method was employed. Figure 5.2a shows that, prior to the use of SMOTE, the Success_I class dominated, accounting for around 42% of the simulations (117 out of 276). After applying the SMOTE technique (Figure 5.2b), all classes now have the same number of occurrences as the class that was dominant, 117, originating 309 new cases.

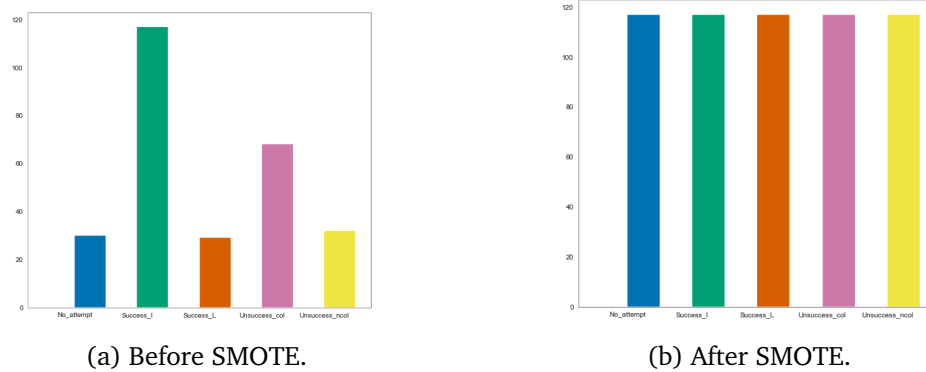


Figure 5.2: Number of occurrences of each class before and after applying SMOTE.

Given this new dataset, the previous procedure of splitting the data into a training and test set and applying the RandomForestClassifier method with the standard data was performed and the accuracy on the test data increased to about 82%. The cross-validation technique was once again applied and the previous value of 0.66 is now increased to 0.82. The two procedures: split between training and testing and the cross-validation, were once again computed 30 times and the mean and standard deviation between the accuracy values of each were calculated. The values are again similar, the first being 0.81 ± 0.03 and the second 0.84 ± 0.007 .

The code excerpt below describes the process of applying SMOTE and the classification prediction method.

```

1 from imblearn.over_sampling import SMOTE
2
3 sm = SMOTE()
4 X_res, y_res = sm.fit_resample(X, y)
5 X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size =
    .25, stratify=y_res)
6
7 clf=RandomForestClassifier()
8 clf.fit(X_train, y_train)
9 preds = clf.predict(X_test)
10
11 print (clf.score(X_train, y_train))
12 print (clf.score(X_test, y_test))
13
14 scores = cross_val_score(clf, X_res, y_res, cv=10)
15 print(scores.mean())
16
17 Out:
18 1,0
19 0.8231292517006803
20 0.8218390804597699

```

Listing 5.6: Code sample to apply SMOTE and RandomForestClassifier method.

The confusion matrix generated with the new dataset is shown in Table 5.2.

Table 5.2: Confusion Matrix.

		Predicted				
		Success_L	Success_I	Unsuccess_col	Unsuccess_ncol	No_attempt
Actual	Success_L	25	3	0	1	0
	Success_I	4	18	5	2	0
	Unsuccess_col	1	3	23	2	0
	Unsuccess_ncol	0	2	1	25	2
	No_attempt	0	0	0	0	30

The confusion matrix generated with the new dataset is more accurate and has a higher percentage of true positive values. In this case, for example, in cell [0,1], the positive actual value is Success_L, but the model predicts 3 times as negative Success_I. The model incorrectly predicts 3 cases (cell [2,1]) as Success_I, and the correct prediction would be Unsuccess_col. The distribution of these values for the Success_I class is represented in Table 5.3 and the results of the metrics applied to the confusion matrix are shown in Table 5.4.

Table 5.3: Distribution of true and negative values for Success_I class.

		Predicted				
		Success_L	Success_I	Unsuccess_col	Unsuccess_ncol	No_attempt
Actual	Success_L	TN	FP	TN	TN	TN
	Success_I	FN	TP	FN	FN	FN
	Unsuccess_col	TN	FP	TN	TN	TN
	Unsuccess_ncol	TN	FP	TN	TN	TN
	No_attempt	TN	FP	TN	TN	TN

Table 5.4: Results of the metrics applied to confusion matrix.

Class	True Positive (TP)	False Positive (FP)	False Negative (FN)	True Negative (TN)	Precision	Recall	F1-score	Macro-Average F1-Score	Accuracy
Success_L	25	5	4	111	0.83	0.86	0.85	0.82	0.82
Success_I	18	8	11	108	0.69	0.62	0.65		
Unsuccess_col	23	6	6	110	0.79	0.79	0.79		
Unsuccess_ncol	25	5	5	112	0.83	0.83	0.83		
No_attempt	30	0	0	115	0.94	1.00	0.97		

The precision values (shown in Table 5.4), which indicates how close or far away a given set of measurements is from its true value, are relatively high, with the No_attempt class having the highest value. As a result, the model can be trusted when it classifies an overtake as No_attempt the majority of the time. Equally, high recall values mean that the model can correctly identify positive cases. The lower precision and recall values for the Success_I class means that the model does not classify that class as correctly. This may be due to the fact that this class is assigned an

overtake due to a combination of factors that differ in each scenario. The `Unsuccess_col` class's high precision and low recall value demonstrate that the model does not always predict the class accurately, but when it does, it does so with considerable confidence. To conclude, the F1-score confirms that the model performs very well in all classes except `Success_I`, which has a relatively high F1-score value but lower than the others. This indicates that another way should be found to improve the performance for this class, perhaps by augmenting the training dataset with more successful but illegal simulations.

To complement, ROC curves were calculated for the Random Forest Classifier model and for two other models Support Vector Classification and Linear Support Vector Classification, and their results were compared. ROC curves are typically used in binary classification to study the output of a classifier. For this work, multiple classes are considered and therefore the One Vs All methodology was implemented. The ROC curve shows the trade-off between sensitivity (or TPR) and specificity ($1 - \text{FPR}$). Classifiers that have curves closer to the upper left corner (AUC value close to 1) indicate better performance. As a baseline, a random classifier is expected to provide points along the diagonal ($\text{FPR} = \text{TPR}$). The closer the curve is to the 45 degree diagonal of the ROC space, the less accurate the test. The result of the metric implementation is shown in Figure 5.3.

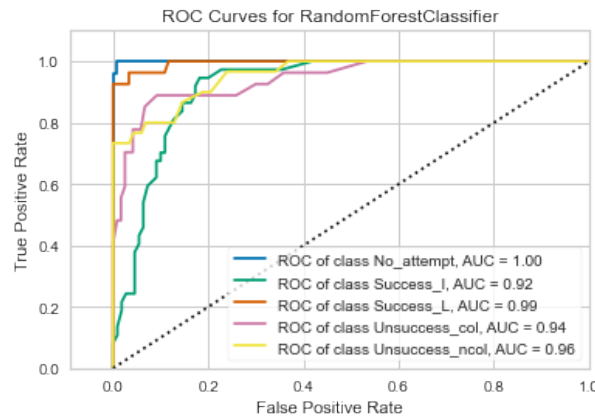


Figure 5.3: Plot of AUC-ROC curve for Random Forest Classifier.

For this work, it can be seen that the curves for all classes have a very high AUC value close to 1. The highest value corresponds to the `No_attempt` class, which means that the model classifies it perfectly. It can be concluded, then, that the Random Forest Classifier model performs very well in distinguishing between the classes. Other prediction models such as Support Vector Classification and Linear Support Vector Classification were also tested. The models don't perform well for the classes taken into consideration in this work, as evidenced by the lower AUC values. The Linear Support Vector Classification model's less successful outcomes demonstrate once more that the data for some attributes are not distributed linearly. Figure 5.4 shows the AUC values for each class, all lower than the `RandomForestClassifier` model.

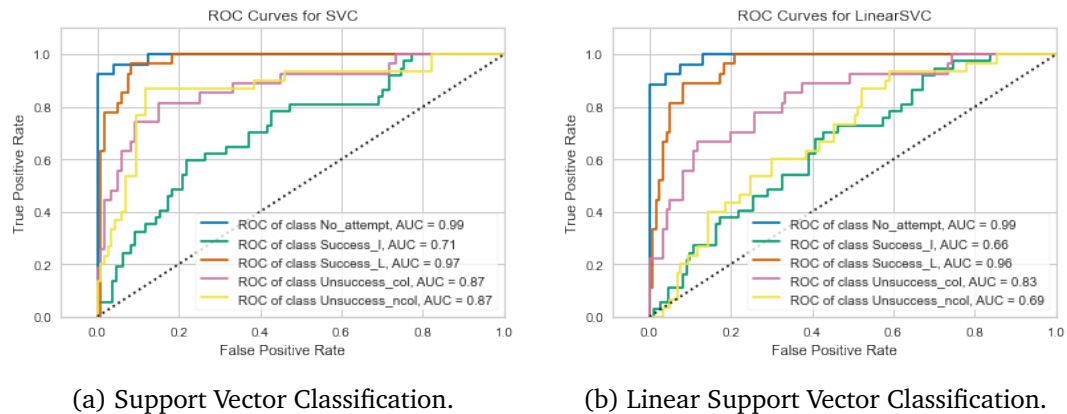


Figure 5.4: AUC-ROC curve for SVM and LSVM metrics.

Finally, the model determined which characteristics were the most relevant for identifying the overtaking manoeuvre. The graph generated from the code extract below is shown in Figure 5.5.

```

1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 feature_imp = pd.Series(clf.feature_importances_, index=X_train.columns.values).
    sort_values(ascending=False)
5
6 %matplotlib inline
7 sns.barplot(x=feature_imp, y=feature_imp.index)
8
9 plt.xlabel('Feature Importance Score')
10 plt.ylabel('Features')
11 plt.title("Important Features")
12 plt.legend()
13 plt.show()

```

Listing 5.7: Code sample to generate the features importance graph.

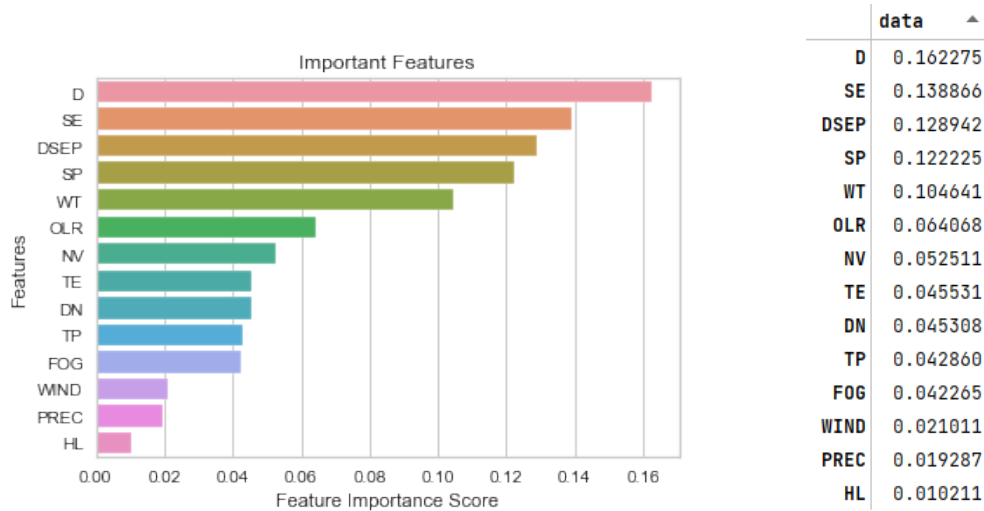


Figure 5.5: Feature importance scores provided by the model.

It could be said that the following characteristics contribute the most to categorization distinction in this order: D (distance between ego vehicle and overtaken vehicle), SE (current speed of ego), SP (current speed of overtaken vehicle), WT (waiting time to overtake) and DSEP (speed difference between ego vehicle and overtaken vehicle). The remaining features have a not so significant importance in the classification of the overtaking manoeuvre.

To prove the importance of feature D in the classification of an overtake, a prediction of the class was made given the data referring to the moment of the decision. For the same dataset but only modifying the value of feature D, for the case where the value is low, 35 (not respecting the 2-second rule), the predicted class is `Unsuccess_col`. That is, there is a strong probability of a collision. By increasing this value by just 9 meters, now respecting the rule, the predicted class becomes `Success_I`. The result was obtained using the code excerpt below:

```

1 small_D = [0,0,0,0,1.7,4,80,60,40,35,0,0,10,0]
2 b = (small_D - np.min(small_D))/np.ptp(small_D)
3 b = b.tolist()
4 print(clf.predict([b]))
5
6 Out:
7 Unsuccess_col
8
9 large_D = [0,0,0,0,1.7,4,80,60,40,44,0,0,10,0]
10 b = (large_D - np.min(large_D))/np.ptp(large_D)
11 b = b.tolist()
12 print(clf.predict([b]))
13
14 Out:
15 Success_I

```

Listing 5.8: Code sample to prove the importance of feature D in the classification of an overtake.

5.2.2 Temporal Data

Once the strategy to identify the actions taken by the driver in the two key moments of the manoeuvre (described in Section 5.1.2) was defined, the implementation phase was started. The decision tree was computed using the DecisionTreeClassifier function from the sklearn package. The resulting decision tree is shown in Figure 5.6 below.

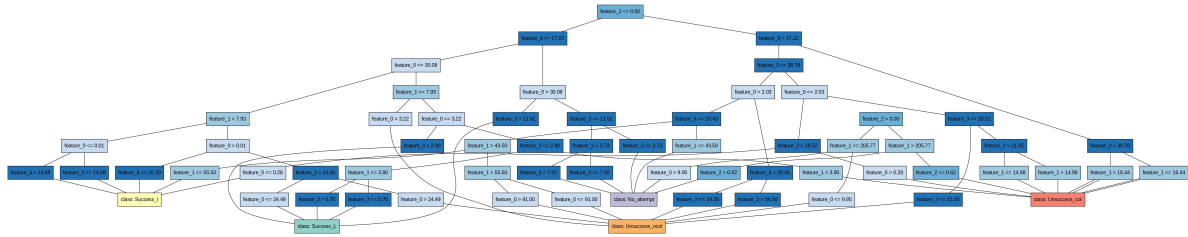


Figure 5.6: Decision tree built from the features presented.

In the tree, feature_1 corresponds to SPEED_I, feature_2 corresponds to SPEED_D, feature DISTANCE_I becomes feature_3 and feature DISTANCE_D becomes feature_4. Looking at the decision tree, it can be seen that feature_3 is the most important for the classification of the manoeuvre, because that has a larger effect on the model that is being used to predict the class. The next step was to analyse the correlation matrix shown in Figure 5.7.

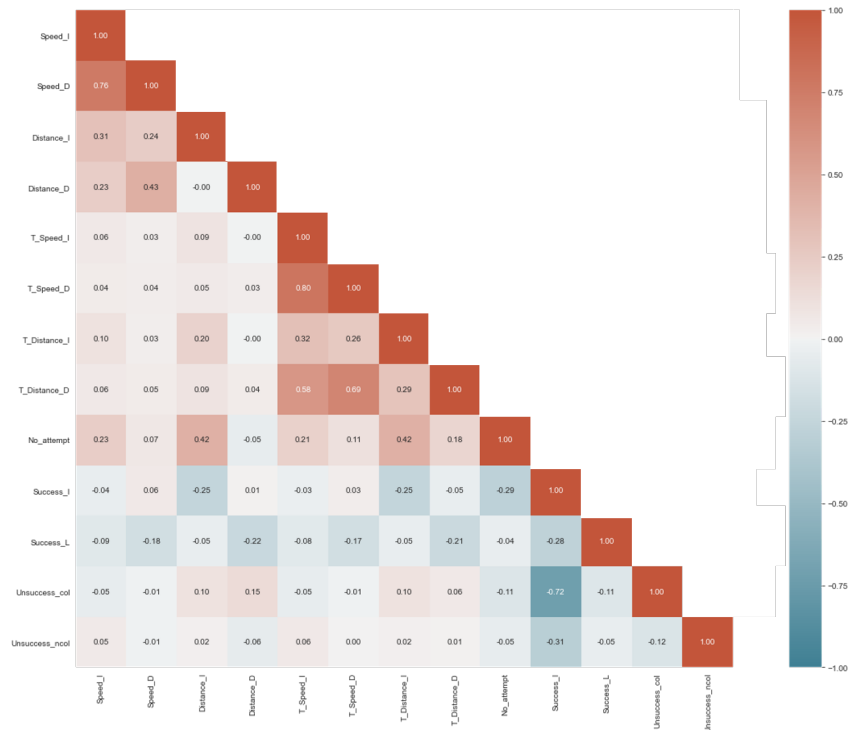


Figure 5.7: Correlation matrix computed from actions features.

One may see from the correlation matrix that no one feature can properly categorize an

overtake. With this, one may comprehend the decision tree's complexity and the sheer volume of its nodes. However, analysis of all the characteristics has shown that feature `DISTANCE_I`, feature_3, has the highest correlation with the classes, as proven earlier by analysing the decision tree.

In order to supplement the previous findings and have a clearer notion of the values that each feature takes, a table was created for each overtaking moment providing the minimum, average, and maximum values for each feature, grouped by simulation class. The table corresponding to the first moment of the overtaking manoeuvre is presented in Table 5.5.

Table 5.5: Statistics of the speed and distance measured up to the first moment of overtaking.

Feature CLASS	SPEED_I			Δ SPEED_I			SPEED_D			Δ SPEED_D		
	min	mean	max	min	mean	max	min	mean	max	min	mean	max
Success_L	0.00	7.55	71.00	0	3	29	0.00	9.24	67.00	0	3	28
Success_I	0.00	27.40	573.00	0	7	118	0.00	32.85	576.00	0	8	80
Unsuccess_ncol	0.00	39.59	448.00	0	10	66	0.00	41.41	444.00	0	10	65
Unsuccess_col	0.00	65.22	1073.00	0	15	177	0.00	69.49	1070.00	0	15	140
No_attempt	0.00	265.70	2635.00	0	48	397	0.00	250.77	2629.00	0	44	338

Feature CLASS	DISTANCE_I			Δ DISTANCE_I			DISTANCE_D			Δ DISTANCE_D		
	min	mean	max	min	mean	max	min	mean	max	min	mean	max
Success_L	0.00	0.00	0.00	0	0	0	0.00	7.62	73.62	0	24	155
Success_I	0.00	0.00	0.03	0	0	1	0.00	22.33	73.83	0	75	299
Unsuccess_ncol	0.00	0.12	3.27	0	4	108	0.00	18.67	48.42	0	115	702
Unsuccess_col	0.00	0.11	5.34	0	3	128	0.00	26.63	55.71	0	98	498
No_attempt	0.00	1.59	12.63	0	43	313	0.06	18.71	51.36	1	288	910

Table 5.5 allows for the visualization of some patterns that link the features to the classes. The most discrepant values are those corresponding to the `Success_L` class. One can conclude that for an overtaking to be successful and to respect traffic rules, speed should be varied few times and on a small scale. The same is true for distance. No simulation classified as `Success_L` registered an increase in the distance to the car in front. A curious fact extracted from the table is also the values corresponding to the `Unsuccess_col` class. For the feature `DISTANCE_D`, this registers the highest value of all classes, meaning that in these cases the vehicle decreases its distance to the car ahead too much, resulting in a collision. The high average values of this class for the `SPEED_I` and `SPEED_D` features confirm the previous premise that the high variance of ego vehicle speeds does not favour the success of the manoeuvre or even its existence, since the highest variance values correspond to the `No_attempt` class.

If these values are interpreted in the real world, a more aggressive driver is expected to have a high probability of collision with other drivers. This aggressiveness can translate into continuously changing speed values for various reasons. He may be driving at high speeds (constantly increasing it) and when he encounters an obstacle he has to brake, decreasing his speed value abruptly. Due to these abrupt adjustments, the distance between the vehicle and

the one it wants to overtake may also change significantly. According to the statistics, there is a strong probability that an accident will occur in these situations, supporting the previous hypothesis.

Table 5.6: Statistics of the speed and distance measured up to the second moment of overtaking.

Feature CLASS	SPEED_I			Δ SPEED_I			SPEED_D			Δ SPEED_D		
	min	mean	max	min	mean	max	min	mean	max	min	mean	max
Success_L	0.00	105.72	763.00	0	35.96	262	0.00	55.02	897.00	0	21.31	200
Success_I	0.00	65.54	753.00	0	22.43	275	0.00	104.54	751.00	0	22.04	285
Unsuccess_ncol	0.00	104.19	708.00	0	42.97	345	0.00	177.31	728.00	0	44.91	364
Unsuccess_col	0.00	5.68	276.00	0	9.03	74	0.00	40.35	357	0	96	114

Feature CLASS	DISTANCE_I			Δ DISTANCE_I			DISTANCE_D			Δ DISTANCE_D		
	min	mean	max	min	mean	max	min	mean	max	min	mean	max
Success_L	80.64	118.47	160.11	201	460.97	904	0.00	0.32	9.27	0	0.83	24
Success_I	41.61	78.36	170.52	7.54	260.44	961	0.00	0.07	8.46	0	0.21	24
Unsuccess_ncol	0.00	28.23	88.05	0	258.31	1003	0.00	17.25	83.40	0	221.60	1039
Unsuccess_col	1.44	12.47	68.64	5	48.21	312	0.00	0.51	14.49	0	8.28	205

Table 5.6 demonstrates that at the second moment of overtaking, the values are reversed from the first moment. Now, for successful overtaking, one must increase speed several times, and the distance to the overtaken car increases over time. In the unsuccessful overtaking without collisions, a high value of the feature DISTANCE_D is observed compared to the other classes. This could mean that the car encountered some obstacle in the lane it was moving to that made it brake while the overtaken car follows its path away from it. The low average values of the features corresponding to the Unsuccess_col class may mean that the driver have crashed shortly after beginning the second moment of the manoeuvre, preventing him from performing numerous actions.

With all of this knowledge, it is possible to find a sequence of actions that ensures a high probability of being successful. In the first moment, the ego vehicle must have smooth movements keeping its speed constant and gradually decrease the distance to the car it intends to overtake. Until conditions are favourable for changing lanes, this process is repeated. Once in the other lane, the car must increase its speed to reach a higher position than the other vehicle, leaving it behind.

5.2.3 Discussion of the Results

The data prediction phase consisted of two major steps: the study of the importance of the features obtained from the simulation and the study of the temporal sequence of actions that should be taken by the car to ensure safety and legality to the manoeuvre.

For the first stage, the Random Forest Classifier algorithm was applied to obtain the level of importance of each feature in the final classification of the overtaking manoeuvre. The model

performance was evaluated and it obtained better results after the data was balanced. Taking this into consideration, the features with a higher importance score were: D (distance between ego vehicle and overtaken vehicle), SE (current speed of ego), SP (current speed of overtaken vehicle), WT (waiting time to overtake) and DSEP (speed difference between ego vehicle and overtaken vehicle). This means that the values of these features have more influence on the final class than the others. These values were expected considering the general knowledge from overtaking scenarios that aggressive driving generates a higher probability of an unsuccessful overtake. In this case, aggressive driving may translate in a short distance left to the car ahead or a high speed of the ego vehicle. Although the performance of the prediction model used is significantly good, it is subject to an error rate that would make it impossible to implement in a real car at this stage. When implemented, problems of unpredictability of the actions of other drivers and factors that the model does not consider – such as the state of the road surface and objects other than cars, or that it simulated manually – such as the influence of weather conditions on the physics of the vehicle would have to be addressed. Again, the most reliable verification of these results would have to be done in the real world with the same features and the same parameters used in the simulation to increase the confidence level of the model prediction.

In the second stage, new features were defined, this time considering temporal actions by the ego vehicle in two moments of the overtaking: (1) before the ego vehicle changes lanes and (2) when it has already changed lanes and is attempting to get ahead of the second vehicle. A decision tree was first implemented to obtain the relationship between the features and the final class of overtaking. As a result, it was obtained that the feature DISTANCE_I (total increase of the distance to the car in front) is the most important feature for the classification of the manoeuvre. This finding confirmed that, even considering other moments of overtaking, the distance left to the car in front is a decisive factor for the success of the manoeuvre. However, from the interpretation of the correlation matrix applied to the new features, it was concluded that no feature can, alone, categorize the overtaking manoeuvre. That is, it is not enough just to increase the distance to the car ahead for the manoeuvre to be successful. In order to study the sequence of actions and find patterns and discrepancies in the values, statistical methods were applied, calculating the minimum, average, and maximum values of each feature associated to each class. The values were obtained for the two moments of overtaking. In the first moment it is concluded that for an overtaking to be considered successful and respect the traffic rules, the speed and distance should be varied few times and in small scale. At the second moment, the values are reversed and for an overtaking to be successful it is necessary to increase the speed several times and increase the distance gradually over time. The sequence of actions presented in this work does not guarantee the success of the overtaking, there is always an associated error rate. However, it is a starting point for future studies and constitutes a clearer notion of the precautionary moves that must be taken by the autonomous car during such a complex and, sometimes, dangerous manoeuvre as overtaking.

Chapter 6

Conclusions

Autonomous driving has become a growing interest in the scientific community. The automotive industry is heading towards a future where the driver will play a less active role in driving his vehicle, even ending up being completely excluded. The challenges of autonomous driving resolve around accurately perceiving the environment surrounding the autonomous car, understanding and distinguishing the various elements that constitute the scene. In addition to perception systems, cars require mechanisms that support decision-making that result in strategies that include behaviours and control commands. Taking all this into consideration, this study focuses on the decision-making process for overtaking scenarios by representing knowledge and encoding traffic rules to make the manoeuvre safe and legal. This work intends to add to the lane change warning systems that already exist in the market a system that not only considers vehicle speeds and location but also traffic rule infraction and behaviour prediction. The main contribution is a vast literature review gathering the main studies in the area of autonomous driving, a synthetic dataset for overtaking manoeuvres and the collection of several important factors to be considered for decision-making during overtaking.

A synthetic dataset was generated considering the important factors for the classification of the overtaking manoeuvre. The knowledge acquired during the simulations was represented, and traffic rules were codified into logical rules in order to guarantee safety and legality to the manoeuvre. New features were created, and their importance was analysed considering a class prediction model. Subsequently, the dataset was compared by means of works found in the literature that use real overtaking data. Finally, data of moments before the overtaking were considered in order to obtain a sequence of actions that guarantees the success of the overtaking. Therefore, the objectives considered for this work were achieved.

One of the biggest beliefs in this project is that synthetic data is a good starting point to predict events and to build a strong knowledge base to generate a decision that offers safety and effectiveness to the overtaking manoeuvre. The performance of the model evaluated in this work was justified with the data balancing that increased the prediction effectiveness. The rules applied guaranteed legality and safety to the system that is intended to be implemented in the

future, taking into account important factors such as the distance to the car to be overtaken, the speed and speed difference of the intervening cars and the waiting time to start the overtaking manoeuvre. Another important finding of this work was the discrediting of the two seconds rule as a safety measure while driving, and it should be replaced by the 3 or even 4 seconds rule.

During the development of this work some limitations were encountered, such as: the lack of some important functionalities in the simulator making it impossible/hard to collect essential information; the lack of datasets in the literature in the scope of overtaking which also made it difficult to validate the artificially obtained data; and the difficulty in collecting real data due to the danger and unpredictability present in the overtaking manoeuvre. These limitations were, to some extent, overcome and can be effectively improved in a future work.

6.1 Future Work

In terms of future work, a more extensive study should be carried out on other forms of knowledge representation and other methods that support data generation. In order to increase the amount and diversity of simulated data, data augmentation techniques or neural networks such as deep generative models should be explored.

A more complex task but one that will enrich this work is the use of a forecasting technique to predict future events with the simulated data as input.

In the case of using the simulator to obtain the data, the number of features used should be increased and a study on the minimum number of frames that maintains the predictive performance of the models should be carried out.

On the other hand, the implementation of the generation of automatic (non-manual) rules for the codification of traffic rules would enrich the work. In order to introduce complexity to the work, it will also be interesting to add some object classes like pedestrians and animals.

The next step of this work will be: first, from the predictive models, collect real data and compare to the synthetic dataset and then, implement a system in a real vehicle to improve the effectiveness and safety of autonomous cars. The idea here is that as soon as the driver demonstrates his will to overtake a vehicle, it will be possible to tell him the percentage of success that the manoeuvre will have so that the driver can decide whether to perform it or not. The success percentage of the manoeuvre is obtained from the model is the one associated with the `Success_L` class when an array with the values of each feature measured at that moment is provided. Other information such as collision danger or the speed at which the vehicle in front is travelling could also be provided to the driver. For this, it is expected that tests will be made with a real car in order to validate as realistically as possible the data obtained in the simulations. The algorithms must be used below the perceptual layer, and tests must be conducted on closed tracks. Due to the inability to acquire certain data, such as vertical traffic signs, the data collected during these testing differs significantly from the information gathered in a real scenario. As a result, a

system using the algorithm described in this work that is implemented in a real car will need to approximate and adjust the simulated data to reality, as well as adapting the implemented rules to account for the enforced constraints. All the decision-making information can be shown in a thoughtful prototype of the dashboard of such a system implemented in the vehicle that is illustrated in Figure 6.1.



Figure 6.1: Prototype of the dashboard with predictive model implemented.
Image adapted from [75].

Bibliography

- [1] [AAA: Today's vehicle technology must walk so self-driving cars can run](#). Feb. 2021 (cit. on p. 24).
- [2] acp. [O que mudou no Código da Estrada | ACP](#). 2022 (cit. on p. 49).
- [3] Somak Aditya, Yezhou Yang and Chitta Baral. 'Integrating Knowledge and Reasoning in Image Understanding'. In: (2019). DOI: [10.48550/ARXIV.1906.09954](#) (cit. on pp. 11, 40).
- [4] National Highway Traffic Safety Administration. [Preliminary Statement of Policy Concerning Automated Vehicles](#). 2015 (cit. on p. 1).
- [5] Andrei Aksjonov and Ville Kyrki. 'Rule-Based Decision-Making System for Autonomous Vehicles at Intersections with Mixed Traffic Environment'. In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 660–666. DOI: [10.1109/ITSC48978.2021.9565085](#) (cit. on p. 14).
- [6] Hassan Abu Alhaija et al. *Augmented Reality Meets Computer Vision : Efficient Data Generation for Urban Driving Scenes*. 2017. arXiv: [1708.01566 \[cs.CV\]](#) (cit. on p. 30).
- [7] 'Amazon invests in Rivian, a Tesla rival in electric vehicles'. In: *The New York Times* (2021) (cit. on p. 22).
- [8] [An introduction to seaborn](#). 2012 (cit. on p. 59).
- [9] Saurabh Arora and Prashant Doshi. 'A survey of inverse reinforcement learning: Challenges, methods and progress'. In: *Artificial Intelligence* 297 (2021), p. 103500. ISSN: 0004-3702. DOI: <https://doi.org/10.1016/j.artint.2021.103500> (cit. on p. 24).
- [10] [Artigo 146º: Contraordenações muito graves | Código da Estrada 2022](#). 2022 (cit. on pp. 49, 56).
- [11] [Artigo 18º: Distância entre veículos | Código da Estrada 2022](#). 2022 (cit. on p. 56).
- [12] [Artigo 27º: Limites Gerais de velocidade | Código da Estrada 2022](#). 2022 (cit. on pp. 47, 56).
- [13] [Artigo 38º: Realização da manobra | Código da Estrada 2022](#). 2022 (cit. on pp. 52, 56).
- [14] [Artigo 41º: Ultrapassagens Proibidas | Código da Estrada 2022](#). 2022 (cit. on p. 56).

- [15] Antonio Artuñedo, Jorge Godoy and Jorge Villagra. ‘[A decision-making architecture for automated driving without detailed prior maps](#)’. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1645–1652. DOI: [10.1109/IVS.2019.8814070](#) (cit. on p. 14).
- [16] Gowri Asaithambi and Gugulothu Shravani. ‘[Overtaking behaviour of vehicles on undivided roads in non-lane based mixed traffic conditions](#)’. In: *Journal of Traffic and Transportation Engineering (English Edition)* 4.3 (2017), pp. 252–261. ISSN: 2095-7564. DOI: [https://doi.org/10.1016/j.jtte.2017.05.004](#) (cit. on pp. 36, 86–87).
- [17] *Authoring Maps - Carla Simulator*. 2022 (cit. on p. 33).
- [18] *Autopilot*. [Online; accessed 19. Nov. 2021]. Nov. 2021 (cit. on p. 31).
- [19] *Autopilot and Full Self-driving Capability*. Feb. 2019 (cit. on p. 23).
- [20] Simon Baker et al. ‘[A Database and Evaluation Methodology for Optical Flow](#)’. In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8. DOI: [10.1109/ICCV.2007.4408903](#) (cit. on p. 29).
- [21] Mayank Bansal, Alex Krizhevsky and Abhijit Ogale. ‘[Chauffeurnet: learning to drive by imitating the best and synthesizing the worst](#)’. In: *Robotics: Science and Systems XV*. Robotics: Science and Systems Foundation, June 2019. ISBN: 978-0-9923747-5-4. DOI: [10.15607/RSS.2019.XV.031](#) (cit. on p. 24).
- [22] Jining Bao et al. ‘[Unpaved road detection based on spatial fuzzy clustering algorithm](#)’. In: *EURASIP Journal on Image and Video Processing* 2018 (Apr. 2018). DOI: [10.1186/s13640-018-0260-3](#) (cit. on p. 34).
- [23] Peter Barnum et al. ‘[Dynamic Seethroughs: Synthesizing Hidden Views of Moving Objects](#)’. In: Oct. 2009, pp. 111–114. DOI: [10.1109/ISMAR.2009.5336483](#) (cit. on p. 26).
- [24] Keith Barry. *Do Levels of Car Automation Matter?* Oct. 2019 (cit. on p. 2).
- [25] Chandrayee Basu et al. ‘[Do you want your autonomous car to drive like you?](#)’ In: *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction - HRI '17*. ACM Press, Mar. 2017, pp. 417–425. ISBN: 9781450343367. DOI: [10.1145/2909824.3020250](#) (cit. on p. 24).
- [26] Carolina Bento. *Decision Tree Classifier explained*. June 2021 (cit. on p. 96).
- [27] Michele Bertonecello and Dominik Wee. ‘[Ten ways autonomous driving could redefine the automotive world](#)’. In: *McKinsey & Company* (June 2021) (cit. on p. 22).
- [28] Henrik Bey et al. ‘[Optimization-based Tactical Behavior Planning for Autonomous Freeway Driving in Favor of the Traffic Flow](#)’. In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1033–1040. DOI: [10.1109/IVS.2019.8813787](#) (cit. on p. 14).
- [29] José-Luis Blanco-Claraco, Francisco-Ángel Moreno-Dueñas and Javier González-Jiménez. ‘[The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario](#)’. In: *Int. J. Rob. Res.* 33.2 (Oct. 2013), pp. 207–214. ISSN: 0278-3649. DOI: [10.1177/0278364913507326](#) (cit. on p. 30).
- [30] BMW. *Overview of the main driver assistance systems | BMW*. 2021 (cit. on p. 23).

- [31] *BMW group and Baidu join forces to accelerate the development of autonomous driving in China*. 2018 (cit. on p. 22).
- [32] *Bosch - Driver Drowsiness Detection*. 2022 (cit. on p. 35).
- [33] *Bosch: The levels of automated driving*. 2021 (cit. on pp. 2, 23).
- [34] Leo Breiman et al. *Classification and regression trees*. Routledge, 2017 (cit. on p. 92).
- [35] C. Busch and E. Debes. ‘Wavelet transform for analyzing fog visibility’. In: *IEEE Intelligent Systems and their Applications* 13.6 (1998), pp. 66–71. DOI: [10.1109/5254.736004](https://doi.org/10.1109/5254.736004) (cit. on p. 26).
- [36] William N. Caballero, David Ríos Insua and David Banks. ‘Decision support issues in automated driving systems’. In: *International Transactions in Operational Research* n/a.n/a (). DOI: <https://doi.org/10.1111/itor.12936>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.12936> (cit. on p. 2).
- [37] Holger Caesar et al. *nuScenes: A multimodal dataset for autonomous driving*. 2020. arXiv: [1903.11027](https://arxiv.org/abs/1903.11027) [cs.LG] (cit. on pp. 27–30).
- [38] *carla: carla::rpc::weatherparameters class reference*₂₀₂₂. 2022 (cit. on p. 47).
- [39] *Carla: Weather parameters*. 2022 (cit. on p. 46).
- [40] *CarSim Overview*. 2020 (cit. on p. 32).
- [41] Sara Castellanos. *Fake It to Make It: Companies Beef Up AI Models With Synthetic Data*. July 2021 (cit. on p. 28).
- [42] Hazar Chaabani et al. ‘A Neural network approach to visibility range estimation under foggy weather conditions’. In: *Procedia Computer Science* 113 (2017). The 8th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2017) / The 7th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2017) / Affiliated Workshops, pp. 466–471. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2017.08.304> (cit. on p. 26).
- [43] Chun-Wei Chang et al. ‘Multi-point turn decision making framework for human-like automated driving’. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–6. DOI: [10.1109/ITSC.2017.8317831](https://doi.org/10.1109/ITSC.2017.8317831) (cit. on p. 14).
- [44] N. V. Chawla et al. ‘SMOTE: Synthetic Minority Over-sampling Technique’. In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. DOI: [10.1613/jair.953](https://doi.org/10.1613/jair.953) (cit. on p. 94).
- [45] Xiaojun Chen, Shengbin Jia and Yang Xiang. ‘A review: Knowledge reasoning over knowledge graph’. In: *Expert Systems with Applications* 141 (2020), p. 112948. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2019.112948> (cit. on p. 41).
- [46] Yen-Lin Chen et al. ‘Nighttime Vehicle Detection for Driver Assistance and Autonomous Vehicles’. In: *18th International Conference on Pattern Recognition (ICPR’06)*. Vol. 1. 2006, pp. 687–690. DOI: [10.1109/ICPR.2006.858](https://doi.org/10.1109/ICPR.2006.858) (cit. on p. 34).

- [47] Yiping Chen et al. ‘LiDAR-Video Driving Dataset: Learning Driving Policies Effectively’. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5870–5878. DOI: [10.1109/CVPR.2018.00615](https://doi.org/10.1109/CVPR.2018.00615) (cit. on p. 30).
- [48] In Collaboration. *The Big Deal with Big Data Isn’t (Just) the Data*. Aug. 2016 (cit. on p. 8).
- [49] Oscar Corcho and Asuncion Gomez-Perez. ‘Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages’. In: (Aug. 2001) (cit. on p. 40).
- [50] *cross-validation: evaluating estimator performance*. 2013 (cit. on p. 94).
- [51] Ben Dickson. *Neuro-symbolic AI brings us closer to machines with common sense*. Mar. 2022 (cit. on p. 50).
- [52] Shilp Dixit et al. ‘Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects’. In: *Annual Reviews in Control* 45 (2018), pp. 76–86. ISSN: 1367-5788. DOI: <https://doi.org/10.1016/j.arcontrol.2018.02.001> (cit. on p. 35).
- [53] Yanchao Dong et al. ‘Driver Inattention Monitoring System for Intelligent Vehicles: A Review’. In: *IEEE Transactions on Intelligent Transportation Systems* 12.2 (2011), pp. 596–614. DOI: [10.1109/TITS.2010.2092770](https://doi.org/10.1109/TITS.2010.2092770) (cit. on p. 35).
- [54] Alexey Dosovitskiy et al. *CARLA: An Open Urban Driving Simulator*. 2017. DOI: [10.48550/ARXIV.1711.03938](https://doi.org/10.48550/ARXIV.1711.03938) (cit. on p. 45).
- [55] *Dython*. Mar. 2022 (cit. on pp. 59, 66).
- [56] Escola. *Escola de Condução Invicta - Tempo de Reacção*. 2022 (cit. on p. 57).
- [57] James A. Evans and Jacob G. Foster. ‘Metaknowledge’. In: *Science* 331.6018 (Feb. 2011), pp. 721–725. DOI: [10.1126/science.1201765](https://doi.org/10.1126/science.1201765) (cit. on p. 39).
- [58] Jin Fang et al. ‘Augmented LiDAR Simulator for Autonomous Driving’. In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 1931–1938. ISSN: 2377-3774. DOI: [10.1109/lra.2020.2969927](https://doi.org/10.1109/lra.2020.2969927) (cit. on p. 30).
- [59] *Feature Engineering Step by Step*. Mar. 2021 (cit. on p. 16).
- [60] Miguel C. Figueiredo et al. ‘An approach to simulate autonomous vehicles in urban traffic scenarios’. In: *2009 12th International IEEE Conference on Intelligent Transportation Systems*. 2009, pp. 1–6. DOI: [10.1109/ITSC.2009.5309524](https://doi.org/10.1109/ITSC.2009.5309524) (cit. on p. 31).
- [61] Colin Fischer, Monika Sester and Steffen Schön. ‘Spatio-Temporal Research Data Infrastructure in the Context of Autonomous Driving’. In: *ISPRS International Journal of Geo-Information* 9.11 (2020). ISSN: 2220-9964 (cit. on p. 33).
- [62] Karl Friston et al. ‘World model learning and inference’. In: *Neural Networks* 144 (2021), pp. 573–590. ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2021.09.011> (cit. on p. 50).
- [63] Paul Gao et al. *Automotive revolution – perspective towards 2030*. 2016 (cit. on p. 22).

- [64] Alberto Garcia-Duran and Mathias Niepert. *KBLRN : End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features*. 2017. DOI: [10.48550/ARXIV.1709.04676](https://doi.org/10.48550/ARXIV.1709.04676) (cit. on p. 39).
- [65] *GM Cruise AVs Drive 200,000 Miles Daily In 3D Simulators: Video*. [Online; accessed 19. Nov. 2021]. May 2020 (cit. on p. 30).
- [66] Margherita Grandini, Enrico Bagli and Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. DOI: [10.48550/ARXIV.2008.05756](https://doi.org/10.48550/ARXIV.2008.05756) (cit. on p. 18).
- [67] Stephan Grimm, Pascal Hitzler and Andreas Abecker. ‘Knowledge Representation and Ontologies Logic, Ontologies and Semantic Web Languages’. In: 2007 (cit. on pp. 11–13).
- [68] Frank E. Grubbs. ‘Procedures for Detecting Outlying Observations in Samples’. In: *Technometrics* 11.1 (1969), pp. 1–21. DOI: [10.1080/00401706.1969.10490657](https://doi.org/10.1080/00401706.1969.10490657) (cit. on p. 17).
- [69] Dominique Gruyer et al. ‘Perception, information processing and modeling: Critical stages for autonomous driving applications’. In: *Annual Reviews in Control* 44 (2017), pp. 323–341. ISSN: 1367-5788. DOI: <https://doi.org/10.1016/j.arcontrol.2017.09.012> (cit. on pp. 6–7, 28).
- [70] Junyao Guo, Unmesh Kurup and Mohak Shah. ‘Is it Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving’. In: *IEEE Transactions on Intelligent Transportation Systems* 21.8 (2020), pp. 3135–3151. DOI: [10.1109/TITS.2019.2926042](https://doi.org/10.1109/TITS.2019.2926042) (cit. on p. 33).
- [71] *Honda Global March 4, 2021 Honda to Begin Sales of Legend with New Honda SENSING Elite*. Nov. 2021 (cit. on p. 2).
- [72] WuLing Huang et al. ‘Autonomous vehicles testing methods review’. In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Nov. 2016, pp. 163–168. DOI: [10.1109/ITSC.2016.7795548](https://doi.org/10.1109/ITSC.2016.7795548) (cit. on p. 28).
- [73] Dimia Iberraken, Lounis Adouane and Dieumet Denis. ‘Safe Autonomous Overtaking Maneuver based on Inter-Vehicular Distance Prediction and Multi-Level Bayesian Decision-Making’. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3259–3265. DOI: [10.1109/ITSC.2018.8569401](https://doi.org/10.1109/ITSC.2018.8569401) (cit. on p. 14).
- [74] Aurora Innovation Inc. *The Offline Executor: Virtual Testing and the Aurora Driver*. [Online; accessed 19. Nov. 2021]. Nov. 2021 (cit. on p. 31).
- [75] *Infotainment and cockpit solutions*. 2022 (cit. on p. 109).
- [76] *Introduction to AI*. 2022 (cit. on pp. 11–13).
- [77] H. Julia, E.B. Kosmatopoulos and P.A. Ioannou. ‘Collision avoidance analysis for lane changing and merging’. In: *IEEE Transactions on Vehicular Technology* 49.6 (2000), pp. 2295–2308. DOI: [10.1109/25.901899](https://doi.org/10.1109/25.901899) (cit. on p. 36).
- [78] Hojung Jung et al. ‘Multi-modal panoramic 3D outdoor datasets for place categorization’. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 4545–4550. DOI: [10.1109/IROS.2016.7759669](https://doi.org/10.1109/IROS.2016.7759669) (cit. on p. 30).

- [79] Nidhi Kalra and Susan Paddock. ‘Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?’ In: *Transportation Research Part A: Policy and Practice* 94 (Dec. 2016), pp. 182–193. DOI: [10.1016/j.tra.2016.09.010](https://doi.org/10.1016/j.tra.2016.09.010) (cit. on p. 28).
- [80] Hang-Bong Kang. ‘Various Approaches for Driver and Driving Behavior Monitoring: A Review’. In: *2013 IEEE International Conference on Computer Vision Workshops*. 2013, pp. 616–623. DOI: [10.1109/ICCVW.2013.85](https://doi.org/10.1109/ICCVW.2013.85) (cit. on p. 35).
- [81] Sinan Kaplan et al. ‘Driver Behavior Analysis for Safe Driving: A Survey’. In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 3017–3032. DOI: [10.1109/TITS.2015.2462084](https://doi.org/10.1109/TITS.2015.2462084) (cit. on p. 35).
- [82] O. Karaduman et al. ‘Interactive risky behavior model for 3-car overtaking scenario using joint Bayesian network’. In: *2013 IEEE Intelligent Vehicles Symposium (IV)*. 2013, pp. 1279–1284. DOI: [10.1109/IVS.2013.6629642](https://doi.org/10.1109/IVS.2013.6629642) (cit. on p. 36).
- [83] Jesper Karlsson. ‘Enhancing Object Recognition Based on Contextual Information Using Markov Logic Networks’. In: (2016) (cit. on p. 40).
- [84] Prabhjot Kaur et al. *A Survey on Simulators for Testing Self-Driving Cars*. 2021. DOI: [10.48550/ARXIV.2101.05337](https://doi.org/10.48550/ARXIV.2101.05337) (cit. on pp. 31–32).
- [85] *Key to success: the BMW Motorsport simulator*. [Online; accessed 19. Nov. 2021]. Nov. 2021 (cit. on p. 31).
- [86] Charles-Éric Noël Laflamme, François Pomerleau and Philippe Giguère. ‘Driving Datasets Literature Review’. In: *arXiv* (Oct. 2019) (cit. on p. 29).
- [87] Nicholas C. Landolfi and Anca D. Dragan. ‘Social Cohesion in Autonomous Driving’. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 8118–8125. DOI: [10.1109/IROS.2018.8593682](https://doi.org/10.1109/IROS.2018.8593682) (cit. on p. 24).
- [88] John Leonard et al. ‘A Perception-Driven Autonomous Urban Vehicle’. In: *Journal of Field Robotics* 25 (Oct. 2008), pp. 727–774. DOI: [10.1002/rob.20262](https://doi.org/10.1002/rob.20262) (cit. on p. 14).
- [89] Christophe Leys et al. ‘Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median’. In: *Journal of Experimental Social Psychology* 49.4 (2013), pp. 764–766. ISSN: 0022-1031. DOI: <https://doi.org/10.1016/j.jesp.2013.03.013> (cit. on p. 65).
- [90] Nan Li et al. *Game-Theoretic Modeling of Driver and Vehicle Interactions for Verification and Validation of Autonomous Vehicle Control Systems*. 2016. DOI: [10.48550/ARXIV.1608.08589](https://doi.org/10.48550/ARXIV.1608.08589) (cit. on p. 14).
- [91] Qingquan Li et al. ‘A Sensor-Fusion Drivable-Region and Lane-Detection System for Autonomous Vehicle Navigation in Challenging Road Scenarios’. In: *IEEE Transactions on Vehicular Technology* 63.2 (2014), pp. 540–555. DOI: [10.1109/TVT.2013.2281199](https://doi.org/10.1109/TVT.2013.2281199) (cit. on p. 34).

- [92] Liangkai Liu et al. ‘[Computing Systems for Autonomous Driving: State of the Art and Challenges](#)’. In: *IEEE Internet of Things Journal* 8.8 (2020), pp. 6469–6486 (cit. on pp. 24, 28).
- [93] Qi Liu et al. *Decision-Making Technology for Autonomous Vehicles Learning-Based Methods, Applications and Future Outlook*. 2021. DOI: [10.48550/ARXIV.2107.01110](#) (cit. on pp. 16, 25).
- [94] Shaoshan Liu et al. ‘[Edge Computing for Autonomous Driving: Opportunities and Challenges](#)’. In: *Proceedings of the IEEE* 107.8 (2019), pp. 1697–1716. DOI: [10.1109/JPROC.2019.2915983](#) (cit. on p. 25).
- [95] Carlos Llorca and Haneen Farah. ‘[Passing Behavior on Two-Lane Roads in Real and Simulated Environments](#)’. In: *Transportation Research Record* 2556.1 (2016), pp. 29–38. DOI: [10.3141/2556-04](#) (cit. on p. 37).
- [96] Carlos Llorca and Alfredo García. ‘[Evaluation of Passing Process on Two-Lane Rural Highways in Spain with New Methodology Based on Video Data](#)’. In: *Transportation Research Record* 2262.1 (2011), pp. 42–51. DOI: [10.3141/2262-05](#) (cit. on p. 36).
- [97] S. M. Mahmud. ‘[Modelling driving behaviour and safety risk in two-lane rural highways in developing countries](#)’. PhD thesis. Sept. 2019. DOI: [10.14264/uql.2019.821](#) (cit. on pp. 87–88).
- [98] S. M. Mahmud, Ishtiaque Ahmed and Md Hoque. ‘[Road Safety Problems in Bangladesh: Achievable Target and Tangible Sustainable Actions](#)’. In: *Jurnal Teknologi* 70 (Sept. 2014). DOI: [10.11113/jt.v70.3487](#) (cit. on p. 22).
- [99] S.M. Sohel Mahmud et al. ‘[Overtaking risk modeling in two-lane two-way highway with heterogeneous traffic environment of a low-income country using naturalistic driving dataset](#)’. In: *Journal of Safety Research* 80 (2022), pp. 380–390. ISSN: 0022-4375. DOI: [https://doi.org/10.1016/j.jsr.2021.12.019](#) (cit. on p. 86).
- [100] *Manual do Ensino da Condução - IMT*. 2010 (cit. on p. 55).
- [101] Bernard Marr. ‘[What Is Unstructured Data And Why Is It So Important To Businesses? An Easy Explanation For Anyone](#)’. In: *Forbes* (Dec. 2021) (cit. on p. 9).
- [102] *MATLAB: Automated Driving Toolbox*. 2022 (cit. on p. 32).
- [103] *matplotlib — visualization with python*₂₀₂₂. 2022 (cit. on pp. 59, 65).
- [104] Nikolaus Mayer et al. ‘[What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?](#)’ In: *International Journal of Computer Vision* 126.9 (Apr. 2018), pp. 942–960. ISSN: 1573-1405. DOI: [10.1007/s11263-018-1082-6](#) (cit. on p. 29).
- [105] *Mercedes-benz: valet guys*. 2021 (cit. on p. 24).
- [106] *Metrics and scoring: quantifying the quality of predictions*. 2013 (cit. on p. 94).
- [107] Ben Miethig et al. ‘[Leveraging Thermal Imaging for Autonomous Driving](#)’. In: *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*. 2019, pp. 1–5. DOI: [10.1109/ITEC.2019.8790493](#) (cit. on p. 25).

- [108] Michael Montemerlo et al. ‘Junior: The Stanford Entry in the Urban Challenge.’ In: Jan. 2009, pp. 91–123 (cit. on p. 14).
- [109] Smart Motorist. *What is a Safe Following Distance? (3 Second Rule)*. Jan. 2019 (cit. on p. 84).
- [110] Lukáš Neumann et al. ‘NightOwls: A Pedestrians at Night Dataset’. In: May 2019, pp. 691–705. ISBN: 978-3-030-20886-8. DOI: [10.1007/978-3-030-20887-5_43](https://doi.org/10.1007/978-3-030-20887-5_43) (cit. on p. 34).
- [111] Holly New. *Chapter 8: Defensive Driving*. Nov. 2013 (cit. on p. 57).
- [112] NVIDIA Newsroom. *NVIDIA, AUDI Partner to Put World’s Most Advanced AI Car on Road by 2020*. 2020 (cit. on p. 22).
- [113] Farzan Erlik Nowruzi et al. *How much real data do we actually need: Analyzing object detection performance using synthetic and real data*. 2019. arXiv: [1907.07061](https://arxiv.org/abs/1907.07061) [cs.CV] (cit. on p. 29).
- [114] *numpy - The fundamental package for scientific computing with Python*. 2022 (cit. on p. 59).
- [115] Maxwell Nye et al. *Improving Coherence and Consistency in Neural Sequence Models with Dual-System, Neuro-Symbolic Reasoning*. 2021. DOI: [10.48550/ARXIV.2107.02794](https://doi.org/10.48550/ARXIV.2107.02794) (cit. on p. 50).
- [116] Eshed Ohn-Bar and Mohan Manubhai Trivedi. ‘Looking at Humans in the Age of Self-Driving and Highly Automated Vehicles’. In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 90–104. DOI: [10.1109/TIV.2016.2571067](https://doi.org/10.1109/TIV.2016.2571067) (cit. on p. 35).
- [117] Cristina Olaverri Monreal et al. ‘The See-Through System: A VANET-enabled assistant for overtaking maneuvers’. In: vol. 123-128. July 2010, pp. 123–128. DOI: [10.1109/IVS.2010.5548020](https://doi.org/10.1109/IVS.2010.5548020) (cit. on pp. 26, 34).
- [118] *only 3% of companies’ data meets basic quality standards*₂₀₁₇. Sept. 2017 (cit. on p. 85).
- [119] Juri Opitz and Sebastian Burst. *Macro F1 and Macro F1*. 2019. DOI: [10.48550/ARXIV.1911.03347](https://doi.org/10.48550/ARXIV.1911.03347) (cit. on p. 18).
- [120] OSRF. *Gazebo : Tutorial : ROS overview*. 2014 (cit. on p. 32).
- [121] *p-value: what it is, how to calculate it, and why it matters*. 2022 (cit. on p. 67).
- [122] *pandas - python data analysis library*₂₀₂₂. 2022 (cit. on p. 59).
- [123] Pedro Patron et al. ‘Semantic knowledge-based representation for improving situation awareness in service oriented agents of autonomous underwater vehicles’. In: Oct. 2008, pp. 1–9. DOI: [10.1109/OCEANS.2008.5151932](https://doi.org/10.1109/OCEANS.2008.5151932) (cit. on p. 38).
- [124] Joshué Pérez et al. ‘Longitudinal fuzzy control for autonomous overtaking’. In: *2011 IEEE International Conference on Mechatronics*. 2011, pp. 188–193. DOI: [10.1109/ICMECH.2011.5971279](https://doi.org/10.1109/ICMECH.2011.5971279) (cit. on p. 36).

- [125] P. Shunmuga Perumal et al. ‘An insight into crash avoidance and overtaking advice systems for Autonomous Vehicles: A review, challenges and solutions’. In: *Engineering Applications of Artificial Intelligence* 104 (2021), p. 104406. ISSN: 0952-1976. DOI: <https://doi.org/10.1016/j.engappai.2021.104406> (cit. on p. 36).
- [126] Peter Pinggera et al. ‘Lost and Found: detecting small road hazards for self-driving vehicles’. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 1099–1106. DOI: [10.1109/IROS.2016.7759186](https://doi.org/10.1109/IROS.2016.7759186) (cit. on p. 34).
- [127] *PreScan: Simulation of ADAS and active safety*. 2022 (cit. on p. 32).
- [128] Gil Press. ‘6 Predictions About Data In 2020 And The Coming Decade’. In: *Forbes* (Jan. 2020) (cit. on p. 8).
- [129] Amir Rasouli and John K. Tsotsos. ‘Autonomous Vehicles That Interact With Pedestrians: A Survey of Theory and Practice’. In: *IEEE Transactions on Intelligent Transportation Systems* 21.3 (2020), pp. 900–918. DOI: [10.1109/TITS.2019.2901817](https://doi.org/10.1109/TITS.2019.2901817) (cit. on p. 35).
- [130] Charles Reinholtz et al. ‘Team VictorTango’s entry in the DARPA Urban Challenge’. In: vol. 25. Jan. 2009, pp. 125–162. DOI: [10.1002/rob.20248](https://doi.org/10.1002/rob.20248) (cit. on p. 14).
- [131] Gene Roe. *The true cost - implementing lidar into your business - LiDAR News*. Apr. 2022 (cit. on p. 27).
- [132] Al Root. *Lidar Needs to Get Cheaper and Better. Why This Stock Could Benefit*. July 2021 (cit. on p. 27).
- [133] Stuart Russell and Peter Norvig. ‘Artificial Intelligence: a Modern Approach’. In: Pearson, 2022. ISBN: 978-0134610993 (cit. on pp. 10, 13).
- [134] Mathias Sablé-Meyer et al. *A language of thought for the mental representation of geometric shapes*. Dec. 2021. DOI: [10.31234/osf.io/28mg4](https://doi.org/10.31234/osf.io/28mg4) (cit. on p. 50).
- [135] Dorsa Sadigh et al. ‘Planning for Autonomous Cars that Leverage Effects on Human Actions’. In: June 2016. DOI: [10.15607/RSS.2016.XII.029](https://doi.org/10.15607/RSS.2016.XII.029) (cit. on pp. 24, 35).
- [136] Robin Schubert, Karsten Schulze and Gerd Wanielik. ‘Situation Assessment for Automatic Lane-Change Maneuvers’. In: *IEEE Transactions on Intelligent Transportation Systems* 11.3 (2010), pp. 607–616. DOI: [10.1109/TITS.2010.2049353](https://doi.org/10.1109/TITS.2010.2049353) (cit. on p. 35).
- [137] *Scikit-learn: machine learning in python – documentation*. 2022 (cit. on p. 59).
- [138] *scipy - Fundamental algorithms for scientific computing in Python*. 2022 (cit. on p. 59).
- [139] *seaborn: statistical data visualization — seaborn 0.11.2 documentation*₂₀₁₂. 2012 (cit. on p. 65).
- [140] *seaborn.swarmplot — seaborn documentation*. 2012 (cit. on p. 65).
- [141] *Semana Mundial da Segurança Rodoviária das Nações Unidas 2021*. 2021 (cit. on p. 22).
- [142] Young-Woo Seo et al. ‘Recognition of Highway Workzones for Reliable Autonomous Driving’. In: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2015), pp. 708–718. DOI: [10.1109/TITS.2014.2335535](https://doi.org/10.1109/TITS.2014.2335535) (cit. on p. 34).

- [143] T. Shamir. ‘[How should an autonomous vehicle overtake a slower moving vehicle: design and analysis of an optimal trajectory](#)’. In: *IEEE Transactions on Automatic Control* 49.4 (2004), pp. 607–610. DOI: [10.1109/TAC.2004.825632](#) (cit. on p. 36).
- [144] German Sharabok. [Why Tesla Won’t Use LIDAR - Towards Data Science](#). Sept. 2020 (cit. on p. 27).
- [145] Mohammad Shokrolah Shirazi and Brendan Tran Morris. ‘[Looking at Intersections: A Survey of Intersection Monitoring, Behavior and Safety Analysis of Recent Studies](#)’. In: *IEEE Transactions on Intelligent Transportation Systems* 18.1 (2017), pp. 4–24. DOI: [10.1109/TITS.2016.2568920](#) (cit. on p. 34).
- [146] [sklearn.ensemble.randomforestclassifier](#). 2022 (cit. on p. 92).
- [147] [sklearn.metrics.balanced_accuracy_score](#). 2015 (cit. on p. 94).
- [148] Bosch Mobility Solutions. [Interior Monitoring Systems](#). 2021 (cit. on p. 2).
- [149] Bosch Mobility Solutions. [Sense, localize, monitor: reliable sensor systems](#). 2021 (cit. on p. 6).
- [150] [Structured vs. Unstructured Data: What’s the Difference?](#) June 2021 (cit. on p. 9).
- [151] Naoki Suganuma and Takahiro Uozumi. ‘Development of an autonomous vehicle — System overview of test ride vehicle in the Tokyo motor show 2011’. In: *2012 Proceedings of SICE Annual Conference (SICE)*. 2012, pp. 215–218 (cit. on p. 25).
- [152] Jiankai Sun et al. ‘[Neuro-Symbolic Program Search for Autonomous Driving Decision Module Design](#)’. In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, Nov. 2021, pp. 21–30 (cit. on p. 33).
- [153] Leonard Swiezinski. ‘[Lifecycle of a Jeopardy Question Answered by Watson DeepQA](#)’. In: (2019) (cit. on p. 41).
- [154] Poonam Tanwar, T. Prasad and Mahendra Singh. ‘[Comparative Study of Three Declarative Knowledge Representation Techniques](#)’. In: *International Journal on Computer Science and Engineering* 2 (Oct. 2010), p. 2274 (cit. on p. 13).
- [155] [Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles](#). 2021 (cit. on p. 1).
- [156] Alex Teichman, Jesse Levinson and Sebastian Thrun. ‘[Towards 3D object recognition via classification of arbitrary object tracks](#)’. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4034–4041. DOI: [10.1109/ICRA.2011.5979636](#) (cit. on p. 28).
- [157] Claes Tingvall and Narelle Haworth. ‘[Vision Zero-An ethical approach to safety and mobility](#)’. In: *6th ITE International Conference Road Safety & Traffic Enforcement: Beyond 2000*. 1999 (cit. on p. 22).
- [158] [Traffic manager - carla simulator₂₀₂₂](#). 2022 (cit. on p. 46).

- [159] Jonathan Tremblay et al. *Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization*. 2018. DOI: [10.48550/ARXIV.1804.06516](https://doi.org/10.48550/ARXIV.1804.06516) (cit. on p. 30).
- [160] *Unstructured Data and the 80 Percent Rule*. Aug. 2008 (cit. on p. 8).
- [161] Charlott Vallon et al. ‘A machine learning approach for personalized autonomous lane change initiation and control’. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1590–1595. DOI: [10.1109/IVS.2017.7995936](https://doi.org/10.1109/IVS.2017.7995936) (cit. on p. 15).
- [162] Jessica Van Brummelen et al. ‘Autonomous vehicle perception: The technology of today and tomorrow’. In: *Transportation Research Part C: Emerging Technologies* 89 (2018), pp. 384–406. ISSN: 0968-090X. DOI: <https://doi.org/10.1016/j.trc.2018.02.012> (cit. on pp. 2, 25–27).
- [163] Jaycil Z. Varghese and Randy G. Boone. ‘Overview of Autonomous Vehicle Sensors and Systems’. In: 2015 (cit. on p. 6).
- [164] *Visão Zero 2030*. Apr. 2021 (cit. on p. 22).
- [165] Eleni I. Vlahogianni. ‘Modeling duration of overtaking in two lane highways’. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 20 (2013), pp. 135–146. ISSN: 1369-8478. DOI: <https://doi.org/10.1016/j.trf.2013.07.003> (cit. on p. 37).
- [166] *Waypoint - The official Waymo blog: Waymo and Fiat Chrysler Automobiles (FCA) expand autonomous driving technology partnership*. [Online; accessed 19. Nov. 2021]. Nov. 2021 (cit. on p. 30).
- [167] *Waypoint- using automated data augmentation to advance our waymo driver 2020*. 2020 (cit. on p. 9).
- [168] Lukas Westhofen et al. *Using Ontologies for the Formalization and Recognition of Criticality for Automated Driving*. 2022. DOI: [10.48550/ARXIV.2205.01532](https://doi.org/10.48550/ARXIV.2205.01532) (cit. on p. 39).
- [169] *What is an entity relationship diagram (ERD)?* 2022 (cit. on p. 51).
- [170] *What is Ground Truth?* [Online; accessed 18. Nov. 2021]. Jan. 2021 (cit. on pp. 8, 28).
- [171] Bichen Wu et al. *SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud*. 2017. arXiv: [1710.07368](https://arxiv.org/abs/1710.07368) [cs.CV] (cit. on p. 30).
- [172] Jian-Ru Xue, Jian-Wu Fang and Pu Zhang. ‘A Survey of Scene Understanding by Event Reasoning in Autonomous Driving’. In: *International Journal of Automation and Computing* 15 (Apr. 2018). DOI: [10.1007/s11633-018-1126-y](https://doi.org/10.1007/s11633-018-1126-y) (cit. on pp. 40–41).
- [173] Chen Yan, Wenyuan Xu and Jianhao Liu. ‘Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle’. In: *Def Con* 24.8 (2016), p. 109 (cit. on p. 27).
- [174] Jinsoo Yang et al. ‘Human-like Decision-Making System for Overtaking Stationary Vehicles Based on Traffic Scene Interpretation’. In: *Sensors* 21.20 (2021). ISSN: 1424-8220 (cit. on pp. 1, 37–38).

- [175] [yellowbrick - A suite of visual analysis and diagnostic tools for machine learning](#). Feb. 2022 (cit. on p. 59).
- [176] Shizhe Zang et al. 'The Impact of Adverse Weather Conditions on Autonomous Vehicles: Examining how rain, snow, fog, and hail affect the performance of a self-driving car'. In: *IEEE Vehicular Technology Magazine* PP (Mar. 2019), pp. 1–1. DOI: [10.1109/MVT.2019.2892497](#) (cit. on p. 26).
- [177] Mengshi Zhang et al. *DeepRoad: GAN-based Metamorphic Autonomous Driving System Testing*. 2018. DOI: [10.48550/ARXIV.1802.02295](#) (cit. on p. 33).
- [178] Zhaoxia Zhang et al. 'Researches on Expert System for Automatic Driving Traffic Rules of Unmanned Vehicle'. In: *Journal of Physics: Conference Series* 1069 (Aug. 2018), p. 012016. DOI: [10.1088/1742-6596/1069/1/012016](#) (cit. on pp. 39–40).
- [179] Zixu Zhang and Jaime Fisac. 'Safe Occlusion-Aware Autonomous Driving via Game-Theoretic Active Perception'. In: *Robotics: Science and Systems XVII* (July 2021). DOI: [10.15607/rss.2021.xvii.066](#) (cit. on p. 26).
- [180] Zixu Zhang and Jaime Fisac. 'Safe Occlusion-Aware Autonomous Driving via Game-Theoretic Active Perception'. In: *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, July 2021. DOI: [10.15607/rss.2021.xvii.066](#) (cit. on p. 34).
- [181] Junbo Zhao, Michael Mathieu and Yann LeCun. *Energy-based Generative Adversarial Network*. 2017. DOI: [10.48550/ARXIV.1609.03126](#) (cit. on p. 10).
- [182] Lihua Zhao et al. 'Ontologies for Advanced Driver Assistance Systems'. In: Mar. 2015 (cit. on p. 38).
- [183] Julius Ziegler et al. 'Making Bertha Drive—An Autonomous Journey on a Historic Route'. In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), pp. 8–20. DOI: [10.1109/MITS.2014.2306552](#) (cit. on p. 14).
- [184] Shaked Zychlinski. *dython:nominal*. 2013 (cit. on p. 66).

Appendix A

Statistics of Static Features

STATIC FEATURES	STATICS										OUTCOME				
Night and Day	Day					Night					Successful (legal) Successful (illegal) Non-successful (no collision) Non-successful (collision) No attempt to overtake Total of simulations				
	af		rf			af		rf							
	20		0.21			9		0.05							
	35		0.37			82		0.45							
	21		0.22			47		0.26							
	14		0.15			18		0.10							
	4		0.04			26		0.14							
	94					182									
Horizon Line	Yes					No					Successful (legal) Successful (illegal) Non-successful (no collision) Non-successful (collision) No attempt to overtake Total of simulations				
	af		rf			af		rf							
	7		0.27			22		0.09							
	10		0.38			107		0.43							
	2		0.08			30		0.12							
	4		0.15			64		0.26							
	3		0.12			27		0.11							
	26					250									
Type of ego vehicle	S		M		L		V		T		Successful (legal) Successful (illegal) Non-successful (no collision) Non-successful (collision) No attempt to overtake Total of simulations				
	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf					
	3	0.06	7	0.13	4	0.10	5	0.07	10	0.16					
	23	0.48	23	0.43	13	0.32	33	0.46	25	0.40					
	4	0.08	7	0.13	6	0.15	9	0.13	6	0.10					
	10	0.21	13	0.24	10	0.24	19	0.27	16	0.26					
	8	0.17	4	0.07	8	0.20	5	0.07	5	0.08					
	48		54		41		71		62						
Type of Preceding vehicle	S		M		L		V		T		MC		B		Successful (legal) Successful (illegal) Non-successful (no collision) Non-successful (collision) No attempt to overtake Total of simulations
	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf			
	5	0.10	6	0.09	5	0.10	7	0.13	6	0.12	0	0	0	0.00	
	21	0.41	29	0.45	18	0.37	21	0.38	26	0.52	2	0.4	0	0.00	
	7	0.14	6	0.09	8	0.16	6	0.11	2	0.04	2	0.4	1	1.00	
	16	0.31	15	0.23	11	0.22	17	0.30	9	0.18	0	0	0	0.00	
	2	0.04	8	0.13	7	0.14	5	0.09	7	0.14	1	0.2	0	0.00	
	51		64		49		56		50		5		1		
Waiting Time	Min		Mean		Max		σ								
	0.05		0.44		2.63		0.63								
	0.05		1.30		5.03		0.98								
	0.05		2.38		23.17		4.19								
	0.05		1.79		9.33		1.86								
	1.12		8.00		20.05		7.75								
	0.05		2.18		23.17		3.74								
Overtaking Time	Min		Mean		Max		σ								
	3.33		7.68		15.00		3.10								
	1.67		4.36		16.00		2.37								
	4.58		11.16		26.72		5.16								
	0.75		2.80		14.78		2.81								
	0.00		0.00		0.00		0.00								
	0.00		4.64		26.72		4.17								
Number of Vehicles	3		4		5		6								
	ra	rf	ra	rf	ra	rf	ra	rf	ra	rf					
	9	0.15	6	0.08	11	0.14	3	0.05							
	29	0.49	35	0.46	34	0.44	19	0.30							
	8	0.14	11	0.14	8	0.10	5	0.08							
	9	0.15	15	0.20	19	0.25	25	0.39							
	4	0.07	9	0.12	5	0.06	12	0.19							
	59		76		77		64								

Figure A.1: Statistics of Static Features.