Model of autonomous driving in a roundabout scenario

Tesi di Laurea Magistrale in
Mechanical Engineering - Ingegneria Meccanica

Author: **Lorenzo Uccello**

Student ID: 976403
Advisor: Prof. Gianpiero Mastinu
Co-advisor: Prof. Giorgio Previati
Academic Year: 2022-23

Abstract

Autonomous Vehicles are one of the future trends defined by the European Commission and, more broadly, a crucial field of research for future private and public mobility. This thesis work aims to create a Digital Twin of a real roundabout scenario within which testing the capabilities of Autonomous Vehicles and the effect of their penetration rate in the simulation environment, specifically considering 20% and 80% AVs. DRL (Deep Reinforcement Learning) policies will move Connected and automated vehicles taking decisions along all the simulations. The connection will be based on edge computing technology and 5G mobile protocol, building a V2N2V communication. Such a complex simulation environment is part of an extended project called AI@EDGE that has the objective of studying the effect of many types of automation inside a real network, among which road mobility. Firstly, the simulation environment is built by creating effective communication between all platforms employed: Flow, which represents the Artificial Intelligence Network; SUMO, the microscopic traffic simulator and the Driving Simulator, used to introduce the human factor to the simulation and making this project unique from previous research activities in this field and capable of achieving innovative results. Once the simulation environment has been created, preliminary tests were carried out in order to optimise the system configuration by developing algorithms for trajectories interpolation and changing the roundabout design to avoid negligence in compliance with the rules of right of way. Secondly, a *Replay* scenario has been developed to study the driving behaviour of AVs in relation to human user perception from a microscopic point of view. The comfort and safety of Autonomous Vehicles have been tested and improved for a comprehensive configuration. Finally, an ultimate simulation environment has been built considering also a traffic calibration to replicate reality as much as possible. Final tests yielded key results: AVs succeeded in considering various environment geometries and driver behaviours; human users felt safer and preferred the scenario with a higher percentage of AVs; the traffic calibration led to a better-performing scenario.

Keywords: Autonomous Vehicles, 5G, Roundabout, Digital Twin, Replay, DRL.

Abstract in lingua italiana

I veicoli autonomi sono uno dei *Future Trends* definiti dalla Commissione Europea e, più in generale, un campo di ricerca cruciale per la futura mobilità privata e pubblica. Questo studio si pone l'obiettivo di generare un gemello digitale di una rotatoria all'interno del quale testare le capacità dei veicoli autonomi e l'effetto del loro grado di penetrazione sul mercato, considerando due percentuali di veicoli autonomi: 20% e 80%. I veicoli connessi e automatizzati sono guidati da algoritmi di DRL che scelgono le azioni da compiere e vengono connessi utilizzando la tecnologia *Edge Computing* e il protocollo *mobile* 5G, costruendo una struttura V2N2V. Questo ambiente simulativo fa parte di un progetto più esteso chiamato AI@EDGE che ha l'obiettivo di studiare l'effetto di vari tipi di automazione all'interno di un sistema complesso, tra cui la mobilità stradale. In primo luogo, è stato costruito l'ambiente simulativo sviluppando una comunicazione efficace tra tutte le piattaforme impiegate: Flow, che rappresenta la rete di intelligenza artificiale; SUMO, il simulatore di traffico microscopico e, infine, il simulatore di guida utilizzato per introdurre il fattore umano nella simulazione. Il DriSMi rende il progetto unico rispetto alle precedenti attività di ricerca in questo campo e permette di raggiungere risultati innovativi. Creato l'ambiente simulativo, sono stati effettuati test preliminari al fine di ottimizzare la configurazione di sistema, sviluppando algoritmi per l'interpolazione delle traiettorie e modificando la struttura della rotatoria per evitare negligenze nel rispetto delle regole di precedenza. In secondo luogo, è stato sviluppato un *Replay* per studiare il comportamento di guida degli AVs in relazione alla percezione dell'utente umano. Sono stati testati e implementati il comfort e la sicurezza dei veicoli autonomi così da ottenere una configurazione completa. Infine, è stato costruito l'ambiente simulativo definitivo considerando la calibrazione del traffico per replicare la realtà. I test finali hanno dato risultati fondamentali: i veicoli autonomi sono riusciti a gestire diverse geometrie e comportamenti dei conducenti; gli utenti si sono sentiti più sicuri e hanno preferito lo scenario con una maggiore percentuale di veicoli autonomi; la calibrazione del traffico ha portato allo sviluppo di uno scenario più performante.

Parole chiave: Veicoli Autonomi, 5G, Rotatoria, Gemello Digitale, Replay, DRL.

Contents

Introduction

Autonomous driving has received special attention in recent years becoming one of the main future trends highlighted by the European Commission. If implemented correctly, this technology will enable a revolution in both public and private mobility bringing major improvements in user accessibility and management of the entire network. The development of Autonomous Vehicles will make possible, among other enhancements, a reduction in traffic accidents, pollution and an increase in the overall efficiency of the transportation sector.

The objective of this thesis is to create a Digital Twin of a real roundabout scenario in which testing some capabilities of connected and automated vehicles and the effect of their penetration in the simulation environment, specifically considering 20% and 80% of Autonomous Vehicles. The roundabout is one of the most widely used intersections for traffic management. It also turns out to be one of the most complex scenarios an autonomous vehicle faces in the city environment because of its structure and the rules of right of way that designate it. AVs will be controlled by a DRL (Deep Reinforcement Learning) policy trained to respect some KPIs, such as the crossing time, the pollution or the comfort and safety of the human user.

Such a complex structure is integrated into a European extended project called AI@EDGE which aims to create a connected-compute fabric useful for various areas of automation. Inside this structure, the main and relevant components are three:

1. the driving simulator, i.e., DriSMi, which allows introducing the human factor directly to the simulation environment, making this study capable of innovative experimental results;
2. the Artificial Intelligent Framework which contains the DRL policy and governs the AVs' actions in the simulation. In this case study the RL algorithms are applied using Flow and its library RLlib;
3. the microscopic traffic simulation built in SUMO, primarily responsible for the environment design and all operating logic of the network and the vehicles within it.

Automated vehicles will be connected considering a particular type of communication: V2N2V. This is a declination of V2I communication in combination with V2V, for which it is considered a piece of the infrastructure placed in the centre of the roundabout that collects movement information from all AVs within a certain distance radius. The information is then sent to the single AVs if requested, thereby also developing inter-vehicular communication. Finally, the data generated by vehicles and infrastructure will be exchanged considering the 5G edge computing technology combining the benefits of local and cloud computing; this completes the foundation of a complex Digital Twin of the real scenario. If achieved correctly, such a system may allow much faster progress in the development of this solution and a major reduction in the associated costs up to the creation of the actual prototype. More in-depth, this thesis work includes the construction of the simulation environment, starting with a preliminary one to get to its final configuration, and the development of the communication between all different software implemented which differ in mechanics and purposes and have never been used in the same simulation. Taking into consideration the limitations in terms of the communication time step, it becomes necessary the establishment of algorithms proposed to interpolate the trajectories of all vehicles in the network and to display them correctly and the implementation of the traffic scenario defining the maximum number of involved vehicles. This environment will then be employed to test the ability of automated vehicles to move within the circulatory roadway. It will also be developed a *Replay* scenario considering the vehicle dynamics and behaviour of all its main mechanical components, which will be used to test the usability of the created Autonomous Vehicles by human users, thus considering their comfort and overall feeling of safety.
This will result in a great step forward in the research activity of the AI@EDGE project and, therefore, in obtaining valid results in order to present a possible future solution for this industry.

1 | Literature overview

This first chapter presents an overview of the literature about the thesis project. The literature will be divided into subsections with respect to all the key topics covered. The previous work related to roundabout optimisation, automated vehicles behaviour, simulation programs and their recent developments will be analysed. All the information presented below will be discussed in the following chapters, considering a detailed comparison with the results obtained during the analysis.

1.1. The roundabout

Roundabouts are a specific type of circular intersection and one of the most critical sceneries for Autonomous Vehicles. They are also safer [35] and allow for a greater capacity, moderating traffic [19], with respect to normal junctions and for this reason, are preferred and largely used. The roundabout is a special traffic scenario of merging roads without traffic lights. They are typically more complex than other types of junctions, but with respect to signalised junctions they manage to keep a reduced number and percentage of vehicles in conflict [52]. Many variables need to be considered to make a correct decision: the number of vehicles inside the roundabout, their direction and speed, the behaviour of every driver and the number of lanes inside the roundabout. Works have been presented to enhance the roundabout's safety, considering signals control methods [32, 57] or studying the advantages of metering signals with unbalanced traffic flows.

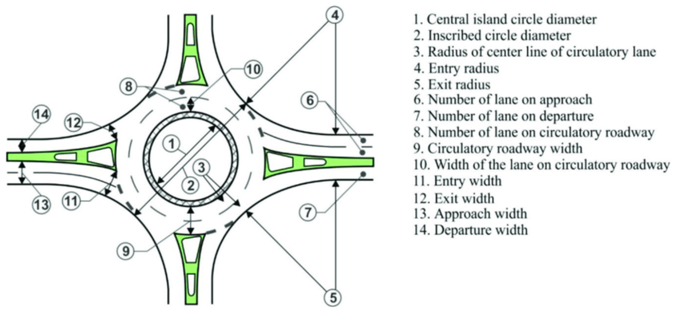

1. Central island circle diameter
2. Inscribed circle diameter
3. Radius of center line of circulatory lane
4. Entry radius
5. Exit radius
6. Number of lane on approach
7. Number of lane on departure
8. Number of lane on circulatory roadway
9. Circulatory roadway width
10. Width of the lane on circulatory roadway
11. Entry width
12. Exit width
13. Approach width
14. Departure width

Figure 1.1: Key dimensions of a generic roundabout [17].

Roundabouts are mainly described by the lane numbers and the leg numbers, equal to 2 and 4 for the roundabout in Figure 1.1. The lane number refers to the number of lanes inside the circulatory roadway. Leg number defines the number of exits of the roundabout. If both of these variables become larger, the complexity of the scenario becomes greater: vehicles may consider also lane change inside the roundabout and they also need to pay attention to many more control points while driving inside the roundabout and during all entry and exit actions which, in any case, are fewer in number than at intersections of another nature. More in general, roundabouts performances depend on the design features that significantly affect both crash frequency and severity. For this reason, it has gained great importance recently to develop official designs and standards. As stated in [49] roundabouts can be classified into three main categories:

1. mini-roundabouts. Small roundabouts with a fully traversable central island. The diameter of the inscribed circle (ICD), which is the largest circle that can be fitted into the junction outline, ranges between 13 m and 28 m.
2. single-lane roundabouts. This type of roundabout is characterised by having all single-lane legs entering it. The ICD is between 27 m and 55 m. A splitter island should be provided at all the legs.
3. multi-lane roundabouts. They have at least one entry with two or more lanes. The ICD ranges between 30 m and 100 m. The number of lanes inside the circulatory roadway can vary accordingly to the number of legs and their lanes.

	Mini-roundabouts	**Single-lane roundabouts**	**Multi-lane roundabouts**
Circulating lanes	1	1	2 or more
Typical ICD	13 m to 28 m	27 m to 55 m	30 m to 100 m
Desirable entry speed range	54 km/h to 72 km/h	72 km/h to 90 km/h	90 km/h to 110 km/h
Central island	Traversable	Raised with traversable truck apron	Raised with traversable truck apron

Table 1.1: Roundabouts classification and design parameters values.

Many are the design variables, which can be considered to optimise the roundabout geometry. Some of them are the central island, which represents an obstruction to traffic and should be easily recognisable; the speed control, which refers to the need of achieving a specific vehicular speed through the roundabout; the entry path radius, which is an important determinant for safety since it governs the speed of vehicles through the junction and whether drivers are likely to give way to circulating vehicles. In [61] is illustrated an example of roundabout's geometry optimisation considering its internal radius and other design elements.

The authors in [49] propose a critical review of the Italian regulations defining geometry design practices, guidelines and standards. The most relevant criticalities refer to the deficient current standard, which only refers to intersections in general and no specifications are made on roundabouts. Moreover, traversable islands are not considered for mini-roundabouts, making them not able to ensure accommodation of larger vehicles; splitter islands design is not treated at all. Finally, inconsistencies, design errors and wrong requirements make the standard not able to achieve some of the main design objectives, such as optimal speed control.

To add an additional layer of complexity, we may also consider the human factor, which in this case study results in possible driving errors, such as incorrect use of turn signals. Crosswalks may also be present, changing the vehicle flow behaviour. For all these reasons it is nowadays relevant to discuss vehicles' and Autonomous Vehicles' behaviour in roundabouts.

1.2. Calibration of the traffic inside a roundabout

The transportation community makes use of two distinct types of simulators:

- Microscopic Traffic Simulation (MTS) models for evaluating the system performance of transportation networks;
- Driving Simulators (DS) to evaluate the response of individual human subjects.

MTS is closely related to obtaining the simulated environment used in further simulations to train and test the AVs policy, which needs to be as much as possible similar to reality. In order to achieve this goal, real measurements can be made of traffic conditions and roundabout geometry. Many are the factors to be taken into account, such as the type of vehicles considered, bikes, cars, heavy-duty trucks or busses, pedestrian crossings, and the flows which every vehicle can follow. In recent years, experiments have been made on the integration of these two separate models to obtain better results and simulation environment performances. [30] is an example of that. It demonstrates how these two models can interact, getting to a vehicle in the MTS able to mimic the DS' vehicle behaviour. This is reflected in obtaining better results for the trained AVs policy, able to lower important indicators, such as travel time, CO_2 emissions and fuel consumption.

This thesis work considers a microscopic simulation, in which both the MTS and DS are used to get a complete description of reality. This is a widely accepted tool to evaluate, assess and develop new design configurations, also thanks to its ability to keep a track of individual vehicle movements on a second to sub-second basis. To get data coherent with reality, it is crucial to calibrate the microscopic simulation environment and ensure that its characteristics are the same or as much as possible similar to the same real scenario, in terms of driver's characteristics or infrastructure in the network. There are many variables, which can be changed to modify the simulation behaviour and all of them must be considered to get the final optimal calibration. As described in [46] the whole methodology can be divided into three main phases:

1. the first one is called pre-modelling. During this first stage, the micro-simulation platform is chosen a measure of effectiveness (MOE) is identified all the requirements in terms of collection and data processing are defined. In this case study the platform is SUMO and the most important data are queuing, flows and directions of all vehicles in the network;

2. during the second phase the simulation platform is used to run many simulations and understand the weight and effect of any variable. Thanks to these runs, it is also possible to understand if the microscopic model demonstrates significant differences with reality;

3. the final stage is the calibration one. This last phase can be divided into four subsections. The first one is the evaluation of the sensitivity of all the variables in the platform. Inside a pre-set of possible sensitive parameters, the most sensitive ones are chosen. During the third stage, the sensitive variables are used to calibrate the model and get it as close as possible to reality. The final step coincides with the calibration optimisation, using for example a genetic algorithm inside MATLAB.

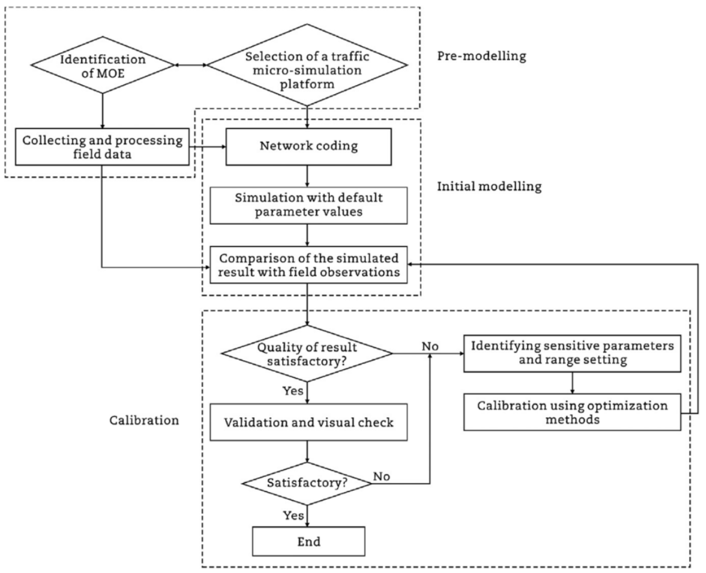

Identification of MOE
Selection of a traffic micro-simulation platform
Pre-modelling
Collecting and processing field data
Network coding
Initial modelling
Simulation with default parameter values
Comparison of the simulated result with field observations
Quality of result satisfactory?
No
Identifying sensitive parameters and range setting
Yes
Calibration
Validation and visual check
Calibration using optimization methods
Satisfactory?
No
Yes
End

Figure 1.2: The three phases of microscopic simulation calibration [46].

Calibration using microscopic simulation data is also useful to obtain the car-following models used in the simulation. Example of this are reported by [29, 60]. One of the key points is the correlation between all environment parameters and, therefore, their joint estimation.

Finally, the model obtained can be used for all further experimental analyses. Calibration improves the robustness of the simulation environment, leading to reliable results. In this case study, calibration is carried out only on the final simulation environment and not on the preliminary one.

1.3. Autonomous Vehicles

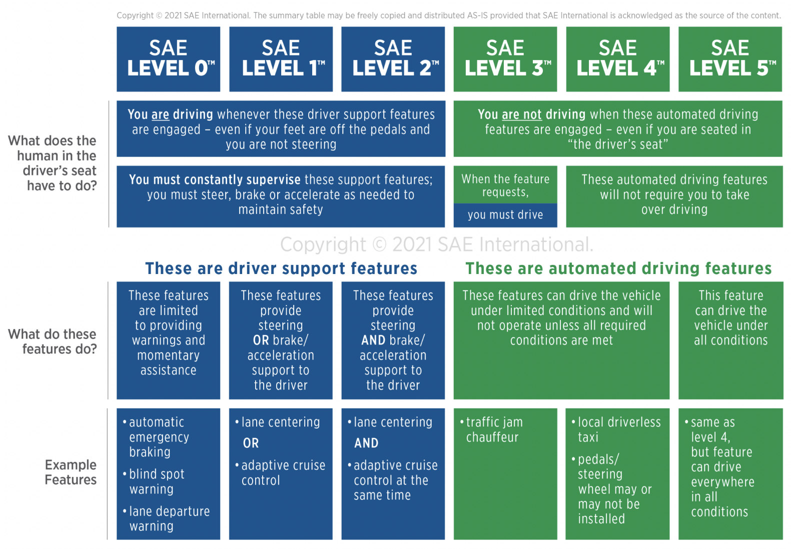
Autonomous Vehicles are directly linked to transportation efficiency: making it safer and improving access to mobility worldwide. Autonomy will be adopted if it creates a better human experience, a crucial detail which must be considered in any AV application. To set a greed-upon standard on what is an autonomous vehicle, the Society of Automotive Engineers (SAE) developed a classification system defining the degree of automation of a car [34]. These levels define a total of five possible control actions, that a vehicle can act on and how these actions are controlled. The actions considered are:

- steering;
- acceleration and deceleration;
- monitoring of driving environment;

- fallback when automation falls;
- automated system control.

The resulting 6 levels of automation are:

1. Level 0, no automation. None of the actions above is handled by the vehicle;
2. Level 1, driver assistance. The vehicle takes care only of acceleration and deceleration only for some driving modes;
3. Level 2, partial automation. For some driving modes, the vehicle handles steering and acceleration/deceleration;
4. Level 3, conditional automation. The vehicle handles steering, acceleration, and deceleration and monitors the driving environment only for some driving modes;
5. Level 4, high automation. Automation takes care of steering, acceleration, and deceleration and does not require the driver to fall back in case of failure for some driving modes;
6. Level 5, full automation. The vehicle is responsible for all the actions defined for any driving condition. Any driver action is required.



	SAE LEVEL 0™	SAE LEVEL 1™	SAE LEVEL 2™	SAE LEVEL 3™	SAE LEVEL 4™	SAE LEVEL 5™
What does the human in the driver's seat have to do?	**You are driving** whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			**You are not driving** when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering **OR** brake/ acceleration support to the driver	These features provide steering **AND** brake/ acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met		This feature can drive the vehicle under all conditions
Example Features	• automatic emergency braking • blind spot warning • lane departure warning	• lane centering **OR** • adaptive cruise control	• lane centering **AND** • adaptive cruise control at the same time	• traffic jam chauffeur	• local driverless taxi • pedals/ steering wheel may or may not be installed	• same as level 4, but feature can drive everywhere in all conditions

Figure 1.3: Summary of SAE international's levels of driving automation for on-road vehicles [34].

Beyond these levels, two different AI systems can be considered. The first one is human-centred autonomy, for which AI is not fully responsible for its actions (levels 1, 2 and 3). The driver must pay attention to the vehicle's actions and take control if needed. The

second system is full autonomy (levels 4 and 5). The AI vehicle is fully responsible for its actions, also from a legal point of view: the vehicle can ask for human help, but it is not guaranteed to ever receive it.
In this thesis work, fully Autonomous Vehicles at level 5 are considered, since the most difficult scenario is also counted from the automation point of view. For this specific condition, it is also important to analyse the human feedback linked to the use of AVs. Actually, human feelings will also be considered to get not only an AV able to take corrective actions to avoid crashes but also an AV that a human would choose as a real daily alternative to human-driven vehicles.

1.4. Autonomous Vehicles control in roundabouts

The human factor remains the most important cause of traffic accidents. Various studies have been concerned with describing the behavioral factors related to it and the implications for urban and non-urban traffic [13, 20, 50]. Autonomous Vehicles are the big future trend, which will enable the optimisation of wheeled traffic. Therefore, the idea of a coordinated traffic system is gaining attention and becoming one of the most studied solutions for future vehicles. This applies also to the roundabout and its peculiar traffic scenario, which makes this use case difficult also for AVs [55]. It is necessary to pay attention to lateral control, the choice of the right exit lane and compliance with the rules of right of way [4, 5, 23]. More in general, two are the main problems to be considered in a roundabout for AVs:

- the vehicles' communication;
- the optimal coordination and control;

The first one defines the way Autonomous Vehicles exchange information. Many communication levels can be considered, such as vehicles to vehicles (V2V) or vehicles to infrastructure (V2I) and vice-versa (I2V) [10]. One of them may be chosen or combinations can also be considered to optimise data exchange. Works have been presented to describe V2V protocols enabling dedicated short-range communications (DSRC) or wireless access in vehicular environments [7, 8]. One of the most important variables to be considered is the delay with which the information in received by all other vehicles and if this delay may contribute to wrong decision-making by AV.

The second problem describes the way the information received is handled by AV and all the related algorithms to optimise the decision-making process, both in terms of the time needed to take a final judgment and the error made, which can lead to possible crashes.

Many works related to this topic have been presented, showing its complexity and the breadth of adoptable possibilities. In [9] is described an optimal coordination logic for AVs considering a single-lane roundabout, by dividing it into two different zones, clustering zone and merging-execution zone, and adapting the AVs' behaviour to the specific zone they are in during their movement. [16] proposes a decision making algorithm for vehicles approaching a roundabout, considering two different traffic conditions.

A relevant detail on which depends the specific solution adopted is the percentage of AVs in the traffic scenario considered. If a single AV needs to deal with all other non-Autonomous Vehicles, there will be no V2V communication and also the infrastructure will not be upgraded for such a small percentage of AVs. On the other hand, if all the traffic is composed of Autonomous Vehicles, V2V and V2I communications will be implemented and their role will depend on the specific optimisation algorithm adopted. The real problem stands between these two opposing conditions when not all the vehicles are AVs and some of them are still humanly driven. This is the field of study to which this thesis work refers since it is not only the most problematic one but also the one that society will need to face in the shortest time range. Therefore, the impact of connected Autonomous Vehicles on real human-driven ones needs to be evaluated, both in terms of using these vehicles and interacting with them. Psychological studies have been conducted on the perception of AVs, showing how even if full AVs are proposed, humans may be emotionally conditioned not to use them. The mix between humans and Autonomous Vehicles must be optimised.

It is also relevant to consider the driving behaviour of the AVs and how different ones act inside the roundabout and, more in general, in many simulation scenarios. Many are the details, that need to be considered in order to create a complete driver model, such as vehicle safety, ride comfort or travel distance. In [26] a game theoretic decision-making framework using model predictive control (MPC) is designed. Three different driving styles are considered: aggressive, conservative and normal. The algorithm obtained showed the capability to ensure safety guaranteeing the personalised driving demands.

Finally, the specific movements of AVs inside the roundabout need to be addressed. [11] shows an optimal intersection control system able to produce an 80% reduction in delay, a 42,5% and 40% reduction in fuel consumption and CO_2 emissions, respectively.

1.4.1. V2V communication

Vehicle-to-vehicle communication enables vehicles to exchange information about their speed, location, brake status, steering wheel angle and vehicle path history and prediction. These last two refer to the set of previous positions and the future trajectory and confidence, respectively. This creates a 360-degrees awareness of other vehicles in the network. V2V enables safety path design and helps the vehicle avoid potential crashes. It has the goal to facilitate efficient and reliable communication without relying on a global system for mobile communication networks (GSM). GSM relies typically on third-party infrastructure bandwidth, which is generally limited and sometimes not adequate. Three are the key areas on which designers must debate:

- creating communication standards to avoid system failures and to allow for an optimisation of the communication itself carried on by the scientific community as a whole;
- developing a system able to withstand all possible interferences with all other communication devices inside modern vehicles, such as Bluetooth or other WiFi devices. For example, antennas play a key role in enabling the address of multiple protocols;
- providing safe connectivity integrated with all other types of communications used by an AV to work correctly.

This type of communication uses GPS with DSRC protocols in the bandwidth of 75 MHz with a 5.9 GHz spectrum and an approximate communication range of 1000 m [12, 47]. The main components needed to allow for this communication are:

1. DSRC radio, receiving and transmitting data over antennae;
2. GPS receiver, providing vehicle position over time to the DRSC radio;
3. memory, storing security certificates
4. safety application electronic control unit (SAECU), running the safety applications of the system;
5. vehicle's internal communication network (VICN) interconnecting vehicle comparts;
6. driver-vehicle interface, displaying and generating warnings to the driver;
7. security credential management system, verifying V2V certificates. Protection from malicious attacks is a relevant theme for AVs. There may be many types of attacks that an AV can receive, such as message spoofing, message replay attacks or integrity attacks;
8. Antennae

V2V communication can be used by many different types of vehicles, such as cars, buses, trucks or motorcycles. It could also be used on bicycles making vehicles able to predict also their behaviour and, finally, choose the better action to be taken in as many as possible different scenarios [18].

1.4.2. V2I and I2V communication

Vehicle-to-infrastructure communication represents one of the key components of next-generation Intelligent Transportation Systems (ITS). V2I technologies are in charge of sourcing traffic data and wirelessly providing information to all vehicles in the network. As part of the ITS, they can also be used to help human drivers estimate accurate arrival times or path conditions. In this case, a control station is responsible for managing and processing information coming from all vehicles in the network and for returning warnings or recommendation commands. In [48] is described an example of the system configuration to account for this type of communication. The main components to be defined are:

- a specific data structure, so as to be usable by any device in the area;
- an intelligent management traffic system. An example is WAVE, which assigns a priority to all incoming messages based on the possible risk linked;

V2I communications are more general than V2V in terms of users, which make use of the information provided. They are also essential for specific types of scenarios, like intersections, and in order to prevent possible risks linked to obstacle detection and avoidance.
This case study considers a special configuration of V2I communications called V2N2V. A piece of infrastructure is placed in the centre of the roundabout and used to link all AVs in the network. Vehicles, when in range, can ask for all other AVs information, such as position, speed or direction. They also communicate all their details to the network, making this information available for all possible clients.

1.4.3. V2X communication

Vehicle-to-everything communication is between a vehicle and any other device, which can directly interact, exchanging data at many levels of information: it integrates V2I, V2V, V2P (vehicle-to-pedestrian) or V2G (vehicle-to-grid) communications. V2X can be base on:

- a WLAN technology, which works directly in V2V and V2I communications. It does not require any communication infrastructure, since it is short-range and it ensures low latency. Messages can be of many types, such as Cooperative Awareness Messages (CAM), Basic Safety Messages (BSM) or Decentralised Environmental Notification Messages (DENM). As described in subsection 1.4.1, radio technology is a relevant part of the communication structure. It is crucial to define the channel access scheme to optimise the communication effectiveness, as described in [28];

- a cellular-based V2X communication. The third Generation Partnership Project (3GPP) decided to study the feasibility of, first LTE technology and then 5G. This second solution can solve some of the problems that the DSRC system suffers from: short-range, large channel access delay and huge capital investment. In [24] strong aspects of LTE are described such as wide coverage, high capacity and high penetration. The authors also describe some of the main challenges of this technology and how 5G could solve many of them in future applications.

[22] presents a detailed description of the first standard for 5G NR V2X communications developed by 3GPP and a reference tutorial that introduces the major 3GPP standard developments essential to understand how V2X communications operate.

Finally, V2X can be used for a variety of applications, such as safety [2], traffic efficiency and passenger infotainment. In addition to these uses, V2X supports also forward collision warning, parking discovery or curve speed warning.

1.5. Connected, Cooperative & Automated Mobility

June 23, 2021 saw the officialisation of the Connected, Cooperative and Automated Mobility (CCAM) partnership [14]. The goal of this European project is to create a more user-centred and inclusive mobility system, considering some of the most critical aspects of mobility in general: safety in road transport, environmental impact, traffic smoothness and inclusive mobility, making it more accessible for all. To achieve this goal, CCAM proposes an attentive and assiduous partnership between research centres, companies and many other industry players. Testing and demonstration projects are the keys in order to accelerate the innovation and implementation of Connected Autonomous Vehicles (CAVs). CCAM partnership was created to support many European and global projects, such as the UN Sustainable Development Goals, the European Green Deal and Sustainable Mobility Strategy.
Achieving such a complex and general goal presents various difficulties nowadays, the most important among them are:

1. there is not sufficient demand for CAVs, since society does not understand their real future potential. This appears to be related to a still incomplete analysis of the long-term implications, benefits and impact of CCAM;

2. current CCAM solutions and R&I investments are not sufficient yet. This makes

this solution not mature enough to penetrate international markets and become a real solution;

3. demonstrations of such a complex technology are today really limited since a complex cross-sectorial value chain is still required to build a complete CCAM solution.

Besides the partnership, also a CCAM not-for-profit Association has been created representing more than 180 innovation stakeholders. The aim of this organisation is to build and nurture cooperation between these and other sector stakeholders, accelerating the development of new technologies and their deployment in real life. Looking at its statute, the following actions are defined as crucial to reach this goal:

- contributing to the design and writing of CCAM projects to be published under Horizon Europe;
- participating in the update of the CCAM Strategic Research and Innovation Agenda;
- helping the European standardisation and regulation development;
- managing and expanding the network of stakeholders engaged in the field of CCAM;
- supporting a joint action with other European partnerships and national activities.

CCAM is a crucial example to understand the relevance of this field of study and the future effort that will be needed to achieve valid results. Having a partnership from the beginning of this path will make much easier the transition phase between our present condition and the one attainable through progress in this area of study.

1.6. 5G and Autonomous Vehicles

5G (fifth Generation) refers to the latest technology of mobile networks created with the aim to reach better communication efficiency. It enables greater adaptability of network applications thanks to:

- the optimisation of network resources through the definition of virtual sub-networks. This concept is called slicing and allows the design of dedicated networks each adapted to serve one type of service [59];
- the use of Software-defined networking (SDN), which differs from a traditional browser, since the virtual network is controlled via software and not hardware. Three are the main parts composing an SDN: applications, which communicate requests; controllers, which translate the requirements and decide how to route data packets and networking devices, which receive the information given by the controllers.

Many advantages are linked to SDN, such as communication agility and speed. On the other hand, by centralizing all the network intelligence on the controllers, the control plane becomes a point of failure for the whole system [15];

- the capacity to handle a greater number of devices, crucial for AVs applications [37];
- the minimisation of communication latency, making it exploitable for real-time applications like automotive ones [38];
- a significant reduction of energetic consumption [21].

Figure 1.4 shows a complete taxonomy of edge computing in 5G to highlight the variety of applications and the potentialities of this technology.

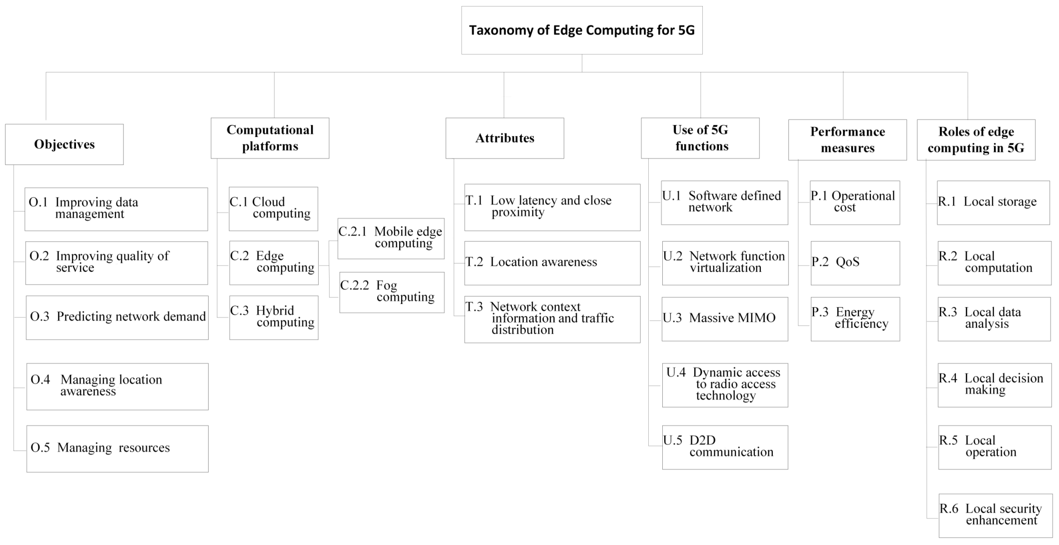

Taxonomy of Edge Computing for 5G
Objectives
O.1 Improving data management
O.2 Improving quality of service
O.3 Predicting network demand
O.4 Managing location awareness
O.5 Managing resources
Computational platforms
C.1 Cloud computing
C.2 Edge computing
C.3 Hybrid computing
C.2.1 Mobile edge computing
C.2.2 Fog computing
Attributes
T.1 Low latency and close proximity
T.2 Location awareness
T.3 Network context information and traffic distribution
Use of 5G functions
U.1 Software defined network
U.2 Network function virtualization
U.3 Massive MIMO
U.4 Dynamic access to radio access technology
U.5 D2D communication
Performance measures
P.1 Operational cost
P.2 QoS
P.3 Energy efficiency
Roles of edge computing in 5G
R.1 Local storage
R.2 Local computation
R.3 Local data analysis
R.4 Local decision making
R.5 Local operation
R.6 Local security enhancement

Figure 1.4: Taxonomy of edge computing in 5G [27].

Edge computing combines both local and cloud computing, being able to have the strengths of these two solutions. In this way, the information produced by any client in the network is uploaded only if relevant for the system as a whole and later made available to other clients. Instead, the rest of the data is stored locally to use them real-time inside each client. This is crucial, since cloud computing alone introduces risks such as safety, storing space available and costs related as described in [51].

For all these reasons, 5G satisfies all the requirements of AVs and can lead to new and better results in this field of application. AVs generate a large amount of data emitted by sensors mounted on vehicles. This information must be sent to all the other vehicles, enabling V2V communications, and to the infrastructure, which will then use it to communicate with other vehicles or infrastructure components. Cloud computing technologies, such as 5G, make this communication efficient, by reducing the time needed to send it and allowing better storage both in time and quantity. A complete AVs network is composed of many nodes, all required to have a complex and functional system for AVs.

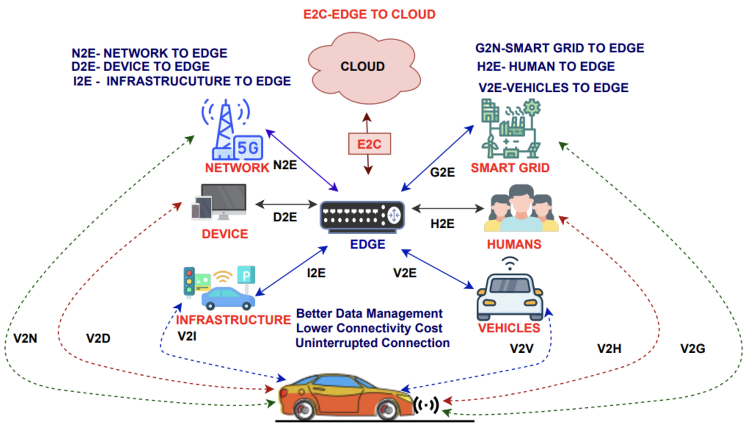


Figure 1.5: All levels of communication in an AVs mobility system [25].

Multi-access Edge Computing (MEC) extends the capabilities of cloud computing by bringing the users closer to the edge of the network. This reduces latency and data congestion and real-time analysis can be carried out. The heart of the systems is represented by the edge, which communicates with all the other devices inside the network and with the cloud, in which all data are stored and made available to all AVs. The edge receives from vehicles, infrastructure nodes, humans and so on all information needed for path planning, obstacle detection and many other actions of the grid as a whole. The information is enormous and must be sent effectively to any device requiring it to take decisions. One of the most important 5G features is the proximity service (ProSe), which represents services available to a group of devices in a specific area and is fundamental to provide awareness to a specific AV about all the other AVs, infrastructures and criticalities in the environment. ProSe is best suited for identifying moving vehicles on the road. Network slicing, as previously mentioned, separates the networks logically: in the case of AVs networks can be divided considering the specific application and requirements, such as safety, infotainment applications and mission-critical applications. More in detail:

- safety can be achieved by using 5G for local perception in short range. Data of the road network must always be available to the vehicles, which will access the infrastructure node and thanks to which it can design the complete route to the destination in advance. Autonomous navigation includes also obstacle detection and avoidance, leading to a safe journey. With respect to traditional communication technologies, 5G leads to better performances also in poor climatic conditions, with snow or fog;

- real-time decision making is one of the most important AV features and it is strictly dependent on Ultra-Reliable Low Latency Communications (URLCC). This is directly linked to the ability to handle the massive amount of data generated by every AV in the environment. AVs need a target latency of 1 millisecond, which only URLCC can achieve.
- vehicle-to-everything (V2X) and 5G enable AVs to visualise objects and obstacles not in the field of view of the vehicle. This is beneficial both in terms of safety and waiting time at intersections.
- a wide coverage with minimum speed is expected in AV for data exchange. eMBB (enhanced Mobile Broadband) is an extension to the 4G LTE networks and provides higher data rates, improved latency and coverage area. Larger bandwidth, data density and lower latency can, therefore, be achieved.

5G technology is also relevant for DRL vehicle training and analysis. It enables the possibility to share the data acquired and processed by any of the vehicles in the network and to use this information to train other vehicles. Collaborative analysis improves the driving experience and must be supported by the efficiency and reliability of the data sharing. This learning process is called Federated learning (FL) and refers to a machine learning technique that trains an algorithm across multiple decentralised edges holding local data and sharing it in clouding services. In [25] the authors have proposed a scheme based on asynchronous FL that uses DRL to train policies (which represent the specific vehicles) at the edge nodes. Considering FL and more in general mobile communications, it is crucial to consider a safe exchange of data. [44] describes a DRL learning framework with a blockchain-based secure FL framework to prevent malicious or unreliable participants.

All of these features make 5G technology a real resource to extend and develop AVs mobility. Beyond that, 5G makes a real difference if used for the network as a whole and, therefore, for all the different services it must ensure to the users. For this reason, this technology is the one considered in this analysis, which is part of a greater project trying to access and demonstrate the feasibility of 5G for AVs communications.

1.7. Deep Reinforcement Learning

Deep reinforcement learning is a subfield of machine learning and the evolution of reinforcement learning. Differently from other machine learning paradigms, reinforcement learning has no supervisor, something or someone telling what correct action is and what is not, but only a reward signal on which the action will depend; the feedback of the action taken is delayed and not instantaneous; the algorithm is sequential and finally, the action taken by an agent affects the subsequent data it receives. The difference between deep reinforcement learning and reinforcement learning is that the first one learns from a training set and then applies that learning to a new data set, while reinforcement learning is dynamically learning by adjusting actions based on continuous feedback to maximize a reward. DRL utilizes a neural network, which in this thesis work is composed of 16 hidden layers, and can handle high-dimensional problems, which RL algorithms cannot. The configuration on which a DRL is designed can be described as follows:

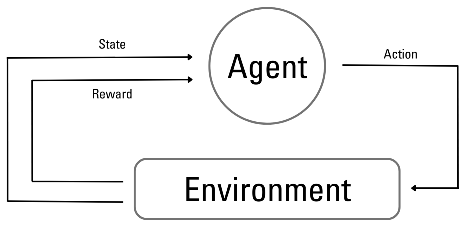


Figure 1.6: Configuration of a generic deep reinforcement learning algorithm [54].

As can be seen in Figure 1.6, the main components of a DRL logic are [43]:

- the agent. At each time step the agent executes an action and receives an observation and a reward from the environment;
- the environment. It receives the action and emits observation and rewards needed for the following time step of the simulation;

The reward is a scalar feedback signal, that indicates how well the agent is doing at a specific time step. During the simulation, the agent will calculate the reward not only for the next action but also the following ones, considering a discount rate γ, which in this case study is equal to 0.99 to address the lower relevance of reward really far from the actual position of the agent. The agent's work is to maximise the cumulative reward. The history is the sequence of observations, actions and rewards of all the time steps of the simulation and has the following structure:

$$H_t = 0_1, R_1, A_1, ..., A_{t-1}, O_t, R_t \tag{1.1}$$

What happens at the next time step depends on the history. Finally, the state is the information used to determine what happens next and it is any function of history. Two

different states can be defined: one linked to the environment, which is generally private and not visible to the agent; the other one is linked to the agent and represents all the information used by deep reinforcement learning algorithms. In this case study, the vehicle state is defined by its position, speed, and longitudinal acceleration and by taking from the environment the information about all the other vehicles in the roundabout.
The major components of a DRL agent may include one or more of the following components [6]:

- the policy. It is the agent's behaviour and it is obtained as a map which links the state to the action made. It can be deterministic or stochastic;
- the value function. It is a prediction of the future reward and it is used to evaluate the goodness/badness of the possible next states and, therefore, to select the best action;
- the model. It is the agent's representation of the environment. Thanks to it the agent predicts how the environment is changed by his actions, getting a prevision of its next state and immediate reward. It is possible to formally define:

$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s^{'} | S_t = s, A_t = a] \tag{1.2}$$

$$R_s{}^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \tag{1.3}$$

The policy is the main component analysed in this thesis work and the one which needs to be optimised to get an AV able to take correct decisions leading to a vehicle capable of driving in the traffic scenario studied considering also the comfort of passengers. The value function is obtained by training the policy with a training data set. The model depends directly on the devices present in the actual AV, such as cameras and sensors. The more they are, the better will be the agent's understanding and representation of the environment. Based on the information provided, the deep reinforcement learning algorithm will initially not know the environment model and state representations, and it will discover them over iterations improving its policy.

1.8. Policy Optimisation

Deep reinforcement learning algorithms can be divided into three main groups:

- model-based;
- value-based;

- policy-based.

The first classification defines the main two families of DRL algorithms: model-based and model-free. In model-based approaches, the algorithm makes use of a predictive model of the simulation environment to understand what will happen consequently to a specific action. Model-free approaches, on the other hand, learn a control policy directly to understand the effects of their actions. Value-based algorithms learn the link between state, action and value to take the correct decision. They are based on the evaluation of a value function for small problems and of a value approximation function for larger problems. Considering the second scenario, many approximators can be used, such as linear combinations, neural networks, decision trees or Fourier bases.

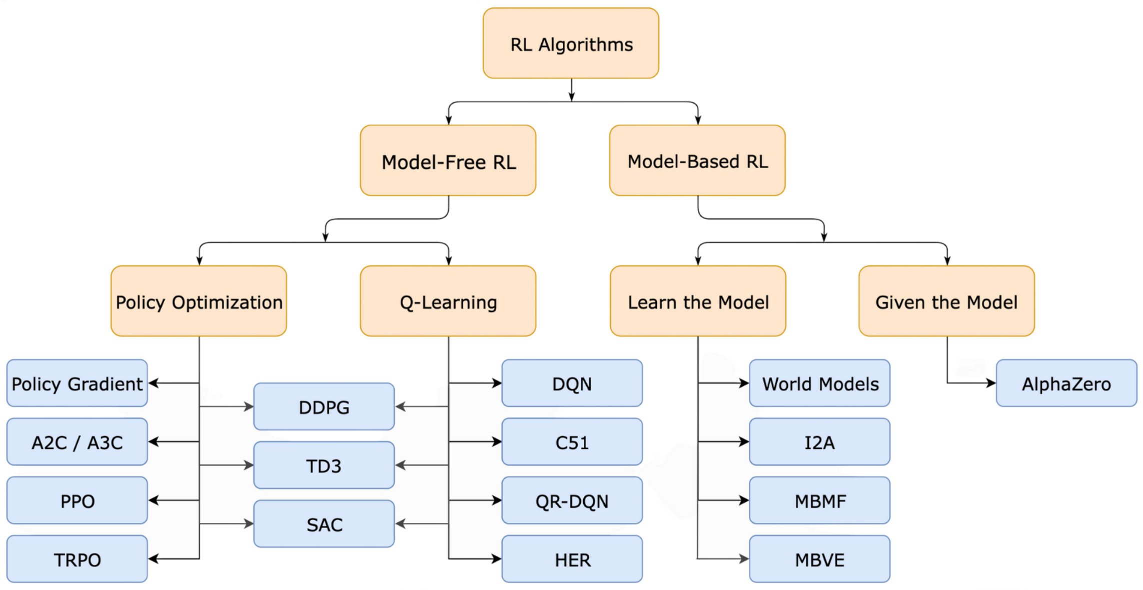


Figure 1.7: Taxonomy of reinforcement learning possible algorithms [58].

In this thesis work, policy-based algorithms have been considered. For these approaches, the policy is directly parametrised and no value function is considered. The advantages of policy-based algorithms are that they have better convergence properties, they are effective in high-dimensional action spaces like the roundabout scenario and they can learn stochastic policies, of which the one of this experiment is an example [53]. The main disadvantages are that they typically converge to a local rather than a global optimum and that evaluating a policy is typically inefficient. Policy-based DRL is an optimisation problem. It can use the gradient to get to better efficiencies like it is adopted in this case of study. Therefore, the gradient is used to find the local maximum by ascending the gradient of the policy with respect to the parameters θ which define it, considering α as the step-size parameter.

$$\Delta\theta = \alpha \nabla_\theta J(\theta) \tag{1.4}$$

Specifically, in this case study, Stochastic Gradient Descent [40] is considered, which is a probabilistic approximation of Gradient Descent, because, at each step, the algorithm calculates the gradient for one observation picked at random, instead of calculating the gradient for the entire dataset. It is for this reason much faster, and more suitable for large-scale datasets. Since the gradient it's not computed for the entire dataset but only for one random point at each iteration, the updates have a higher variance. This makes the cost function fluctuate more on each iteration when compared to Gradient Descent, making it harder for the algorithm to converge. Several solutions to this problem can be considered: [39] describes a method to accelerate Stochastic Gradient Descent using Predictive Variance Reduction, while [42] considers an adaptive step size method. The problem related to the policy gradient method considered is that the reward is obtained at the end of the simulation and, therefore, even if some of the actions of the simulation were not correct, the final reward can be good.

1.9. Simulation environment

This subsection presents the simulation environments and all the tools used to simulate inside the driving simulator of the Politecnico di Milano (DriSmi). A similar simulation environment has been used in [36], using flow to train the AVs policy, SUMO to simulate the roundabout traffic and, finally, a 1:25 scale testbed for connected and automated vehicles in the University of Delaware's Scaled Smart City. It has been demonstrated a reduction of 5% in average travel time and 22% in max-travel time. A simple roundabout model has been considered and no human driver action has been simulated. Using DriSMi simulator, adding the possibility to drive inside the simulation environment, and directly interacting with AVs will be greatly beneficial in terms of the results obtained and the generality of the solution obtained.

1.9.1. SUMO

SUMO (Simulation of Urban MObility) is a microscopic traffic simulator allowing intermodal traffic systems [45]. In this case study it is used as the main simulation environment in which all Autonomous Vehicles are trained and take their decisions all over the simulations. SUMO can be used to create complex simulation environments with many types of vehicles, different lanes and different kinds of intersections (regulated, non-regulated, with traffic lights). To be able to run a generic simulation three files are needed: the simulation network containing all edges, lanes and trajectories that any vehicle can follow; the route file, in which all vehicles' routes are described; the configuration file which commands the simulation and links all the other files together. SUMO represents every vehicle as a single point, for which it is possible to have many different details at every time step such as position, speed or fuel consumption. Vehicles move along trajectories, which the user can customise, defined as a list of points linked together using straight lines. SUMO has been used to accomplish the MTS of the whole simulation.

TraCI

TraCI (Traffic Control Interface) is a SUMO library giving access to a running road traffic simulation. It allows retrieval of any data about the simulation status and all the vehicles' properties. It uses a TCP (Transmission Control Protocol) based client-server architecture. TraCI allows one to choose the deterministic number of clients, who will connect to the simulation to retrieve data or to change any simulation property. It is important to consider the speed of this communication protocol to assess its performance. The amount of slowdowns depends on many factors, of which the most relevant is the number of TraCI functions called in every simulation step. In this case study.

1.9.2. Flow

Flow is another library which enables the creation of an interface between the SUMO simulation and all Deep Reinforcement Learning libraries (i.e. RLlib) used to run the Autonomous Vehicles in the case study [56]. Flow takes as an input the simulation network file. It can define any new vehicle inside the simulation, specifying its route, initial speed, starting point and car following model used (e.g. in this case study IDM is used). Inflows can be used to create flows of vehicles with the same characteristics. Inflows can be inserted inside the simulation following a probabilistic law or a deterministic one, based on vehicles per hour logic. Flow is also used to start and end the simulation and to define its main properties, such as delay time, time step or rendering properties. In the figure below, a representation of the Flow workflow during the simulation is presented, with a focus on the overall logic.

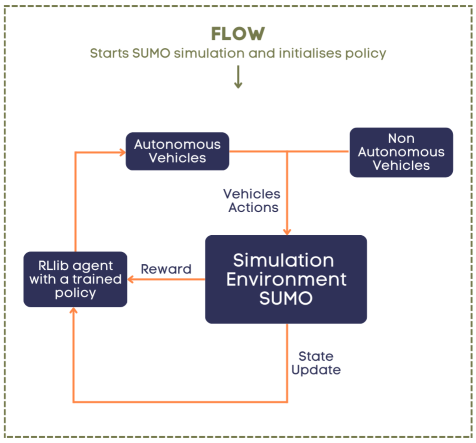

FLOW
Starts SUMO simulation and initialises policy
Autonomous Vehicles
Non Autonomous Vehicles
Vehicles Actions
RLlib agent with a trained policy
Reward
Simulation Environment SUMO
State Update

Figure 1.8: Complete scheme of the simulation environment.

A final note: unfortunately, Flow is no more in a development phase and all its packages are not up to date. This may provoke some compatibility issues with SUMO or any other library used in the process.

1.9.3. DriveSim

DriveSim loads all the files needed such as the simulation environment and all the others containing the vehicles which will be displayed and the initial conditions for the Ego car (i.e. the car driven by the human in the driving simulator) inside the simulation. DriveSim is the main simulation component referred to the DS. It solves the vehicle dynamic model and accounts for all driver feelings inside the simulation environment. It is the most relevant component and the one enabling the obtaining of quality results, replicable and usable for further studies outside the simulation.

1.10. Simulation communication protocols

A final important component of the simulation environment is represented by the communication protocols used to exchange data between all the tools used and precedently described. There are two communication protocols used, between which it is possible to consult a comparison in [3]. The first one is the ethernet/IP (industrial protocol) protocol, used to link the SUMO simulation with the DriveSim one. The transfer of basic I/O data happens via User Datagram Protocol (UDP) implicit messaging. This is one of the most adopted communication protocols to send messages in an IP network. UDP provides

checksums, small-sized blocks of data to delete errors, and port numbers used to compose the necessary datagram sockets, which are unique and linked to precise I/O exchange data. There is no handshaking between the two services. Therefore, no authentication or coordination is given or required by the guest machines and the hypervisor, respectively. This is not a problem for the application considered, since all the communication happens on the local network and the information sent and then received is considered always secure and correct. For the same reason, this Protocol happens to be faster and more convenient for this thesis work.
The second protocol is Transmission Control Protocol (TCP). It is used in this study both to ask SUMO for all vehicle data, such as position, speed or direction and for updating in SUMO the ego car position moved by the human driver in DriveSim. This is one of the most used protocols for orderly fashion data communication and one of the most common uses is for computer-to-computer file transfer. TCP allows communication between several computers, called hosts, connected to a local network with other computers, called clients. It offers hardware independence, which is a good property for this case study. Initially, the server must be passively listening and waiting for all expected clients to be connected. In this case, a three-way handshake, re-transmission and error detection is considered. This, on the other hand, lengthens communication latency, making this protocol the bottleneck of the systems in terms of communication time efficiency. For this reason, as previously described in section1.9, this part of the communication must be optimised to reduce the total time needed for each time step and respect the limit imposed by the designer of the simulation.

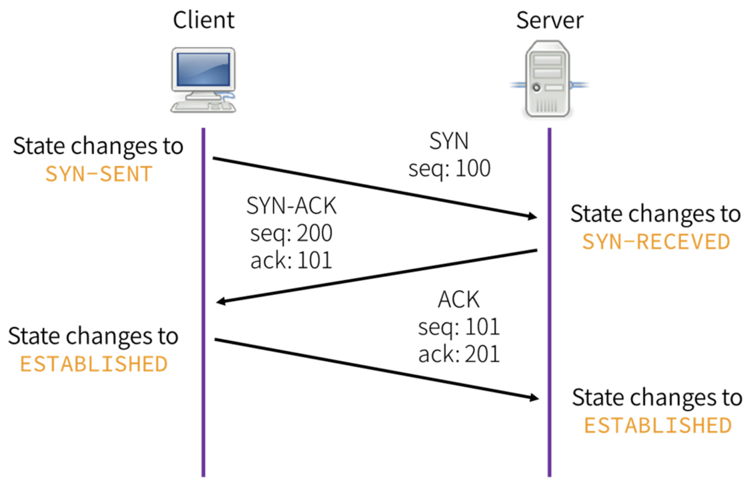


Figure 1.9: Communication steps for a generic TCP [31].

Looking at figure 1.9, it is possible to see the general structure of TCP communication.

The two main actors are the initiator and the receiver. While the communication is active, a segment is sent between the two actors. This segment has a specific structure and it is divided into the header and the data section. Inside the data section, 8 flags are used to control communication. Among these values appear:

- ACK, which indicates that the Acknowledgment is relevant. It confirms the receipt of data from the initiator;
- FIN, used to pause the communication from the side of the sender. When both of the actors send a FIN equal to 1, the communication is considered closed;

1.11. Goal statement

This thesis work is part of a wider and more complex project called *AI@EDGE*. This project aims at the realisation of a connected-compute fabric for creating resilient and secure end-to-end slices, which can be used for a diverse range of AI-enabled applications. This connect-compute platform is part of a wider system architecture within which is also located a second layer: the network and service automation platform. This platform automates the management of different Orchestrators, provides non-real-time intelligence and ensures the latency needed for the AIF, which in this case study represents the RL policy, to reproduce reality. The connect-compute platform contains the virtual infrastructure to run the mobile edge computing applications, used to virtualise reality [1], and it is the main component which the project aims to optimise for different types of applications, among which the automated and connected mobility. Figure 1.10 presents a complete representation of the system architecture.

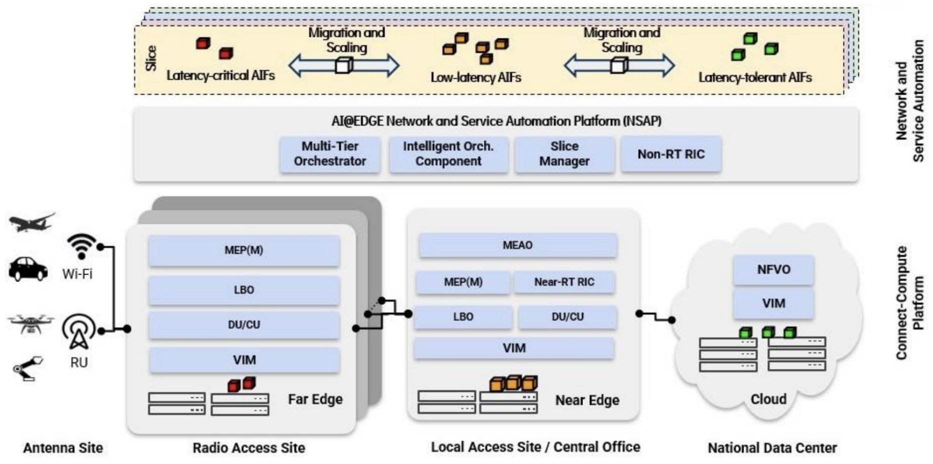


Figure 1.10: AI@EDGE system architecture [33].

The *AI@EDGE* project will focus on six main themes:

1. Artificial Intelligence and Machine Learning for closed-loop automation;
2. ML for multi-stakeholders environments;
3. connected-compute platforms;
4. provisioning of AI-enabled applications;
5. serverless platforms for AI/ML;
6. cross-layer radio access.

The four use cases below presented have been chosen to validate the platform created. They cannot be satisfied by current 5G networks according to the 3GPP standards cited in section 1.4.3.

- UC1: virtual validation of vehicle cooperative perception. This project will show how edge technology can be implemented to orchestrate a complete and complex network by implementing and testing the Digital Twinning of a mix of real and emulated vehicles. Considering the connection between vehicles, also the network-level data exchange will be studied in order to build a cooperative perception between emulated vehicles and human-driven vehicles.
- UC2: secure and resilient orchestration of large (I)IoT networks. This second use case aims to the development of AI for network security considering intrusion detection approaches.
- UC3: edge AI-assisted monitoring of linear infrastructures using drones in BVLOS operation.
- UC4: smart content & data curation for in-flight entertainment services, whose goal is to develop an edge cloud infrastructure on-board aircraft creating 5G connectivity inside the aircraft cabin.

Specifically, this thesis work is part of the first use case and is also closely related to the CCAM objectives and actions. The relevance of this case study is represented by the availability of DriSMi, a powerful and technologically advanced driving simulator, thanks to which it is possible to develop and study different scenarios and AV configurations. This project is made possible by the cooperation between different research institutions such as Politecnico di Milano, Fondazione Bruno Kessler and other companies, among which Telecom and various stakeholders. To obtain a complex and similar to the reality simulation environment, it is needed to study the system both from a microscopic and a macroscopic point of view: the vehicle must be accounted as composed by all its mechanical components to return to the AV policy correct data about its behaviour as discussed in section 4.2; the AVs must be considered also as a part of a more extensive system, in which appear many different actors. The policy trained by FBK and delivered capable of responding to behavioural demands must be studied and compared with:

- different types of network, getting a detailed description of its adaptability in terms of geometry into which it is inserted;
- different types of drivers. AV must be able to interact with multiple types of drivers as in real life. This can be tested by changing the driver model inside SUMO.

The fundamental goal of the work is to create and modify the simulation environment with the aim of obtaining a faithful representation of reality. All software involved must be linked and the information provided needs to be compliant with the specific software requirements. An example of that can be seen in section 4.2 or in section 1.9. The ego car and the simulation will act as a Digital Twin of the real vehicle and of the real scenario. The Digital Twin has gained more relevance in recent years due to technological development and it is able to reduce drastically prototyping errors, costs and environmental impact. Therefore, the goodness of the model and the simulation environment is critical considering the project in its entirety and its future phases, even beyond this thesis work. Moreover, this thesis aims also at demonstrating the effectiveness of the AV policy in terms of comfort and safety for passengers by considering a *Replay* scenario. The simulation environment will be finally tested by many users to understand its performances and future developments of this technology.

1.12. Thesis structure

This thesis work is organised as follows:

1. Chapter 2 is dedicated to the creation of the preliminary simulation environment. It will be useful to acquire relevant knowledge in the environment design phase, to comprehend the effect of all design components and to carry out the preliminary tests. Particular attention will be posed to the communication implementation between all software involved in the process. Both the communication setup and its levels' realisation will be deeply considered;
2. Chapter 3 will describe the preliminary tests, the questionnaire used to obtain the preliminary results and, finally, the modifications implemented in the final simulation environment to obtain a valid Digital Twin of the real environment;

3. Chapter 4 is devoted to the achievement of the *Replay* scenario. It will include the specific solution adopted and all modifications applied to the vehicle's mechanical model to obtain the perfect replication of the AV movement within the SUMO simulation;

4. Chapter 5 will, ultimately, describe the final simulation environment. All modifications proposed in Chapter 3 will be considered and their specific implementation will be described. Furthermore, the final chapter will present the real traffic calibration inside the SUMO network.

The structure was chosen considering a path to build the most complete and accurate simulation environment possible, having to consider the importance of such a model and the extensive use of it beyond this thesis during the AI@EDGE project.

2 | Preliminary simulation environment

In this chapter will be described the preliminary simulation environment used to obtain the first results and use them to get the final simulation environment. Initially, the complete project test bed will be depicted, so as to make clear the overall complexity of the project and a more general configuration of it. After that, the simulation environment development and its characteristics will be defined. A detailed communication of all the tools used to run the simulation will be presented and, finally, the results of the preliminary experimental tests will be discussed, highlighting criticalities and modifications adopted for the final simulation environment.

2.1. Test bed

The test bed presents all the main components of the project architecture. This thesis work is, in fact, embedded in a larger project involving many partners. The final simulation structure is composed of all devices needed to ensure the aim of the project. It can be divided into two main components:

1. the real-time environment;
2. the structure involved in introducing communication latency.

These two parts will be described separately and, finally, its link will be presented.

Real-time environment

The task of the real-time environment is to ensure communication between three main devices: the real-time database, the driving simulator and SUMO. This must happen as quickly as possible since in reality will happen inside the same vehicle.

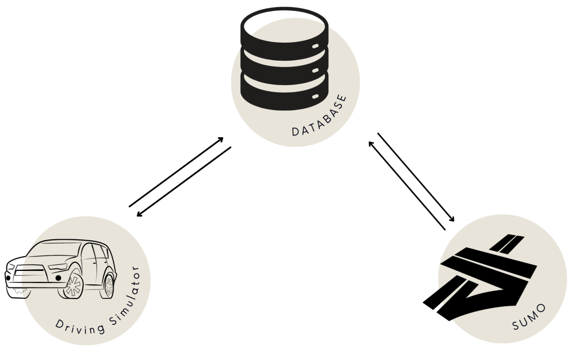


Figure 2.1: Three main components of the real-time communication of the test bed.

The real-time database (RtDb) is the centre of the communication and it is linked both to SUMO and the driving simulator. The two communications happen on both levels, with the actors sending and asking for information. Firstly, it is described the communications happening within this database, which runs inside the concurrent real-time computer.

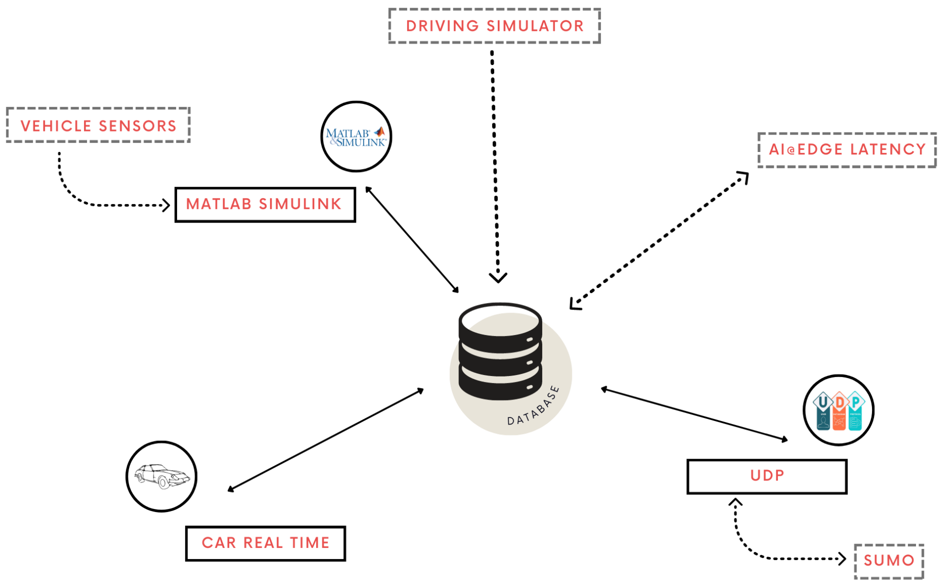


Figure 2.2: Internal communication of the concurrent. Dotted lines represent the communication between the concurrent and other external components.

The database gives Car Real Time the driver's action obtained by the driving simulator. The ego car actions, such as steering, accelerating or decelerating, are forwarded to car real time, which returns the vehicle dynamics data. Additionally, the simulation's traffic

information is updated using SUMO data thanks to a UDP communication protocol. All this data (i.g. traffic data, ego car dynamics and instructions to be delivered to the driver) are exchanged with Matlab Simulink, which obtains information also from the vehicle and infrastructure sensors. Finally, all the information regarding the ego vehicle is exchanged with the portion of the structure devoted to communication latency.
The second section is the driving simulator one. It is mainly responsible for anything the driver feels or sees during the simulation. A detailed description of how the simulator works is presented in section 3.1.

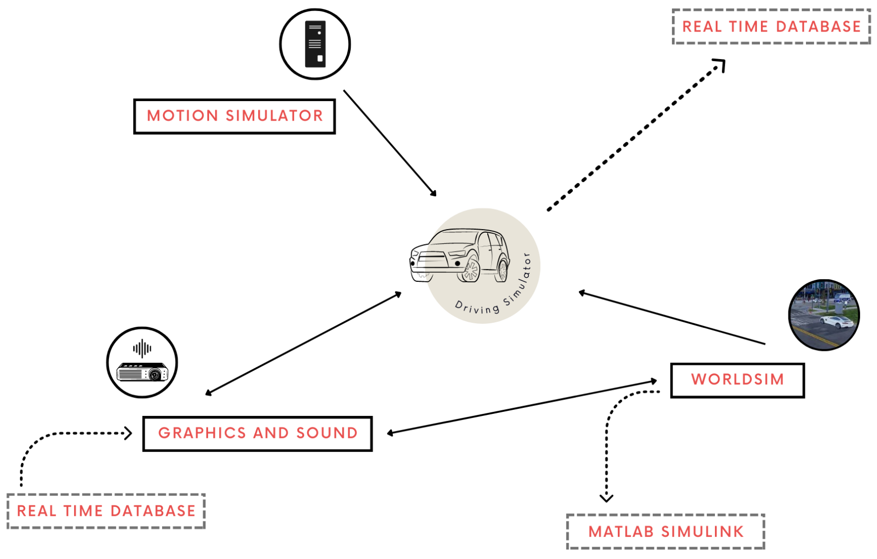


Figure 2.3: Internal communication of the driving simulator. Dotted lines represent the communication between the driving simulator and other external components.

The driving simulator acts mostly as a receiver. It gets the visual and audio information which is obtained considering both the environmental data coming from Worldsim and signals which must be sent directly to the driver, derived from all the simulation data exchanged with Matlab Simulink. It also receives the motion information needed to move correctly all the actuators. The driving simulator is one of the most crucial components since on it depends everything experienced by the driver. As will be described in the following sections, data obtained from the database could be modified with respect to the source, to have a final experience as close as possible to reality. Furthermore, DriSMi is the component which elevates the project and lets it be a unique simulation environment. Thanks to this powerful and realistic driving simulator, it will be possible to bridge the

gap between simulation and reality and, finally, succeed in obtaining crucial data for this field of study. This is also why this project is a fundamental resource for CCAM.
The third and last section of the real environment is the Linux workstation which runs SUMO.

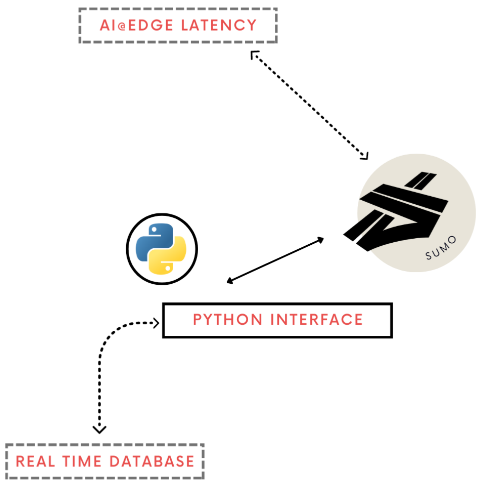


Figure 2.4: Internal communication of the linux workstation. Dotted lines represent the communication between the Linux workstation and other external components.

This is the section that this thesis work focuses on together with the one devoted to policy management. Inside SUMO runs the principal microscopic simulation. The information generated in SUMO is given to python, which modifies it if necessary and sends it through UDP to the real-time database. Python is also used to acquire ego car data and send it to SUMO to update the ego vehicle position. SUMO traffic data is also shared with the section devoted to the implementation of the system latency and, specifically, to the server on which runs the policy.

AI@EDGE latency

The second main component is the one devoted to latency. It is needed to introduce a systemic delay which the policy must consider to have a simulation closer to reality. It is composed of two separate sections:

1. the edge server;
2. the 5G transmission system.

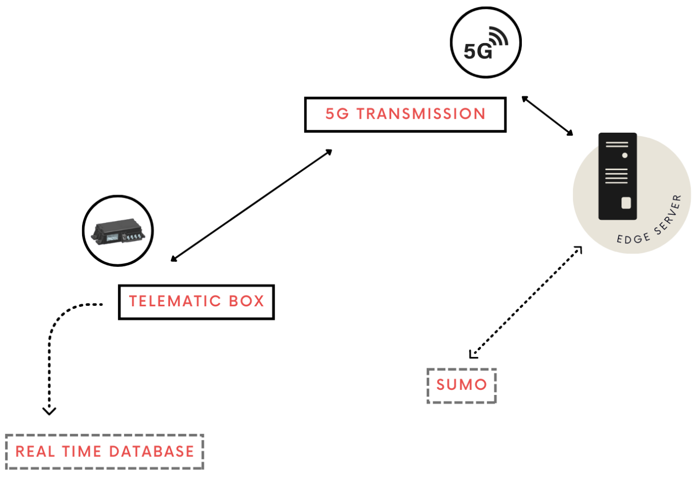


Figure 2.5: Communication inside the AI@EDGE latency section.

The edge server exchanges data with the simulation running in SUMO. Inside the edge server, this information is transferred to the Artificial Intelligence Functions (AIF) using an Advanced Message Queuing Protocol (AMQP). This communication protocol is used for message-oriented middleware, which is the software and hardware used to send and receive messages between different systems. The two main components of this communication protocol are the broker and the client. Inside the edge server is present the AMQP broker. This broker receives messages from publishers, which in this scenario are SUMO and the 5G transmission system, and routes them to consumers, represented by the AIF. The 5G system is linked to the telematic box inside the vehicle with which it exchanges data on the ego vehicle and about the information required by the driver. The telematic box acts as the AMQP client interacting with the AMQP broker through the 5G transmission system. It receives all data through CAN from the real-time database. The policy, therefore, receives information about the environment, both ego vehicle and traffic, with a systematic delay. This is done to simulate the real scenario, in which sensors would give to the AIF running the vehicle data with a delay dependent on the specific sensor considered.
All the components in this section are managed by third parties, like Fondazione Bruno Kessler (FBK) which takes care of the policy training.

2.1.1. Test bed summary outline

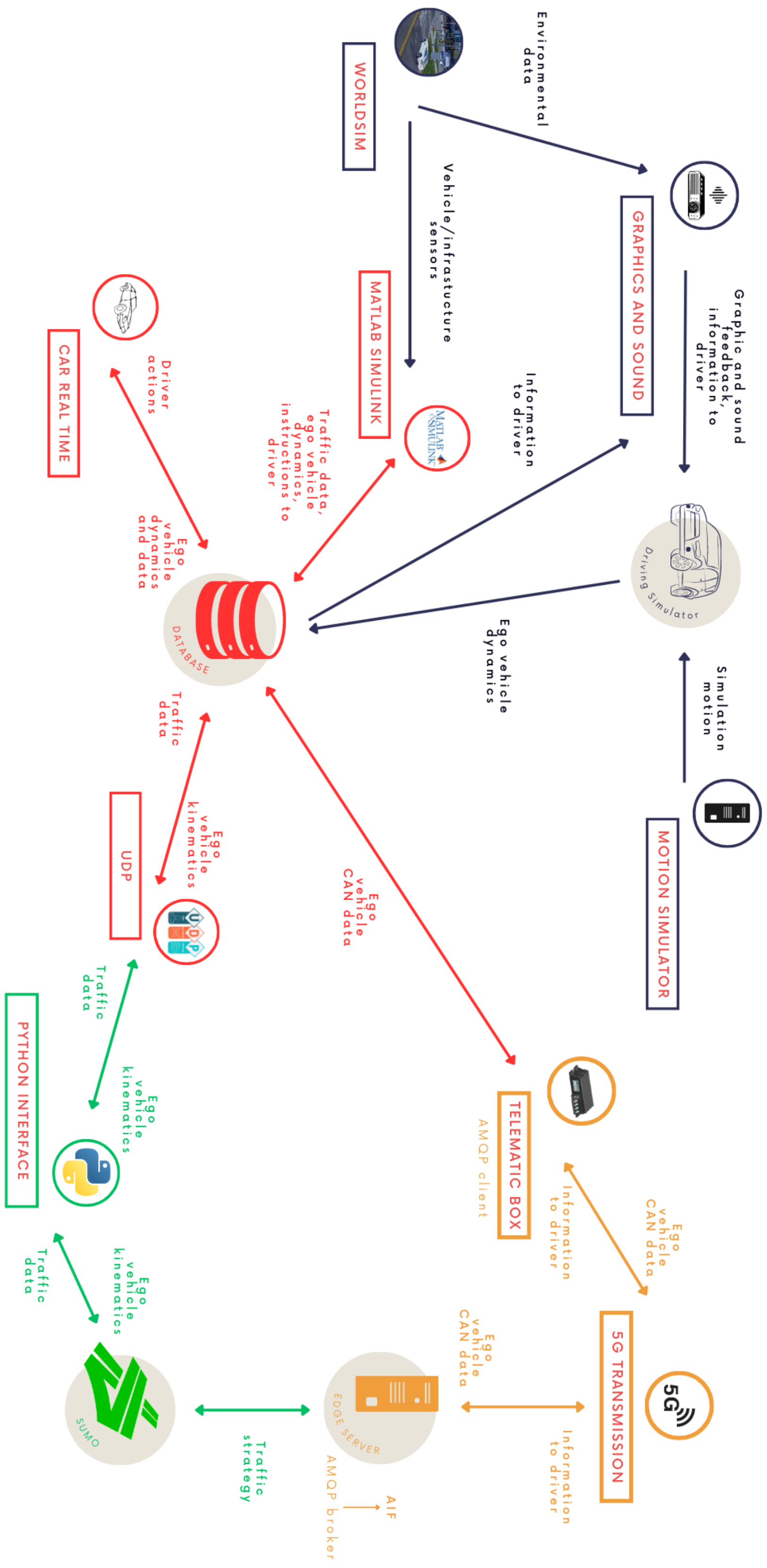


Figure 2.6: Communication summary outline.

2.2. SUMO simulation environment

SUMO is used as the main simulation environment both to train and test the policy used for the AVs. The first design phase coincides with the creation of the roundabout inside this micro-mobility simulation environment. Inside SUMO many details can be defined in terms of vehicles, lanes and geometries, making it usable to create really complex scenarios. There are two approaches to building a generic simulation scenario:

1. using Netedit. It is the creation environment, which SUMO makes available to users. It is a graphical tool in which can be defined edges, lanes, vehicle flows and types. It is simpler and usable for any user, even with no experience;

2. directly coding the network. It is possible to use a specially created computer language. It is more difficult to understand and use, but it is more powerful and it lets the user easily personalise all the design instances.

The main design component is represented by the junctions. They can be linked together creating edges and are the main reference to which all the other components refer to. Inside edges, lanes can be obtained and defined in terms of number or hierarchy.

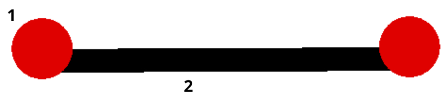


Figure 2.7: Basic design components in SUMO. (1) is the junction, (2) is the edge.

From these two essential design components, it is possible to create really complex networks, such as roundabouts or also complete cities' geometries. To get the final network it is possible both to design it from the beginning in Netedit or use the computer language and use OSMWebWizard, which is based on OpenStreetMap and lets the user choose a scenario from reality and directly import it inside SUMO. Obviously, modifications should be made to get the perfect replication of reality, but most of the work is generally done.

2.2.1. Edges

All network edges are described inside a .typ.xml file extension. Inside this file, all roads and their types are listed. Some of the main characteristics which can be defined are:

- ID. The id is a string variable and defines the name of the edge considered. This is the only mandatory attribute for an edge;
- allow, which is a string defining the allowed vehicle classes on the specific edge, such as pedestrians, bicycles, busses, cars and so on;
- disallow, which is the same as allow, but refers to the vehicle classes not allowed on the edge;
- discard, which is a boolean variable defining if the edge type needs or not to be imported into the network. It defaults to false;
- numLanes. This is an integer and explicates the number of lanes per direction on the edge. Lanes are one of the most important components of the edge and, more in general, of the simulation. The specific vehicle trajectory will depend on the shape of the lanes;
- oneway,a boolean variable making an edge one-way direction;
- priority, an integer defining the priority between different lanes;
- speed, representing the speed limit on the edge

Once a specific type of edge is defined, it can be used as a reference for all other edges of the same type, making the development phase really shorter.

2.2.2. Routes

Routes express the path that a specific vehicle or a flow of vehicles will follow during the simulation. When defining a route, the user must consider the edges the vehicles will drive along, the number of times that these edges will be repeated or their colours to better visualise them. Moreover, some details must be considered:

- routes must be connected. The edges of a route must be subsequent, otherwise, the simulation will rise an error. The edges chosen must be available for the specific vehicle class;
- routes must contain at least one edge;

- the route files, containing all the routes inside the network, must be sorted by starting times;

It is possible to define also only the beginning and ending edges of the route. In this case, the simulation will perform the fastest-path routing based also on traffic conditions. Finally, the user can also define traffic assignment zones (TAZ) from which the route can begin and to which it can end.

2.2.3. Vehicles

Vehicles are defined inside the .rou.xml extension file. It is possible to create single vehicles types and instances and also complete traffic demands. The vehicle definition in SUMO consists of three main components:

1. a vehicle type describing its physical properties such as length, maximum speed or maximum longitudinal acceleration;
2. a route that the vehicle will follow
3. the specific vehicle itself

Vehicle types and routes can be shared by many vehicles in the network. On the other hand, the specific vehicle is unique.

```
<routes>
    <vType id="type1" accel="0.8" decel="4.5" sigma="0.5" length="5"
       maxSpeed="70"/>

    <vehicle id="0" type="type1" depart="0" color="1,0,0">
      <route edges="beg middle end rend"/>
    </vehicle>

</routes>
```

The code above shows an example of a vehicle definition. A type1 vehicle is created, defining its route and characteristics. In this case, the route is defined inside the specific vehicle. It is also possible to define a specific route, which will then be shared by many types of vehicles, getting a final complete flow.
Also for vehicles, it is possible to define many attributes. Only the ones used in this thesis work are defined below:

- departLane, a string defining the lane on which the vehicle will be positioned initially;
- departSpeed, the initial speed of the vehicle;
- type, which in this case is used to differentiate AVs from humanly driven ones.

Inside the preliminary simulation environment flows are used. They let the user define a flow of vehicles all equal following the same route. It is possible to define properties such as vehicles per hour or probability to address the method by which vehicles are positioned inside the network during the simulation. More in detail, using vehicles per hour, it is possible to define explicitly the number of vehicles per unit of time to be shown; it is also possible to define the probability of emitting a vehicle each second.

2.2.4. Preliminary network

Below is shown and described the final preliminary network.

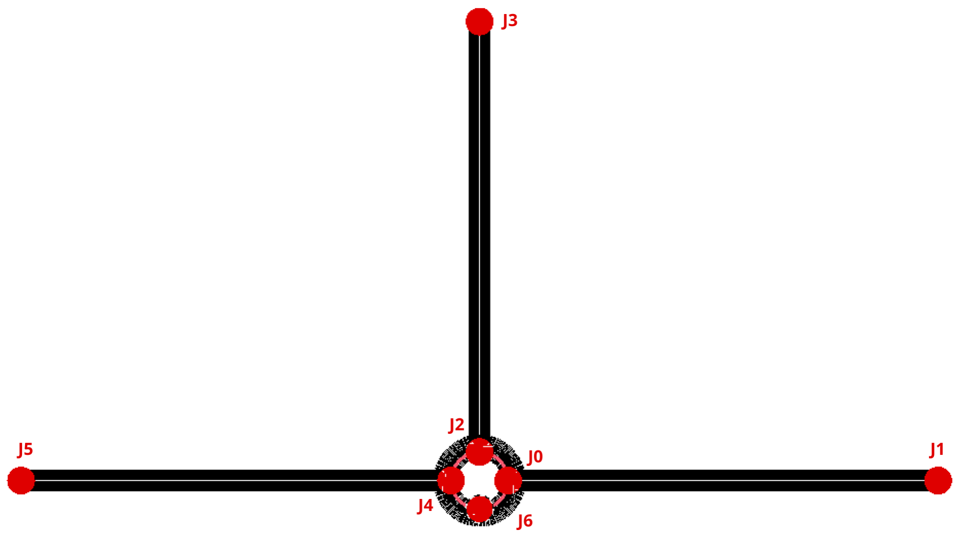


Figure 2.8: Overview of the preliminary network.

It is composed of 7 junctions, 6 of which are directly connected one to another. All of the edges outside the roundabout have 2 lanes, one per direction. The roundabout obtained is a mini-roundabout with three legs. The internal radius of the roundabout is 9 m and the external one is 15 m. More in-depth, considering each internal edge:

- -*E2* and *E2*, edges from J5 to J4 and from J4 to J5, respectively;
- -*E0* and *E0*, edges from J1 to J0 and from J0 to J1, respectively;

- *-E1* and *E1*, edges from J3 to J2 and from J2 to J3, respectively.

Inside the roundabout, 4 edges can be defined linking the 4 main junctions, called J0, J2, J4 and J6. More in detail:

- *E10*, from J2 to J4;
- *E11*, from J4 to J6;
- *E8*, from J6 to J0;
- *E9*, from J0 to J2.

On each of these edges, there's only one lane. Inside the roundabout, all junctions have three lanes, except for J6.

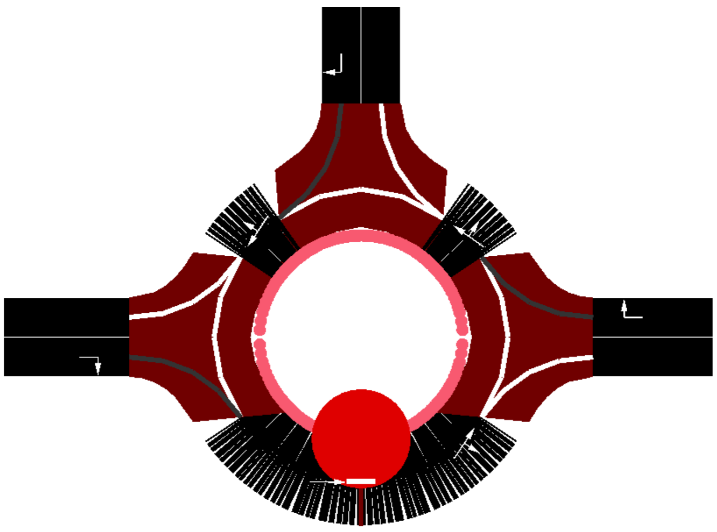


Figure 2.9: Details of the circulatory roadway.

In Figure2.9 it is possible to see all 4 internal junctions in detail and all the lanes inside the circulatory roadway. SUMO defines a lane as a list of points connected by straight lines. This must be considered since it directly influences also the characteristic vehicle angles, such as the yaw angle. Junctions' shapes can be personalised also to have a good visual result, but they do not affect the vehicle's behaviour. Looking at the four junctions, J6 is the simplest one with just one lane. Considering the other three, lanes must be defined considering that ending and starting points must be coherent with the same points for the subsequent lanes into the other junctions. Initially, it is defined a flow of 40 vehicles equally distributed along the three possible incoming directions. This number considers the time needed by the TCP protocol to exchange all vehicle data between DriveSim and SUMO at every step of the simulation, as described in section 2.3.2. Moreover, vehicles

can be of two different types: human-driven and AI-driven.
Considering all the edges of the network and dividing them based on the maximum allowed speed, it is possible to define:

- external edges. The edges outside the roundabout. The maximum speed is set to 50 km/h, considering Italian regulations;
- internal edges. The edges inside the roundabout, for which the maximum speed is set to 40 km/h.

For this preliminary simulation environment, no study has been made on the effect of the internal edges' maximum speed.
This network configuration has been used for all preliminary tests to set all the requirements in terms of AVs policy, communication protocols and human feedback as illustrated in the following sections.

2.3. Communication setup

In this second section, the simulation is deeply described from the point of view of the communication between all the main devices. These devices are presented down below. This description has been built considering the three main phases of the simulation:

1. initialization;
2. a generic time step;
3. the end of the simulation.

For all three phases, the actions of all the devices are separately described to have easier access to information and to be able to track efficiently what just one of the three does during the simulation. At the end of the section, it is possible to see a summary outline of all the communication phases.

Terminal 1 with Flow

This first terminal uses flow to link SUMO and RLlib. It creates the simulation and owns it till the end of the experiment. Inside this terminal, it is possible to find both SUMO commands, which are used to create vehicles, edges, and routes or to create the simulation and RLlib commands, which let the policy work correctly all over the simulation.

Terminal 2 with Traci

The second terminal is the one completely devoted to communication. It is a mediator between SUMO and DriveSim. It contains many different functions, which are needed to correctly read and send data between the two different traffic simulators.

DriveSim

DriveSim is used to get the final and used simulation. It asks for many different data and it also sends the Ego car details to the second terminal. The Ego car is the one driven by the real person inside the DriSMi simulator.

2.3.1. Initialising the simulation

Before starting the actual simulation all the devices need to carry out preliminary activities. These activities are needed to align all devices and let them start correctly and together once the user is ready inside the simulator.

Terminal 1

Inflows are used to create all vehicle flows during the simulation. Two different types of vehicles are created:

- Human-driven vehicles;
- AI-driven vehicles that use the policy to take decisions.

To have the correct balance between these two types of vehicles, it is needed to define the number of vehicles per hour for both the two categories and for all three legs of the roundabout. This way, vehicles are defined deterministically and it ensures the replicability of the tests conducted. Two different scenarios are considered: the first one with 80% of AVs and the second one with 20% of AVs.

Terminal 2

First of all, it is necessary to list all ids of the vehicles which will run in the simulation. One must define, for every vehicle that is needed to be seen, a unique port and some characteristics such as starting angle – which depends on the initial position – to be able to render those vehicles in DriveSim. Since in SUMO trajectories are just straight lines one after another and considered that it is needed to precisely get the yaw angle at every time step, it is also important to interpolate all the trajectories inside the roundabout.

At the beginning of the code, for all of them, some relevant values are expressed to get a correct interpolation during the simulation. Finally, some variables (e.g. Ego car initial position, yaw angles) are initialized.

DriveSim

DriveSim loads all the files needed such as the simulation environment and all the others containing the vehicles which will be displayed and the initial conditions for the Ego car inside the simulation.

2.3.2. Simulation time step

Once all the preliminary actions have been carried out, it is possible to link the three devices and start the simulation.

Terminal 1

As the simulation has been created as shown in chapter 2.1., the first terminal does not process many other commands. It just inserts vehicles in the simulation as requested by the user and enables the policy to communicate correctly with SUMO. About the policy, at every time step, a generic vehicle state is defined considering position, speed, acceleration and data about the other AVs in the simulation. These values are given to RLlib by SUMO through Flow.

Terminal 2

The second terminal is responsible for most of the actions taken while simulating. First of all, the following Ego car data are obtained by DriveSim:

- Cartesian coordinates, x and y;
- Steering angle;
- Speed [m/s].

The cartesian coordinates are first converted, considering the offset between the two traffic simulators. All the variables are then sent to SUMO to move the Ego car to its updated position. Once all vehicles are moved (the non-AVs will be moved by SUMO, accordingly to its logic; the AVs will use the policy to choose their action; the Ego car is moved based on the data received by DriveSim) TraCI requests the following data for all the vehicles active in the simulation:

- Cartesian coordinates, x and y;
- Steering angle;
- Speed [m/s];
- Road ID.

The only difference between what DriveSim gives and TraCI requests to SUMO is the Road ID. This is needed to correctly interpolate the trajectory of the vehicles and to finally transmit the correct yaw angle for every vehicle to DriveSim. At this point, a single step of the simulation is completed and the command "traci.simulationstep()" is called.

DriveSim

DriveSim commands the start of the simulation. As soon as the command "start" is sent by the simulator, TraCI lets the SUMO traffic simulation begin. During the simulation steps, DriveSim sends the Ego car's updated position, speed and steering angle to Terminal 2. It also receives all the data of all the active vehicles and moves them inside its simulation environment. Currently, DriveSim can open ports up to a maximum of 39 vehicles, Ego-car excluded. For this reason, once all of these ports are taken, no more vehicles can be rendered in DriveSim. This problem does not affect the case study considered since the travel time is usually between 35 seconds and 60 seconds and no more than 40 vehicles are active in this time interval. Furthermore, the simulation can still be carried out (without seeing any vehicle above the 40th one) and the maximum number of ports can be extended in the future.

2.3.3. End of simulation

After all the simulation steps needed, the simulation can be finally ended.

Terminal 1

Once Flow receives the command to end the simulation from Terminal 2, it closes everything and saves the data acquired by the policy used during the experiment to be able in the future to load it or to use them to train further the policy.

Terminal 2

Inside this terminal, the simulation loop is obtained through a while loop, which generally can be closed for 3 reasons:

- the number of active vehicles in the simulation becomes 0;
- the number of cycles exceeds a limit set by the user;
- the communication between Terminal 2 and DriveSim is closed.

In this case study, the first condition can never be reached since vehicles are inserted in a simulation on a veh/hour basis with no end. The second condition is generally avoided since logically it must be DriveSim to command the simulation and decide when to stop (third condition) as it decided when to start. Flow needs to know in advance the total number of seconds for the simulation and, for this reason, it is set at a high value – higher than the actual maximum possible simulation time. Once DriveSim sends the “stop” command to Terminal 2, the latter closes the while loop and sends the same command to Terminal 1. Finally, some plots can be obtained considering any of the vehicles listed in the simulation and any of the properties exchanged between the three devices.

DriveSim

The user can stop the simulation whenever they want. It is just needed to send the “stop” command to Terminal 2 and everything will be ended.

2.3.4. Communication summary outline

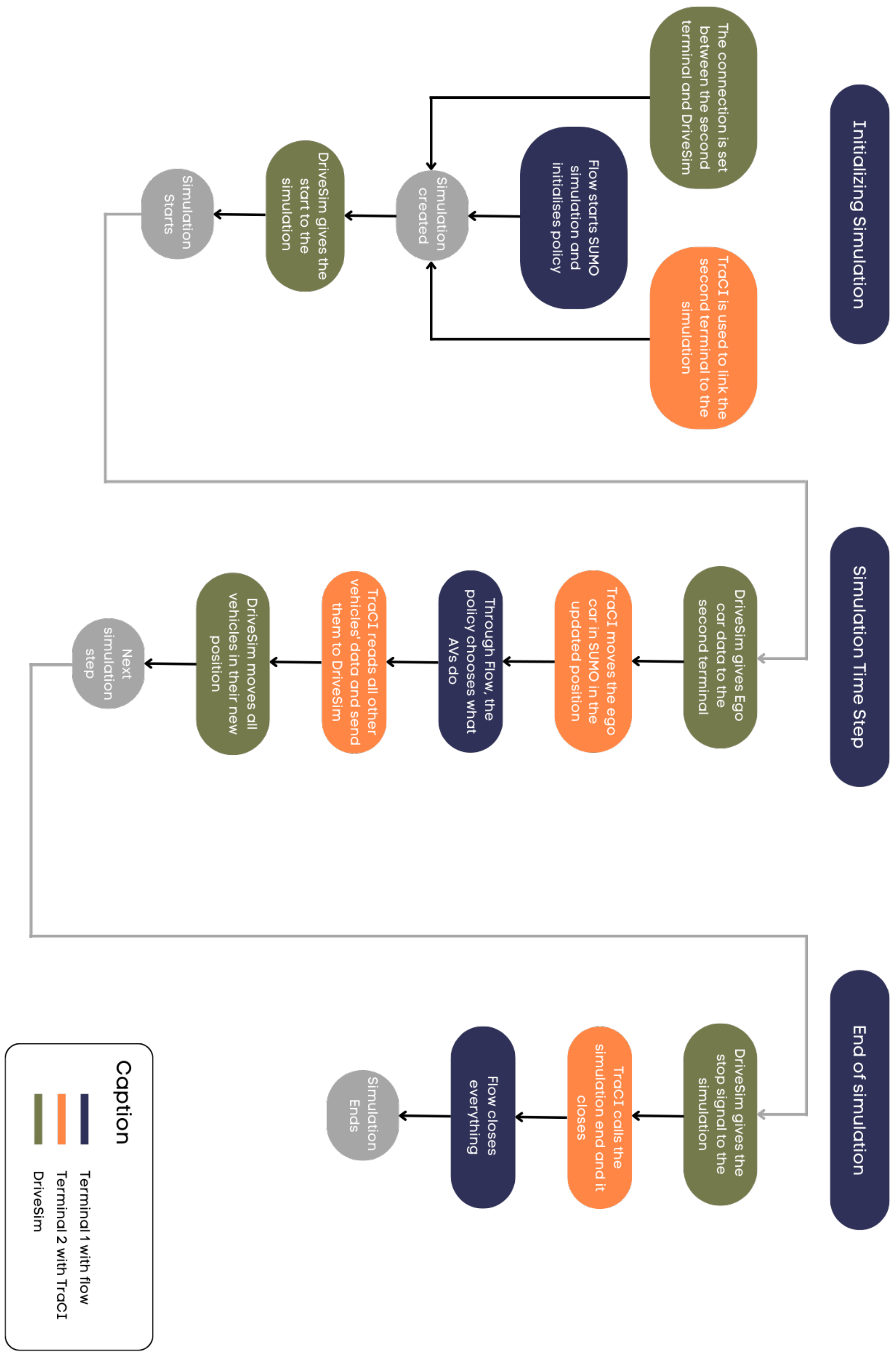

Initializing Simulation
The connection is set between the second terminal and DriveSim
Flow starts SUMO simulation and initialises policy
TraCI is used to link the second terminal to the simulation
Simulation created
DriveSim gives the start to the simulation
Simulation Starts
Simulation Time Step
DriveSim gives Ego car data to the second terminal
TraCI moves the ego car in SUMO in the updated position
Through Flow, the policy chooses what AVs do
TraCI reads all other vehicles' data and send them to DriveSim
DriveSim moves all vehicles in their new position
Next simulation step
End of simulation
DriveSim gives the stop signal to the simulation
TraCI calls the simulation end and it closes
Flow closes everything
Simulation Ends
Caption
Terminal 1 with flow
Terminal 2 with TraCI
DriveSim

Figure 2.10: Communication summary outline.

2.4. Communcation Implementation

This section is devoted to the communication implementation. Both terminal 1 and terminal 2 codes will be analysed, considering their main and most important parts. Therefore, two communication levels will be considered:

1. first level communication, considering Flow and SUMO;
2. second level communication between SUMO and DriveSim.

Thanks to this description it will be possible to have a complete understanding of how the simulation environment works. This is crucial since on it depends all the simulation characteristics and many of its criticalities, that will be considered to get the final simulation. By considering terminal 1, it will be possible to comprehend how the policy is trained and all the parameters used to get to the converged final optimised policy. The analysis of terminal 2 will be relevant to understand how all simulation devices are connected and all data are shared.

2.4.1. First level communication and policy optimisation details

The first communication implemented to get the simulation run is the one between Flow and SUMO. As described in section 1.9, Flow is devoted to policy training, while SUMO is the central device to run the simulation. Inside terminal 1 all vehicles flows are defined and the information is provided to SUMO, which independently updates its processing files.

```
vehicles.add('human',acceleration_controller=(IDMController,{}),num_vehicles=11)
vehicles.add('ai',acceleration_controller=(RLController,{}),num_vehicles=4)
```

These first two lines of code let the user define the starting vehicles inside the simulation. In this example, 11 human-driven vehicles and 4 AV are obtained. The specific number is linked to the configuration adopted and chosen between the two possible scenarios: 20% and 80% of AV in the network. The acceleration controller command refers to the car following model used for the vehicles defined. In this thesis work, the Intelligent Driver Model (IDM) is used for humanly driven vehicles and the reinforcement learning model (RL) for AV. The car following model influences the vehicle's behaviour with respect to the other vehicles. Many coefficients can be defined, such as the minimum gap when standing, acceleration and deceleration abilities, driver delays or the ability to follow a speed value. AVs will consider their policy to optimise all of these variables and take completely independent decisions.

```
USE_INFLOWS:

    inflow.add(
        veh_type='human',
        edge='-E1',
        #probability=INFLOW_PARAMS['human']
        vehs_per_hour = 280,
        route = "route-E1_0",
    )

    inflow.add(
        veh_type='ai',
        edge='-E1',
        #probability=INFLOW_PARAMS['ai']
        vehs_per_hour = 70,
        route = "route-E1_0",
    )
```

These code lines define all other vehicles of the simulation. They are split into sub-flows for which are defined: the vehicle type, human or AV; the starting edge; the route to be followed and the principle for which they are inserted into the simulation. Initially, the probability logic has been used, imposing a x probability to AV and a $(1 - x)$ probability to humanly driven ones, with x equal to 20% or 80%. Throughout the development phase, the logic has been changed to vehicles per hour. This has been done to have a final deterministic and perfectly replicable simulation, which is needed to have comparable results. In total, 12 sub-flows are considered, divided as follows:

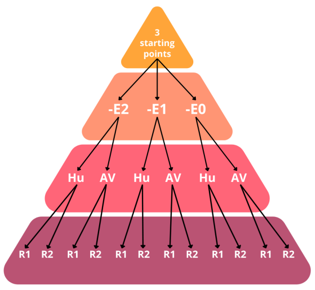


Figure 2.11: Sub-flows configuration. "Hu" stands for Humanly-driver vehicles; "AV" stands for Autonomous Vehicles; "R1" and "R2" stand for the two possible routes for every edge.

The final 12 sub-flows have a vehicle per hour value equal to 280 for human-driven vehicles and 70 for AVs and vice versa for the scenario with 80% of AVs. There are two possible routes: taking the first or the second exit of the roundabout for all three starting edges.

```
sim_params = SUMOParams(render=False, sim_step=0.005, num_clients=2)
```

This line of code defines the basic SUMO parameters for the simulation. The network can be rendered or not. This will obviously increase the time needed for every step and for this reason no visualisation is used, apart from the one on DriveSim. The simulation step is set to 0.005 s, which has been considered the optimal compromise between the minimum value equal to 0.001 s and the necessity to exchange all data during the simulation. Finally, the number of clients for the specific communication must be defined in advance, as specified in section 1.10 for the TCP protocol. In this case study, the two clients are terminal 1 and terminal 2.

```
flow_params['env'].horizon = 16000
exp = Experiment(flow_params)
_ = exp.run(1)
N_CPUS = 2
N_ROLLOUTS = 1
ray.init(num_cpus=N_CPUS)
alg_run = "PPO"
agent_cls = registry.get_agent_class(alg_run)
config = agent_cls._default_config.copy()
config["num_workers"] = N_CPUS - 1
config["train_batch_size"] = 2000
config["gamma"] = 0.999
config["model"].update({"fcnet_hiddens": [16, 16]})
config["use_gae"] = True
config["lambda"] = 0.97
config["sgd_minibatch_size"] = 2000
config["kl_target"] = 0.02
config["num_sgd_iter"] = 10
config["horizon"] = flow_params['env'].horizon
```

This code section defines all the policy characteristics and runs the simulation. The algorithm used for policy training is Proximal Policy Optimisation (PPO). This policy gradient method learns from online data as well. This is done to ensure low variance

during training and guarantee that the updated policy is not too much different from the old policy over the simulations. Furthermore, in PPO clipping is taken into account, thanks to which it is considered just an interval where the policy variation is obtained. The policy knows if in general its action was good or not, but it does not know if the action will still be positive far from its actual position. γ discounts all future moves, with respect to the immediate next one and is set to 0.99. The neural network used has 16 hidden layers. Generalised advantage estimation (GAE) is used to evaluate the difference between what the policy predicted an action would return and the actual return it got. Lambda is a GAE parameter used for reducing the variance in training which makes it more stable. The Kullback-Leibler divergence (KL divergence) is a measure of how a probability distribution differs from another probability distribution. In this case, this is equivalent to minimising the difference between an approximate distribution and the true data distribution. Stochastic gradient descent logic is used to optimise the policy and the number of SDG iterations is expressed.
For the preliminary simulation environment, the policy has been trained considering only the crossing time of the roundabout. From preliminary tests on the SUMO simulation, without considering the whole simulator configuration, the policy has been effectively able to reduce the crossing time inside the roundabout and a quantitative difference has been obtained between the two scenarios in which the percentage of AV is 20% or 80%, as described in section 3.2. No consideration has been carried out on any other factor such as comfort or safety. This is done also considering that these preliminary tests are carried out to get a complete description of the problem, with its points of strength and criticalities. One of the effects related to not having considered comfort and its subsequent solution is described in section 4.4.

2.4.2. Second level communication

The second level communication refers to the one created between SUMO and DriveSim. This is the core communication along all the simulations, connecting the two environments in which tests are carried out. Its main objectives are data retrieval and submission, data storage for future analysis and algorithms implementation. One of the leading factors considered in all development phases is the time consumption of any command added. The time available to exchange all data is really short and any change could cause an increase in the time required, exceeding the limit imposed.

```
UDP_settings = {'UDP_IP_SELF': '192.168.100.77',
                'UDP_IP_CONCURRENT': '192.168.100.22',
                'UDP_PORT_SELF': 30001,
                    'time_out': 10,
                    'n_vehicles_worldsim': len(scenerio['id_vei_worldsim']),
                    'n_variables_vehicle_worldsim': 5,
                    'n_variables_ego': 3}
UDP_settings['buffer_size'] = 8 * (1 + UDP_settings['n_variables_ego'])
UDP_settings['encoding_receiving'] = '<' + str(1 +
    UDP_settings['n_variables_ego']) + 'd'
UDP_settings['encoding_sending'] = '<' +
    str(UDP_settings['n_variables_vehicle_worldsim']) + 'd'
```

First of all, UDP communication is implemented. This is the fastest communication protocol and lets terminal 2 send all data to DriveSim and back to SUMO. Its main settings are the IP addresses of the two clients. The concurrent refers to the driving simulator. As described in section 1.10, specific ports must be defined. The last important detail to be declared is the number of variables which DriveSim receives from SUMO (e.g. speed, x and y position) and the ones sent to DriveSim to move correctly all vehicles (e.g. x and y position, direction, speed and steering angle).

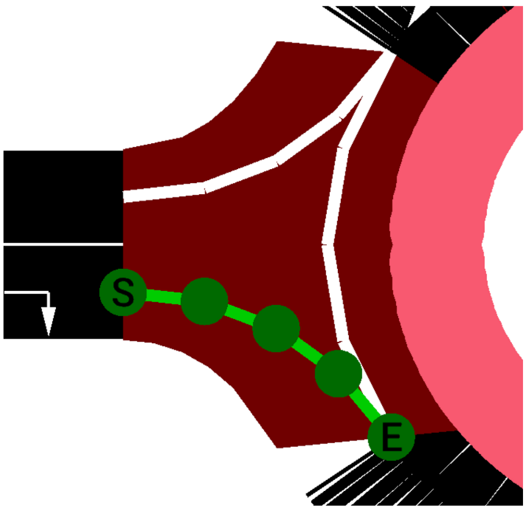
```
lane_interp[":J0_0"]={"type": "curve", "data": (18.44,11.87,10.28,-0.24,2.64)}
lane_interp["-E2"]={"type": "straight", "data": (0,0)}
```

Two types of lanes are considered: straight lanes and curves. With regard to the straight lanes, the user must define the steering and yaw angles of the vehicle. Curves must be interpolated to get a function since the lanes geometry definition in SUMO is not enough precise to move correctly the vehicles in DriveSim. In these preliminary tests, curves are considered as arcs of circumference for which the user must define centre coordinates, radius, steering angle and reference yaw. This is a simple interpolation of curves but coherent with the need of keeping as low as possible the computational time.

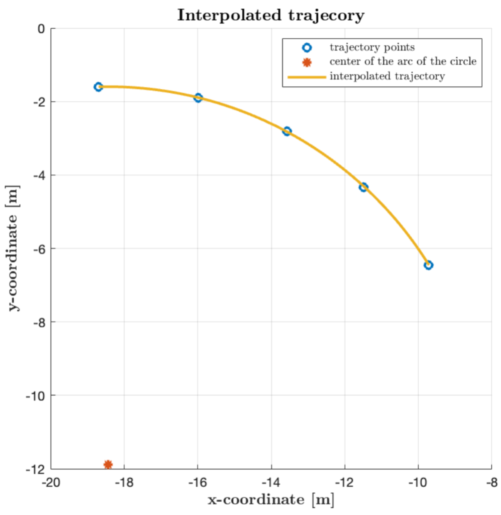
```
if corsia['type']=='curve':
    beta=np.arctan2(x[1]-lane_interp['data'][1],x[0]-lane_interp['data'][0])
    if beta<0:
        beta=beta+2*pi

    string[veh_id]['x_interp']=lane_interp['data'][2]*np.cos(beta)
        +lane_interp['data'][0]
    string[veh_id]['y_interp']=lane_interp['data'][2]*np.sin(beta)
        +lane_interp['data'][1]
    string[veh_id]['delta']=lane_interp['data'][3]
```

More in detail, this is the first algorithm used to interpolate curves inside the roundabout. Once the specific lane ID is acquired through TraCI, its corresponding data are used to get the arc of circumference and the final interpolated point coordinates are saved.



(a) Original trajectory in SUMO.



(b) Interpolated trajectory.

Figure 2.12: Comparison between the original trajectory obtained in SUMO and the interpolated one.

In 5.7 is shown an example of the interpolation obtained. In this case, junction J4 is considered and, specifically, lane $J4 - 2$ is taken into account. Considering the original trajectory, all points are defined and among them are highlighted the starting and the ending ones. All points are kept equal in the interpolated trajectory to preserve the simulation functionality in SUMO. The resulting path is a function and not a sum of straight lines linking interpolation points as defined in SUMO. An alternative to this approach would be to define in SUMO many points through which the final trajectory is designed. This solution is not as good as the one firstly proposed and still introduces a percentage of error.

```
for vei_id in scenario['veicoli_attivi']:
        traci.vehicle.subscribe(vei_id, (tc.VAR_POSITION, tc.VAR_SPEED,
            tc.VAR_ANGLE, tc.VAR_ROAD_ID, tc.VAR_DISTANCE))

values = traci.vehicle.getAllSubscriptionResults()
```

To acquire all data from SUMO, TraCI subscriptions are used. They boost TCP communication performances and let it deal with up to 50.000 vehicles per second: almost doubling the number of dealt vehicles per second. Subscriptions allow the user to ask all at once for a complete set of vehicle and simulation variables, which will be retrieved at every simulation step. Once all variables are obtained, they are assigned to specific python variables and manipulated to obtain the final values sent to DriveSim. For example, the x and y coordinates must be changed by applying the offset between the two scenarios, SUMO and the driving simulator.

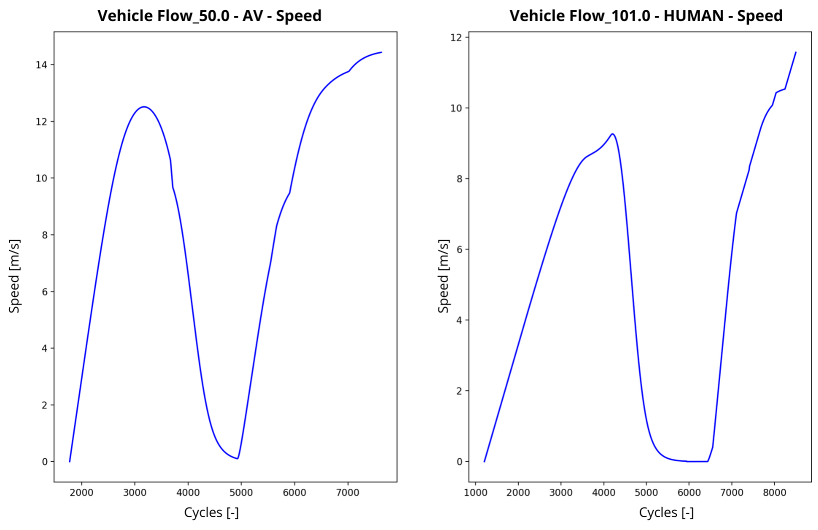


Figure 2.13: Speed values retrieval obtain from TraCI.

Figure 2.13 shows an example of data retrieval from TraCI. In this example, speed is compared between a generic human-driven vehicle and an AV. The speed profile is similar, denoting an AV behaviour coherent with the driver model used inside the simulation and a correct policy training.

```
cmd = "ps aux | grep SUMO"
stdoutdata = subprocess.getoutput(cmd)

PORT = int(stdoutdata.split()[index+1])
traci.init(PORT,tc.DEFAULT_NUM_RETRIES,"127.0.0.1")
traci.setOrder(2)

while traci.simulation.getMinExpectedNumber() >= 0
    ...
```

Once every preliminary action has been made, the terminal retrieves the open port linked to the SUMO simulation started by Flow. As depicted in section 1.10, the TCP also

needs the hierarchy of the clients, which is expressed at line 2. Line 8 gives the real start to the loop evaluated every time step. One important difference with the scenario in which TraCI is used to retrieve data from a simulation started with TraCI itself is that *traci.simulation.getMinExpectedNumber()* must be set >= 0 and not just > 0. This is a crucial step for the communication implementation phase and one of the most relevant details to be considered. This way the simulation loop for TraCI continues even if the number of vehicles in the simulation is equal to 0, which is the condition at the beginning of the simulation since vehicles are created by Flow and not directly by SUMO. The loop will be closed by the user as described in section 2.3.
In figure 2.14 it is possible to see the time step duration over a simulation of 50 seconds. The computational time remains below the limit of 0.005 seconds imposed as time step duration in the simulation. As the simulation starts, the number of vehicles inside the roundabout is smaller and the computational time needed to interpolate curves is low. Once many vehicles approach the roundabout, many more trajectories must be interpolated and the simulation time step reaches its maximum. It is possible to see also some out-layer data higher than 0.005 s, but they do not affect the simulation, since it is almost unnoticeable, being a total of 0.1 s.

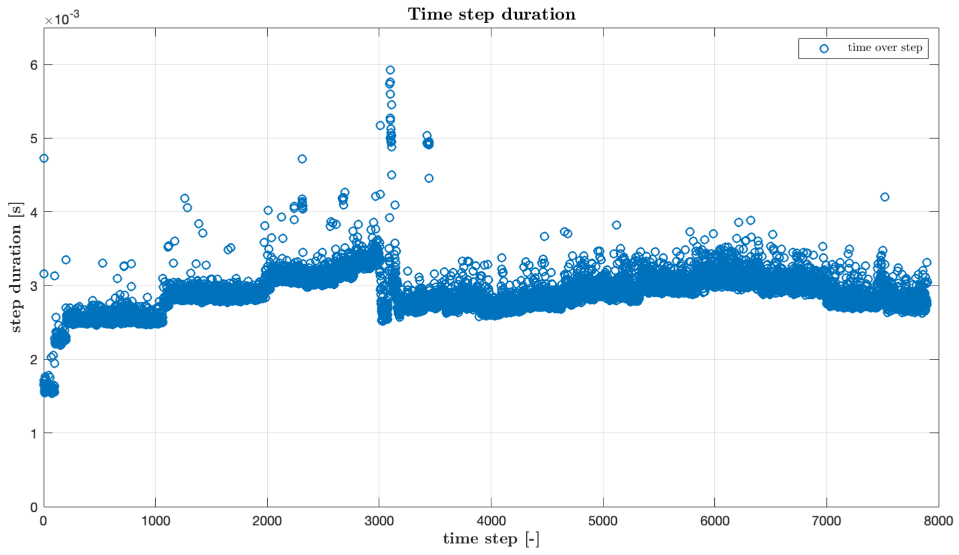


Figure 2.14: Time step computational time over the whole simulation.

The time step duration has a mean of 0.0029 seconds, making available about 0.0021 seconds to develop new and more complex interpolation algorithms. Terminal 2 will wait for DriveSim to exchange data during the final simulations. The simulation step will be fixed and equal to 0.005 seconds. The simulation environment is complete and can be used to acquire the first results.

3 | Preliminary tests and questionnaire

Once the simulation environment has been completely developed, it is possible to carry out the first experimental tests and evaluate the system performances, both from a technical point of view and from a psychological one considering the human feelings inside the driving simulator. This chapter is devoted to a detailed description of the tests, the driving simulator employed to perform them and the questionnaire used to evaluate the results. From these results, it will be also possible to understand which modifications are required to improve the simulation environment effectiveness.

3.1. Preliminary tests

The main objective of preliminary tests is to understand the simulation environment's ability to replicate reality. On this property depends all the system performances and the manner in which the user perceives the simulation and how much they are able to feel it as a real situation. The second objective is to understand if the user is able to appreciate any difference between the two main scenarios considered: 20% of human-driven vehicles and 80% of AVs and vice-versa. These two scenarios are chosen to simulate the two opposite conditions, where most vehicles are human-driven or AVs. Since the policy has been trained to minimize the crossing time, the main difference that the user should feel is a reduction of this time and a greater smoothness inside the roundabout.
Specifically, every user will perform six simulations. They will start from all three legs of the roundabout two times, considering the two scenarios previously described. In table 3.1 a detailed description of the tests configuration is presented.

Scenario	Starting point
20% of human-driven vehicles and 80% of AVs	Leg 1 Leg 2 Leg 3
80% of human-driven vehicles and 20% of AVs	Leg 1 Leg 2 Leg 3

Table 3.1: Preliminary tests simulations.

An important detail is the order in which the six simulations are carried out. To let the user better estimate the differences between the two simulation scenarios, every leg is simulated before with the first scenario configuration and immediately after with the second scenario configuration.

Test	Scenario	Starting point
1	20% of human-driven vehicles and 80% of AVs	Leg 1
2	80% of human-driven vehicles and 20% of AVs	Leg 1
3	20% of human-driven vehicles and 80% of AVs	Leg 2
4	80% of human-driven vehicles and 20% of AVs	Leg 2
5	20% of human-driven vehicles and 80% of AVs	Leg 3
6	80% of human-driven vehicles and 20% of AVs	Leg 3

Table 3.2: Preliminary tests simulations order.

Table 3.2 presents the detailed order of the six simulations that every user must perform. For all three legs, the starting position is set at the beginning of the straight corresponding lane and users must take the second exit of the roundabout. The waiting time between all simulations must be as limited as possible to avoid loss of information. In this simulator configuration, the highest waiting time is obtained when the starting leg is changed since a new scenario must be loaded.
Participants are unaware of the specific scenario they are facing. The only information they receive is that they will repeat every starting point two times, considering two different scenarios with some configuration differences and that they will perform six simulations in total. Finally, they also know that every time the first simulation of a specific leg refers to the same scenario and the same holds for the second simulation.

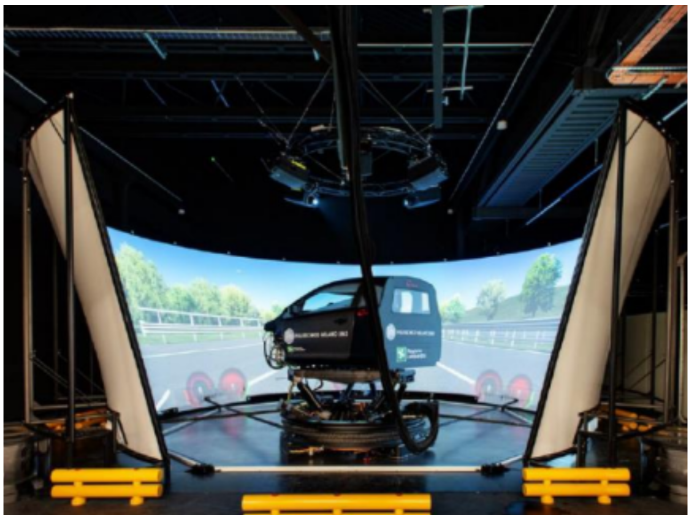


Figure 3.1: Dynamic driving simulator used to perform the simulations.

Figure 3.1 shows the driving simulator used to perform all tests. The simulator moves the vehicle chassis thanks to six electric actuators, allowing for all three displacements, along x,y and z directions, and three rotations, yaw, roll and pitch. The longitudinal and lateral displacements are allowed by the pull of four cables, controlled by four independent electric motors decoupling the two displacements. All simulations are performed in dynamic configuration. This means that all actuators moving the vehicle chassis are active. Also, seat belts are operated and used to provide the feeling of a braking manoeuvre to the user.

3.2. Questionnaire

The questionnaire has been built considering the objectives discussed in the previous subsection and also to acquire data as much as possible comparable. Both quantitative and qualitative information has been retrieved, to have a complete description of the simulations from the point of view of the user. Since the participants cannot know to which specific scenario the questionnaire refers, in the following sections first and second scenarios do not refer to a specific configuration, but to the specific order every user experienced. The first section of the questionnaire is devoted to acquiring general participants' information. Specifically, it is asked:

- name, surname and e-mail address;
- if they need eyeglasses for driving;
- if they have experience with driving videogames;

- how many years they have held their driving license.

This first set of information can be divided into two main groups: the first one is composed of personal information and contact details; the second one collects data on the user's expertise both about driving and playing videogames since the driving simulator experience can be similar to the one of a simpler simulator. This second group is important to understand if the participants' behaviour is in any way conditioned by their experience.
The second questionnaire's section is used to understand if the policy behaviour has been considered correct by the user. Questions are repeated for the first and the second simulation scenarios, considering that the participant cannot know which is the one referring to the 20% of AVs and vice-versa: only test operators know which are the specific scenarios. In particular, it was asked:

1. if the vehicles in the simulation respected the rules of right of way (i.g. the vehicles inside the circulatory roadway have right of way over all other vehicles);
2. how many vehicles did not follow the rules of right of way;
3. if the vehicles ahead waited too much time before entering the roundabout.

To measure effectively the user answers and to obtain comparable results the Likert's scale has been used. Participants can answer the questions in this section by choosing from the following options:

- completely disagree;
- disagree;
- neither agree nor disagree;
- agree;
- completely agree.

The second question is done only if the user gives a negative answer (i.g. disagree, completely disagree) to question one and they can indicate whether one or more than one vehicle has not respected the rules of right of way. It has been chosen as a reference value one vehicle, since every user will take the second exit of the roundabout. For this reason, two critical points are obtained in terms of right of way. The main policy characteristics that have been investigated refer to its capacity of respecting the general rules inside a roundabout and its ability to understand if it can or cannot enter the roundabout without crashing into other vehicles. The last question has been asked also to measure

some behavioural responses of users during the simulation. If a vehicle ahead waits too much before entering the roundabout, users tend to be nervous and take wrong decisions or perceive their surroundings erroneously, leading to an effect on the data acquired.
The third and last section of the questionnaire is used to directly compare the two scenarios from a smoothness and safety point of view. Specifically, it is asked to the user:

- if they perceived the first scenario more smooth than the second one;
- if they felt safer in the first or in the second scenario;
- which scenario they prefer.

As for the second section, to obtain comparable results the answers follow a Likert's scale and express a:

- significant difference or preference both in negative and positive;
- partial difference o preference both in negative and positive;
- no difference or preference.

The smoothness of the scenario in which 80% of vehicles are AVs should be one of the most important perceptions of users. For this reason, the first question is one of the most important ones and the one which will validate participants' answers, since the policy actually reduces the crossing time. Tests have been conducted on the effect of the AVs inside the simulation environment. Crossing time is calculated for all vehicles in the simulation and a final average time is obtained as follows:

$$\mu(t) = \frac{\sum_{i=1}^{n}({t_{vi}}^{out} - {t_{vi}}^{out})}{n} \tag{3.1}$$

For every vehicle, the crossing time is described as the difference between the time instant at which that vehicle enters the circulatory roadway and the one at which it exits the roundabout. Considering n vehicles, the average time μ is calculated. Results are presented in table 3.3

AV percentage	$\mu(t)$ **[s]**
10	6.33
20	6.27
30	6.18
40	6.11
50	5.88
60	5.26
70	4.73
80	4.72
90	4.69
100	4.66

Table 3.3: Average time as a function of the percentage of AVs in the simulation environment.

The results show a difference of about 2 seconds between the scenarios considered in preliminary tests. This average crossing time is limited since it considers only the circulatory roadway. Section 5.2.1 will present the same value with respect to the complete path of vehicles in the simulation.
Question two is also relevant from a psychological point of view: as for the third question of the second section, feeling unsafe directly modifies the user behaviour in the simulation, making them less careful of surrounding details and compromising the results. The last question asks for a direct preference of the user. This question is added to understand if, in general, the participant's answers are coherent in the two main sections or if compilating errors have been made. To complete the questionnaire presentation, it is important to cite that previous versions considered many more questions with a greater level of detail to understand as much as possible all feelings and perceptions of users during the simulation. For example, also the personal meaning of safety was investigated by asking how many times the participant got distracted or felt in danger. Considering the smoothness of the simulation, a precise question on the perceived crossing speed was asked. These questions would have led to many more details and a clearer depiction of simulation criticalities. On the other hand, considering too many questions in a such complex questionnaire for the user could have produced many more compilating errors and results unusable from a scientific point of view. For this reason, it was preferred to have a restricted version, also considering the small number of participants in these preliminary tests.

Section	Question	Type	Possible answers
Personal info	Name		
	Surname		
	E-mail		
	Do you need eye-glasses to drive?		
	How many km do you drive annually?		
	How much experience do you have with video games?		
First simulation and second simulation perception			
Perception	How much do you agree with the following statement? The vehicles in the simulation obeyed the rules of right of way.	Likert's scale	• completely disagree • disagree • Neither agree nor disagree • agree • completely agree
	If you answered the previous question by indicating "Disagree" or "Completely disagree", how many vehicles did NOT obey the rules of right of way?	Single answer	• one vehicle • more than one vehicle
	How much do you agree with the following statement? The vehicles in front of mine waited too long before entering the roundabout generating traffic.	Likert's scale	• completely disagree • disagree • Neither agree nor disagree • agree • completely agree

First and second scenarios comparison			
Comparison	In relation to traffic flow, with which of the following statements do you most agree?	Likert's scale	• Traffic in scenario 1 was significantly smoother than in scenario 2 • Traffic in scenario 1 was partially smoother than in scenario 2 • I perceived no difference in the smoothness of traffic in the two scenarios • Traffic in scenario 2 was partially smoother than in scenario 1 • Traffic in scenario 2 was significantly smoother than in scenario 1
	With respect to the feeling of safety in the traffic situation, with which of the following statements do you most agree?	Likert's scale	• In scenario 1 I felt significantly safer than in scenario 2 • In scenario 1 I felt partially safer than in scenario 2 • I perceived no difference between the 2 scenarios in the feeling of safety • In scenario 2 I felt partially safer than in scenario 1 • In scenario 2 I felt significantly safer than in scenario 1
	Overall, which of the two scenarios did you prefer?	Likert's scale	• I significantly preferred scenario 1 to scenario 2 • I partially preferred scenario 1 to scenario 2 • I can't say which of the two scenarios I preferred • I partially preferred scenario 2 to scenario 1 • I significantly preferred scenario 2 to scenario 1

Table 3.4: Complete questionnaire used to evaluate simulation results.

3.3. Preliminary simulations results

Preliminary results are important to understand the overall goodness of the model and to address all its criticalities. The test has been submitted by a total of twelve participants. All of them successfully completed all six simulations. The two scenarios were proposed alternately between successive participants. Down below are presented the results of the two main sections of the questionnaire, referring to the user perception and comparison of the two scenarios.
The second section results can be split into two subsections: the first one with regards to the rules of right of way and the second one considering traffic entering the roundabout.

First Question	Quantity
Completely agree	0
Agree	5
Neither agree nor disagree	1
Disagree	5
Completely disagree	1

(a) Results of the first question of the second section of the questionnaire.

Second Question	Quantity
One vehicle	4
More than one vehicle	2

(b) Results of the second question of the second section of the questionnaire.

Table 3.5: Perception of compliance with the rules of right of way in the roundabout, considering the scenario with 80% of AVs.

First Question	Quantity
Completely agree	1
Agree	4
Neither agree nor disagree	1
Disagree	6
Completely disagree	0

(a) Results of the first question of the second section of the questionnaire.

Second Question	Quantity
One vehicle	4
More than one vehicle	2

(b) Results of the second question of the second section of the questionnaire.

Table 3.6: Perception of compliance with the rules of right of way in the roundabout, considering the scenario with 20% of AVs.

Tables 3.5 and 3.6 show the results in terms of perception of compliance of the right of way with respect to the 80% AVs and 20% AVs scenarios, respectively. No relevant difference can be seen between the two scenarios considered. In both of them, some of the vehicles have not been able to respect the rules of precedence. Considering that the answers refer to all three simulations of a specific scenario, in general, not so many vehicles failed in respecting the rules of right of way. The main reasons for which this happened are:

- the AVs policy has been trained considering only the crossing time. AVs will try to enter the roundabout even if the space available is limited, resulting in a perception of not respecting the rules. In some cases, this will also lead to crashes, reducing also the safety perception of participants;
- due to SUMO architecture, vehicles are able to see all others, only if they follow a path near the entrances and exits of the roundabout. Some of the participants cut the corners of the roundabout a lot, making the SUMO vehicles and AVs not able to see them.

This will need to be considered in the final network architecture, as described in section 3.4.

Third Question	**Quantity**
Completely agree	0
Agree	1
Neither agree nor disagree	4
Disagree	6
Completely disagree	1

(a) Results of the third question of the second section of the questionnaire, referred to 80% AV scenario.

Third Question	**Quantity**
Completely agree	0
Agree	3
Neither agree nor disagree	6
Disagree	2
Completely disagree	1

(b) Results of the third question of the second section of the questionnaire, referred to 20% AV scenario.

Table 3.7: Perception of traffic in the roundabout.

Table 3.7 shows the results of the last question of the perception section. Traffic has been directly related to the waiting time before entering the roundabout and its dependence on the capacity of both SUMO vehicles and AVs to quickly get into. Looking at the answer it is possible to highlight a slight preference of users to the scenario in which 80% of the vehicles are AVs. In fact, nine participants expressed neutral or positive opinions in the 20% AVs scenario and just five in the 80% one.
The third section refers to the direct comparison of the two scenarios. The tables below present the result of this section.

TRAFFIC SMOOTHNESS	
First question	**Quantity**
Traffic in scenario 1 was significantly smoother than in scenario 2	0
Traffic in scenario 1 was partially smoother than in scenario 2	4
Traffic in scenario 2 was partially smoother than in scenario 1	5
Traffic in scenario 2 was significantly smoother than in scenario 1	1
I perceived no difference in the smoothness of traffic in the two scenarios	2

(a) Comparison of perception of traffic's smoothness in the roundabout.

TRAFFIC SAFETY	
Second question	**Quantity**
In scenario 1 I felt significantly safer than in scenario 2	1
In scenario 1 I felt partially safer than in scenario 2	2
In scenario 2 I felt partially safer	3
In scenario 2 I felt significantly safer than in scenario 1	0
I perceived no difference between the 2 scenarios in the feeling of safety	6

(b) Comparison of perception of safety in the roundabout.

GLOBAL PREFERENCE	
Third question	**Quantity**
I significantly preferred scenario 1 to scenario 2	1
I partially preferred scenario 1 to scenario 2	2
I partially preferred scenario 2 to scenario 1	4
I significantly preferred scenario 2 to scenario 1	2
I can't say which of the two scenarios I preferred	3

(c) Global preference between the two scenarios.

Table 3.8: Anwers of the third questionnaire's section, referring to the scenarios comparison.

Results have been collected in such a way that scenario 1 always refers to the scenario in which 20% of vehicles are AVs and scenario 2 to the one in which 80% of vehicles are AVs. Looking at the answers, in terms of traffic smoothness participants did not perceive relevant differences. A really slight preference can be seen for scenario 2, but data do not show an absolute preference. This should have been the most important detail observed by users since the policy has been trained only on this parameter. Possible reasons are:

- the participants' number is too small to appreciate a relevant and absolute difference between the two scenarios;
- every user had to face many simulations one by another, without almost no time to process the information received;
- simulations lasted from a minimum of 25 seconds to a maximum of 40 seconds. Participants had no time to appreciate differences.

Also in terms of safety, no differences have been perceived: half of the participants found no difference between the 2 scenarios. This is a positive result since the AVs errors do not depend on the policy but on the SUMO simulation environment and its logic. On the other hand, to minimise vehicle errors during the simulation, modifications to the network must be considered.
Considering global preference, the scenario in which 80% of vehicles are AVs has been overall preferred: half of the participants slightly or significantly preferred it.
The last part of this results presentation is devoted to the analysis of the coherence of all the participants. This analysis is fundamental to understand if some of the users answered without a full comprehension of the questions or without remembering the details of all the simulations they carried out.

Participant ID	**Comments on their answers**
1	Prefers the 20% scenario having reported a vehicle disregarding the right of way in the 80% scenario (consistent with the fact that the 80% scenario was perceived to be significantly less safe). Continues to prefer it despite the fact that the 80% scenario was perceived to be smoother
2	Prefers the 80% scenario. In agreement with the fact that in that scenario the precedences were followed correctly, unlike the 20% scenario. Continues to prefer it despite being perceived as less smooth
3	Prefers the scenario with 80% (for slightly greater safety). No vehicles broke the right-of-way rules in either scenario. However, in the 80% scenario, they noted less smoothness and vehicles stopped for longer before entering the traffic circle.

Participant ID	Comments on their answers
4	They preferred the 80% scenario. They felt safer in the 80% scenario despite a vehicle disregarding the right of way. The preference can be attributed to the smoother traffic flow
5	They preferred the 20% scenario. In the 80% scenario, more than one vehicle did not respect the right of way and vehicles waited too long before entering the traffic circle
6	They preferred the 80% scenario. The preference is related to smoothness alone. In the 80% scenario, one vehicle did not respect the right of way.
7	They preferred the 80% scenario, although they felt less safe in this scenario because of a vehicle that did not respect the right of way. The preference can be attributed to only partially better flow
8	They preferred the 80% scenario. However, they perceived no difference in either safety or smoothness. In both cases, they also reported a high waiting time when entering the traffic circle due to the misbehaviour of other vehicles
9	They did not prefer either scenario. In both scenarios, one vehicle did not respect the right of way. The 20% scenario was rated smoother
10	They preferred the 20% scenario even though they described it as less smooth. They did not detect any difference in safety, although, in the 20% scenario one vehicle did not respect the right of way.
11	They did not prefer either scenario. They found no difference in safety, although in scenario 20% one vehicle failed to yield the right of way.
12	They preferred neither scenario. In both scenarios, more than one vehicle disregarded the right of way.

Table 3.9: Coeherence analysis on participants' answers. Green highlights coherence, yellow not complete coherence, and red incoherence.

This analysis indicates a general coherence in participants' answers. Unfortunately, two of them showed a complete inconsistency between the answers they gave in different sections of the questionnaire. Considering the overall small number of participants, there is no point in neglecting their answers, but this highlights the need of modifying both the simulations structure and the questionnaire. The consistency analysis also demonstrates that different participants have different ideas about what safety represents and which scenario they prefer. Many of the users preferred scenarios less safe but smoother. In the final questionnaire version, this difference should be considered and questions should be adapted to understand what every specific user considers as the best condition.

3.4. Comments and modifications

Results obtained through preliminary tests can be used to understand the main criticalities of the simulation environment and propose modifications to get a final environment as much as possible similar to reality. Many data have been collected, showing some major weaknesses of the network:

1. vehicles are not always able to detect the ego car inside the circulatory roadway;
2. traffic does not represent a realist scenario;
3. vehicles do not follow trajectories adherent to reality;
4. participants did not understand most of the time if a vehicle was going to exit from the roundabout or not.

With regard to the first issue, it depends on the fact that the circulatory roadway is too wide for the preliminary roundabout. Being too wide and considering that vehicles in SUMO are able to detect just one point per vehicle, participants often moved along trajectories near the central island. It is possible to enlarge the central island and reduce as much as possible the road width, considering the limitation of having to obtain a realistic network. Further tests should be done on the final network to verify that this solution effectively fixes the problem.
The second problem must be carefully addressed. In this preliminary network, the number of vehicles has been chosen considering the data exchange limit in terms of computational time. This led to a network in which queues are artificially built, with the only need of having the participants encounter more AVs or human-driven vehicles according to the specific scenario. This problem is also linked to the small simulation time, discussed in section 3.3. To solve these two problems a complete calibration of the model must be considered. By calibrating the model it will be possible to obtain the specific number of vehicles, their departure and arrival times and also the effect of their interaction with other traffic components like bicycles or pedestrians. It is also crucial to start from a real

roundabout model and not build an artificial one: this way also approaching legs and geometry, in general, will be realistic and will lead to longer crossing times. An example of calibration will be presented in section 1.2 for the final simulation environment.
The last two points are related to one another. To fix them two solutions have been adopted:

- upgrade the trajectories interpolation inside the roundabout;
- use car turn signals to give a visual indication of the vehicle direction.

The first solution is the most important modification applied to the preliminary network and the most complex one. To upgrade the interpolation of trajectories it is needed to use interpolating functions of a higher degree, considering the limitation given by the computational time, which in total cannot be higher than 0.005 seconds. Two interpolation algorithms were created, described in detail below.

First interpolation algorithm

The first interpolation algorithm is more precise but requires a higher computational time to be processed. The interpolator used is the one inside python, which will create an interpolated function passing through some points given by the designer. The first phase will be to define this points. To do it as precisely as possible has been used a dense grid, in which the lines are 3 cm apart.

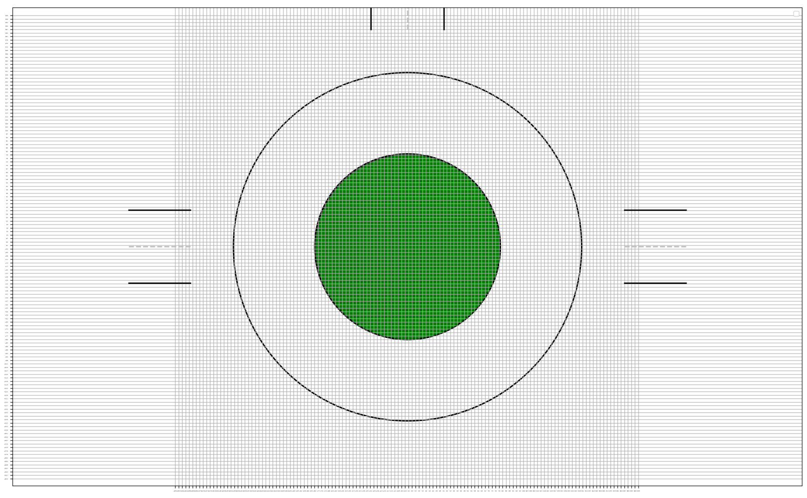


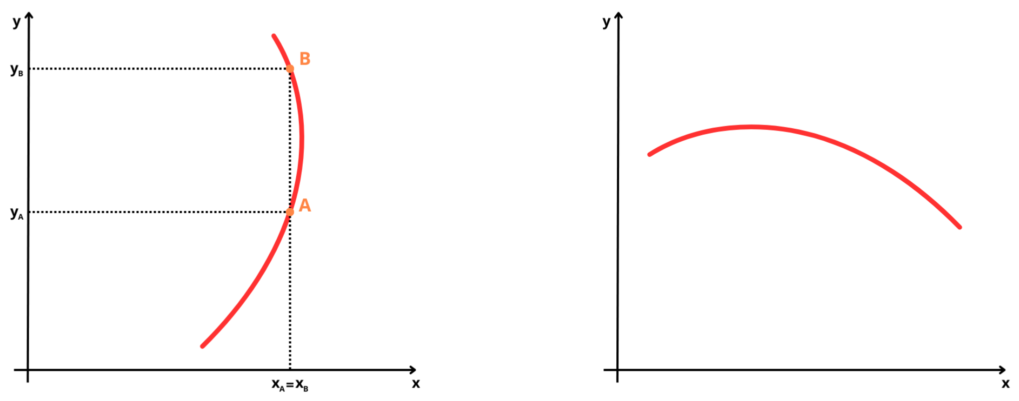
Figure 3.2: Grid used to obtain waypoints for interpolated trajectories.

Trajectories are, therefore, firstly sketched by the designer. Waypoints are obtained and, finally, given to python to obtain the final digital interpolated trajectory. The number of waypoints is not fixed and depends on the specific shape to be obtained. When designing

the trajectory, the designer must take into account the fact that the starting and ending points are fixed by the straight lanes reaching the roundabout. To prevent vehicles from making sudden rotations when entering or joining the circulatory roadway, they should exit and enter as much as possible parallel to the straight lane direction. For this reason, many points are obtained at the end and at the beginning of every trajectory.
This first phase divides all the trajectories into two main groups:

- the ones for which the interpolator is directly able to get the final function;
- the ones for which the interpolator is not able to get the final function.

The second groups refer to the trajectories in which two or more points have the same x coordinate. In this case, the interpolator cannot be used. To fix this problem the following solution has been proposed:



(a) Original designed curve. (b) Rotated curve.

Figure 3.3: Curve rotation to solve interpolation problem.

As shown in figure 3.3, the trajectories belonging to the second group must be rotated by 90 degrees to have a final shape in which no points have the same x coordinates. This can be easily done by switching the x and y original coordinates, which will then be given to the interpolator. Considering this case, complete trajectories may be obtained as a composition of two or more sub-trajectories, in order to avoid getting a trajectory impossible to be interpolated: a path for which both original and rotated trajectories has one or more points which share the same x coordinate. Final trajectories have been designed also minimising the number of sub-paths to be interpolated and, finally, getting a network as much simple as possible.

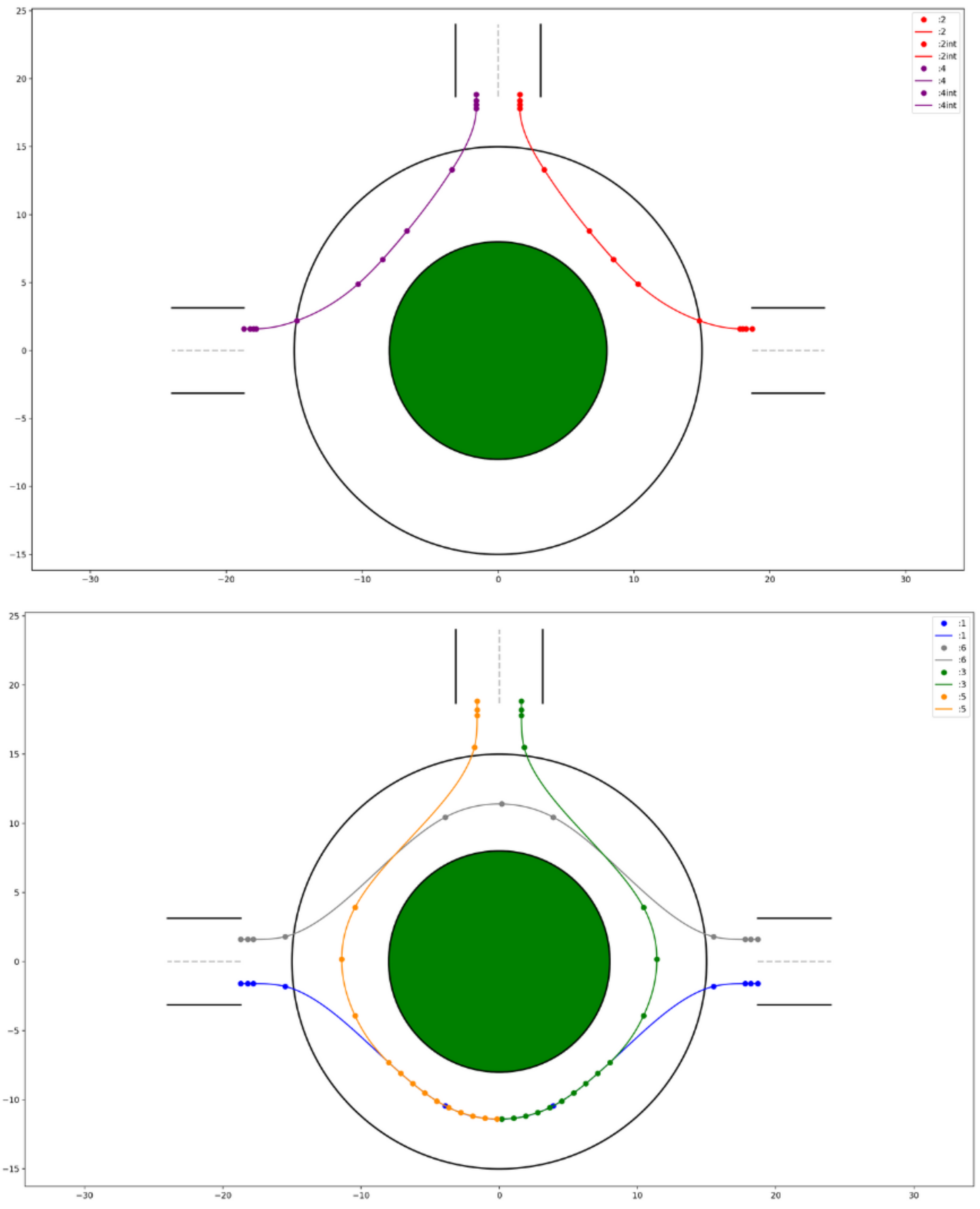


Figure 3.4: Final interpolated trajectories.

Figure 3.4 presents all final trajectories and their waypoints. For all entering lanes three trajectories have been obtained linked to the three possible exits, for a total of nine

possible paths. Some of these paths are obtained as a composition of many trajectories. For example, a vehicle coming from the upper straight lane and exiting at the second exit will cross trajectories 5 and 1. Trajectories 2 and 4 are divided to get the vehicles as much parallel as possible to the entering and exiting lanes, as described before. From figure 3.4 it is also possible to see problem number one previously presented: the circulatory roadway is disproportionately large, making it also harder to create realistic paths. Considering trajectories 5 and 3, many waypoints have been obtained when they join with trajectory one, in order to have the final path without any sudden change of direction.
Once trajectories have been completely defined, it is possible to describe the algorithm used to obtain the final interpolated points, starting from the position given by SUMO. The figure below presents graphically this algorithm.

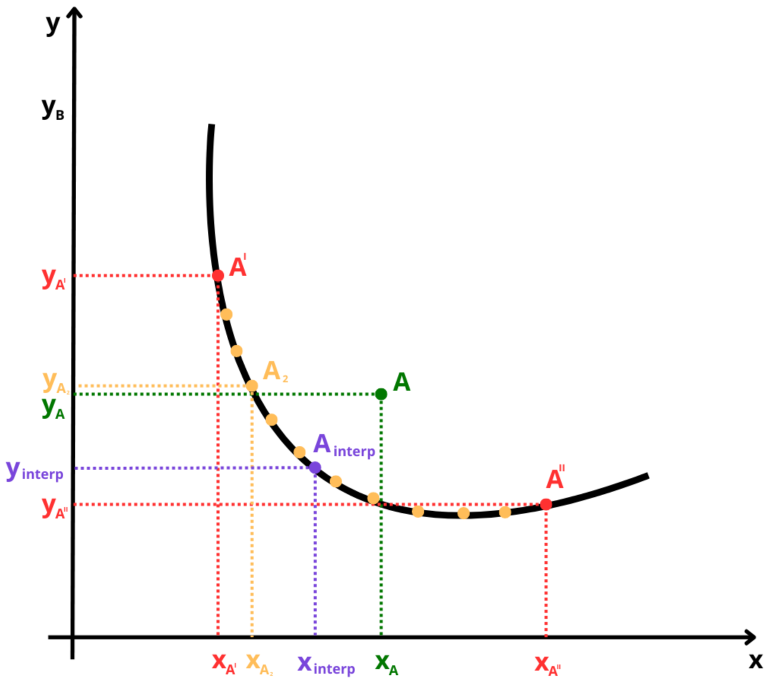


Figure 3.5: Graphical representation of the interpolation algorithm.

Point A is obtained by retrieving the vehicle position coordinates. An interval is defined by getting its two extreme points A' and A''. Inside the interval, a number of equally spaced points are defined and for each of them, it is obtained the y-coordinate in the interpolated trajectory. The coordinates of the point with a y-coordinate nearest to the one of the original point A are saved and used to obtain the final interpolated point. This algorithm has been implemented to get a final point correctly translated into the

interpolated trajectory. By using directly the x coordinate of point A, a wrong point would have been obtained.
The algorithm is below deeply analysed.

```python
tra_A = ['human_6','human_7','human_8','human_9']
tra_B = ['f_0','f_1','f_8','f_9']
tra_C = ['ai_0','ai_1','ai_2','ai_3','human_10','f_3','f_4']
tra_D = ['f_2','f_5']
tra_E = ['f_6','f_11']
tra_F = ['human_1','human_2','human_3','human_4','human_5','f_7','f_10']
```

Firstly, all vehicles and flows are assigned to their specific trajectories. This is possible since the scenario is deterministic and vehicles' paths are defined a priori.

```python
if scenario['veicoli'][vehicle]['id_tra'] in tra_A:
        if x_input < 0 and y_input > 0:
            IDlane = ':5'
        else:
            IDlane = ':1'
        if x_input < -8 and y_input < 0:
            IDlane = ':5'
```

Every complete path is then entirely built. In the example above it is built the path of a vehicle coming from the upper straight lane and taking the second exit of the roundabout. The point at which the path passes from trajectory 5 to trajectory 1 is included in the section where the two trajectories coincide. The same principle is applied to all other paths.

```python
if IDlane == ":3" or IDlane == ":5" or IDlane == ":2int" or IDlane == ":4int":
        x_generic = y_input
        y_generic = x_input
```

Now the algorithm can really start to obtain the final interpolated point. If the trajectory in which the vehicle is moving is a rotated one, the coordinates got from SUMO are switched.

```python
x_list = np.linspace(max(min(interpola_corsie[IDlane]["x_values"]),x_generic -
    value),min(x_generic + value,max(interpola_corsie[IDlane]["x_values"])),
    num=50, endpoint=True)
y_list = interpola_corsie[IDlane]["interpolata"](x_list)
```

A first interval is defined. It goes from $(x - value)$ to $(x + value)$, where x is the maximum or minimum between the trajectory extremes and the actual SUMO coordinate got from SUMO (i.g. for the lower limit will be considered the maximum and vice-versa for the upper limit) and "value" is a constant equal to 0.1 or to the difference between the x-coordinate given and the minimum of the trajectory. This is done to prevent the obtained interval from being outside the trajectory and thus not being able to proceed with the interpolation of all points.
From this first interpolation, it is obtained the slope of the line linking the points inside the interval.

```
slope, intercept = np.polyfit(x_list,y_list,1)
delta_int = slope
    if IDlane == ":2int" or IDlane == ":4int":
        delta_int = slope/0.78
    if IDlane == ":2" or IDlane == ":4":
        delta_int = slope/1.2
    if IDlane == ":1":
        delta_int = slope*1.4667
```

The slope will be used to obtain the final interval and the final interpolated point. Before using the slope value, it is needed to apply a correction. This correction is crucial to avoid errors when passing from one trajectory to another if one of them has been rotated. The specific correction is obtained by a trial and error approach, looking at the interpolated points along the crossing point.

```
x_list = np.linspace(max(min(interpola_corsie[IDlane]["x_values"]),x_generic -
    abs(delta_int)),min(x_generic + abs(delta_int),
    max(interpola_corsie[IDlane]["x_values"])),num=points, endpoint=True)
y_list = interpola_corsie[IDlane]["interpolata"](x_list)
```

The second interval is, therefore, obtained. One crucial variable is the number of points inside the interval, expressed by the *num* variable. The interpolation will be more precise when more points are used. On the other hand, more points mean a higher computational time. The final value chosen in this thesis work is 120 points. A final third interpolation can be done only when the point given by SUMO is really distant from the trajectory. In this case, the interval must be wider to obtain a correct interpolated point. This never happens in this example, but it protects the algorithm.

```
difference_array = np.absolute(y_list - y_generic)
index = difference_array.argmin()
```

```
y_2 = y_list[index]
x_2 = x_list[index]
x_interp = (x_generic + x_2)/2
y_interp = interpola_corsie[IDlane]["interpolata"](x_interp)
```

Finally, the algorithm obtains the interpolated point by considering the mean x-coordinate value between the input and the point in the interval with the y-coordinate nearest to the input value. The final output coordinates must be switched again to get the correct values, coherent with the position SUMO gave.

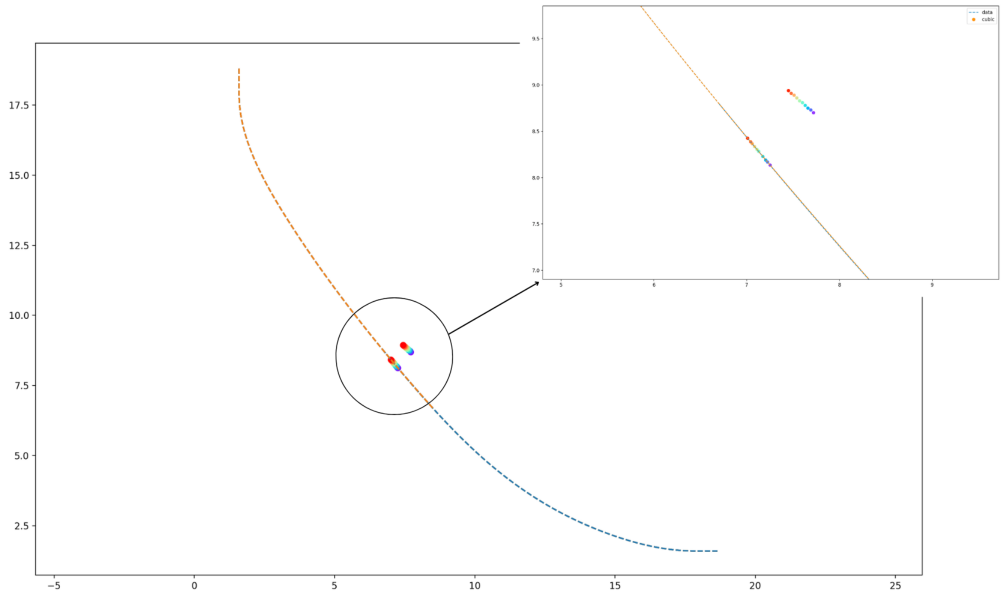


Figure 3.6: Example of the final interpolation over a crossing section.

In figure 3.6 it is presented an example of interpolation over the crossing section between trajectories 4 and 4int. The algorithm developed is able to solve the problem stated at the beginning of this section: trajectories can be completely personalised and adapted to any kind of requirement. Further tests have been done to describe the effect of such a modification. Vehicles move along really realistic paths, which makes them more predictable and human-like. On the other hand, this algorithm is more complex than the first one presented in section 2.4.2 and, therefore, correlated to a higher computational time. The time needed for every step is highly dependent on the number of points inside the intervals. This value, equal to 120 points, has been obtained considering the maximum number of vehicles the algorithm should deal with and the limit imposed by communication and equal to 0.005 seconds. This variable must be adapted to the specific scenario, its geometry and complexity, in terms of legs and vehicles number inside the circulatory roadway.

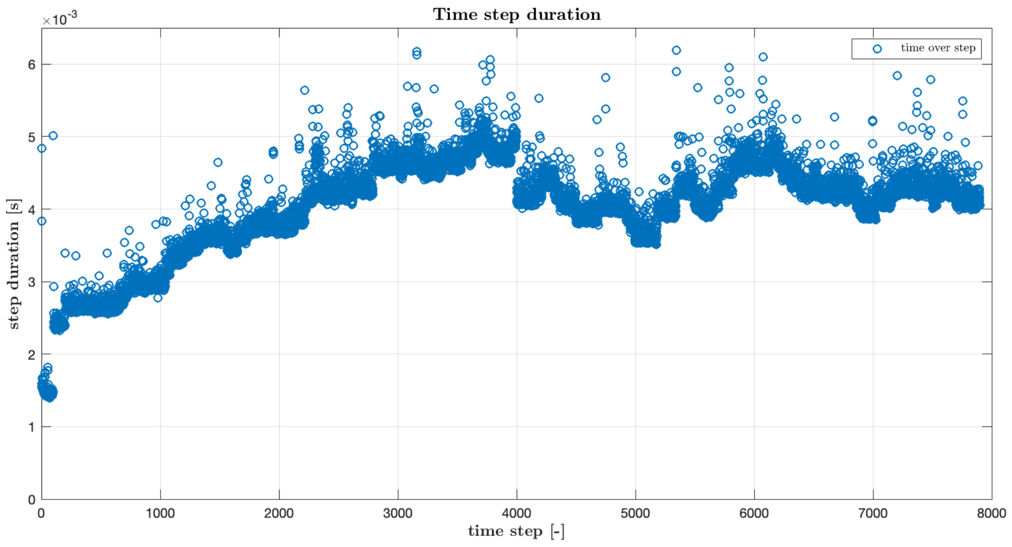


Figure 3.7: Time step computational time over the whole simulation with interpolated trajectories.

Figure 3.7 shows the time step duration during a generic simulation. It is possible to see that communication takes more time than the scenario without interpolating trajectories. Once many vehicles are in the roundabout, the step duration becomes higher. Some out layers can be identified, but they do not introduce any problem into the simulation: the number of points higher than the limit is limited and all other values below 0.005 seconds let the systems recover the delay eventually acquired. The mean step duration is now equal to 0.0040 seconds.

Second interpolation algorithm

The second interpolation algorithm is less precise but faster and, therefore, useful in the case of many vehicles in the simulation. The starting point of this algorithm is the same as the first one. The Python interpolator is used to obtain the interpolated trajectories through the definition of some passing points. Once all paths have been obtained, the x and y coordinates are concatenated to get the final trajectories. In this case, complete trajectories begin from the starting edge and are concluded in the final edge: differently from the first algorithm, they also comprehend the finish and exit straights. This way, they can be completely personalised and no geometrical requirement must be considered with respect to SUMO logic. Concatenation is done as described below.

```
x =
    np.concatenate((path[":1"]["xnew"],path[":2"]["interp"](path[":2"]["xnew"])))
y =
    np.concatenate((path[":1"]["interp"](path[":1"]["xnew"]),path[":2"]["xnew"]))
```

As described in section 3.4 two types of trajectories are defined. If the trajectory needs to be rotated, its x and y will be switched and this should also be taken into account in the concatenation. In this example, path *":2"* is rotated and its y coordinate is concatenated in the x coordinate. All concatenated trajectories can be used to fill the lookup table on which the algorithm is built. This table is composed of three columns: the distance travelled by the vehicle, its x coordinate and its y coordinate. Figure 3.10 shows an example of this lookup table.

Distance [m]	**x -coordinate**	**y-coordinate**
0,0974	91,5436	-95,0135
0,0994	-91,5422	-95,0120
0,1014	-91,5409	-95,0106
0,1034	-91,5395	-95,0090
0,1055	-91,5381	-95,0075
0,1076	-91,5367	-95,0060
0,1097	-91,5352	-95,0044
0,1119	-91,5338	-95,0028
...	...	...

Table 3.10: Example of the lookup table used to obtain interpolated points.

The distance column is built by calculating the distance between subsequent points obtained in the interpolated complete trajectories. During the simulation, through SUMO will be acquired the distance travelled by vehicles in the simulation and this value will be used inside the lookup table, corresponding to the path of the vehicle, to get the final interpolated points. The process is deeply described below.

```
for vei_id in scenario['active_vehicles']:
        distance = string[vei_id]['distance']
        chosen_path = scenario['vehicles'][veh_id]['id_tra']

```

```
xnew = np.interp(distance,
    complete_paths[chosen_path]["distance_column"],
    complete_paths[chosen_path]["x"])
ynew = np.interp(distance,
    complete_paths[chosen_path]["distance_column"],
    complete_paths[chosen_path]["y"])

string[veh_id]['x_interp']= xnew
string[veh_id]['y_interp']= ynew
```

When initialising every vehicle, it is defined an id_{tra} to pick the corresponding lookup table. The numpy interpolator is used to acquire the final x and y interpolated values which will be finally sent to the concurrent. Two problems must be solved in order to obtain a fully functioning algorithm:

1. since the trajectories are modified with respect to the ones visualised in SUMO, the distance travelled by vehicles is different. For this reason it is calculated a multiplication factor which will modify the distance calculated in python to consider this difference. Specifically:

   ```
   if i == ":1":
           if complete_paths[i]["x"][j] <= -15.1999 and
               complete_paths[i]["x"][j] >= -26.6006:
               distance = distance/1.02
   ```

   The multiplication factor, in this case, is equal to 1.02 and, finally, the total distance travelled by the SUMO and the DriveSim vehicles is the same. An error below 5 cm is defined as a reference;

2. the distance must be modified if the vehicle does not start from the beginning of the corresponding starting edge. In this case, in fact, the vehicle needs to be translated to its actual position. This happens only for vehicles inserted in the simulation at the first time interval: all other vehicles will start from the beginning of the edge. To solve this problem, at the first instant of the simulation it is asked the position of every vehicle and it is calculated the distance from the corresponding starting edge. Finally, it is obtained the reference distance value which will be added as a constant to the distance travelled by the vehicle. More in detail:

   ```
   ...
   scenario['vehicles'][veh_id]["init_distance"] = math.sqrt((init_x -
   ```

```
    x_ref)**2 + (init_y - y_ref)**2) - string[veh_id]['distance']
distance = string[veh_id]['distance'] +
    scenario['vehicles'][veh_id]["init_distance"]
chosen_path = scenario['vehicles'][veh_id]['id_tra']
...
```

Where $init_x$ and $init_y$ are the initial position coordinates and x_{ref} and y_{ref} are the coordinates of the corresponding starting edge.

One of the main characteristics that change the computational time is the number of points obtained in the interpolated trajectories and, therefore, the number of rows inside the lookup tables. To reduce as much as possible this value, the number of points for straights is set to 500 and for curves to 5000.
As described initially, this algorithm is much faster than the first one, but it is also less precise in obtaining the interpolated points inside the circulatory roadway. Even if the multiplication factor reduces this error, the first algorithm is more precise and lets obtain better results. For this reason, both of these two algorithms have pros and cons and must be carefully chosen considering the specific simulation conditions, in terms of trajectories complexity and number of vehicles.

Turn signals

So as to achieve an even more impactful result, also turn signals have been added to the environment. Every vehicle can use them to communicate its intentions to other vehicles. They are used inside the roundabout only and every AVs and human-driven vehicle make use of them. To develop this further solution, it is needed to ask SUMO for a new variable using TraCI subscriptions. This variable is called *"Signals"* and contains all vehicle signals information. They are all encoded in an integer, whose value defines which are being used on a binary basis.

Name	Bit
VEH_SIGNAL_BLINKER_RIGHT	0
VEH_SIGNAL_BLINKER_LEFT	1
VEH_SIGNAL_BLINKER_EMERGENCY	2
VEH_SIGNAL_BRAKELIGHT	3
VEH_SIGNAL_FRONTLIGHT	4
VEH_SIGNAL_FOGLIGHT	5
VEH_SIGNAL_HIGHBEAM	6
VEH_SIGNAL_BACKDRIVE	7
VEH_SIGNAL_WIPER	8
VEH_SIGNAL_DOOR_OPEN_LEFT	9
VEH_SIGNAL_DOOR_OPEN_RIGHT	10
VEH_SIGNAL_EMERGENCY_BLUE	11
VEH_SIGNAL_EMERGENCY_RED	12
VEH_SIGNAL_EMERGENCY_YELLOW	13

Table 3.11: List of possible signal values which the user can retrieve using TraCI.

Table 3.11 sums up all possible retrievable signals for a vehicle in SUMO. Among them, only two are relevant for this case study: right blinkers and brake lights. The front lights are imposed always active and as sidelights. Every bit is associated with a power of two. All values are then multiplied by 1 or 0 if the corresponding light is on or off, respectively, and finally, they are all summed up. The final integer value contains all lights data. In this thesis work, the integer can assume four different values:

1. 0, no braking lights or blinkers are on;
2. 1, only blinker right is on;
3. 8, only braking lights are on;
4. 9, braking lights and blinkers are on.

Inside the roundabout, vehicles can only use the right blinker to communicate the intention to exit the circulatory roadway. Therefore, the left blinker is never used. Once the value has been retrieved, it is sent to the concurrent as a float variable and the corresponding lights are switched on. It is not needed for blinkers to send an intermittent *"on"* value, as for a generic CAN bus protocol.
Brake lights are used in SUMO whenever the vehicle is standing (not stopped) or when it

decelerates beyond a threshold value. With regard to blinkers, they are activated in the following conditions:

- the vehicle will take a turn at the next intersection. The activation happens 7 seconds before reaching the intersection;
- a vehicle is about to stop for parking;
- when activating emergency braking lights.

In this case study, it is used only in the first condition. Unfortunately, SUMO considers any turn in the following junction. For this reason, even if the vehicle will not exit at the first subsequent exit lane, it will turn right at the beginning of the intersection, resulting in having the right blinker activated even if it will not turn right. This generates easy incomprehension with other human users, driving the ego car. To avoid this to happen, the right blinker signal is shut off on every straight lane and entry lane. It can be used only on exit lanes. This result is obtained by simply manipulating the signal value retrieved and making it equal to 0 or 8 when the vehicle is not on an exit lane, getting only braking lights activated.
Apart from these technical modifications, it must also be discussed and improved the questionnaire and the way it is proposed to participants. One possible solution would be to ask users to fill in some sections right after reaching checkpoints at the end of simulations (e.g. every time they complete the possible scenarios for the same legs, participants will answer some questions about the simulations they just carried out). This could limit a lot the loss of data due to long and numerous simulations. Further details on the specific tests and associated questionnaire will be addressed in section 5.2.
This completes the detailed description of all the modifications that will be considered in the final network. This will ensure a realistic and effective simulating scenario, able to demonstrate mainly the policy's ability to be used in a roundabout scenario, without losing information due to implementation and development errors in the simulation environment.

4 | Replay

The fourth chapter of this thesis work is devoted to the presentation and deep analysis of *Replay*. It refers to a specific simulation in which participants are placed in a vehicle driven by the AV policy. Inside the driving simulator, the ego car will repeat all actions of a generic AV that run inside a previous simulation. Through the Replay, it is possible to deeply evaluate some basic and crucial policy performances, which are essential to achieve AVs that are truly usable in a realistic environment. DriSMi represents a cutting-edge technology that allows the possibility of implementing a complete and proposable replay to users during experimental testing. This is, moreover, a step forward from other previous studies in this field, which did not have the possibility to use such a powerful driving simulator and did not achieve to obtain results in this field of research. Replay has been used in the preliminary test just to develop its final configuration and to understand the policy's main criticalities. It will be crucial for the final simulation environment, providing an important source of data crucial to address completely the policy performances inside the roundabout scenario and extending the questionnaire capabilities.
The key challenges in implementing a working Replay are related to the communication between the various simulators used, and the data that each of them needs and instead produces during the simulation. Not least, it is also necessary to consider the vehicle model required by DriSMi and that which can be implemented with the data held by SUMO: data must be complete and they must fulfil the requirements needed to move all DriSMi actuators and electrical motors, replicating the movement of the chosen AV.
In this chapter, all of these characteristics will be analysed in detail. Starting from the data obtained by SUMO and getting to the final working replay scenario.

4.1. Replay model configuration

First of all, the replay model must be defined. It means the definition of the system used for obtaining the final experiment and comprises the definition of all software and strategies implemented to define the vehicle model used in the driving simulation to move all actuators and electrical motors. The chosen software must be able to represent a vehicle model complex enough to obtain at each instant of time all the necessary information, such as forces, displacements or acceleration and braking intensities. The origin of the data is SUMO, which gives a limited amount of information. As stated in the previous sections, SUMO represents a vehicle as a single point for which the following relevant characteristics can be retrieved:

- x and y coordinates, defining its position;
- longitudinal speed;
- longitudinal acceleration.

This information is not enough even to describe a simple bicycle model. Moreover, the quality of these data is low, as shown in figure 2.13. Finally, it is also crucial to consider the sampling time with which data are collected. In SUMO, this value is equal to 0.005 seconds, as imposed in the design phase. The driving simulator needs the same data every 0.001 seconds to correctly move the vehicle during the Replay simulation. Data cannot be collected in DriSMi during a generic simulation, since it is used only as a graphical tool, as described in section 2.3. To solve these problems, a third software must be used and all other essential data must be obtained.
The first considered solution was to use the *Bicycle model* in Simulink. It is a block which implements a rigid two-axle single-track vehicle body model to calculate longitudinal, lateral, and yaw motion.

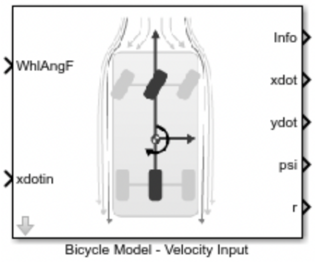


Figure 4.1: Bicycle model - velocity input in Simulink.

This model needs the steering angle and speed at every time step of the simulation. The speed is made available by SUMO. The steering angle can be obtained using Terminal 2 by considering simple cinematic relations. This solution, although implementable, does not return a result with a physical meaning. The bicycle model block does not take into account any tire model or any other model for all vehicle components, such as suspensions. For this reason, this solution cannot be accepted and a more complex configuration must be used, as described below. In this thesis work, two different paths have been considered:

1. using Simulink;
2. using VI-CarRealTime.

In the first proposed solution, all available data are obtained for a generic AV in the SUMO simulation. These data are used in a Simulink simulation to make a driver follow the given trajectory at the given longitudinal speed, considering these values imposed on the centre of gravity of the vehicle.

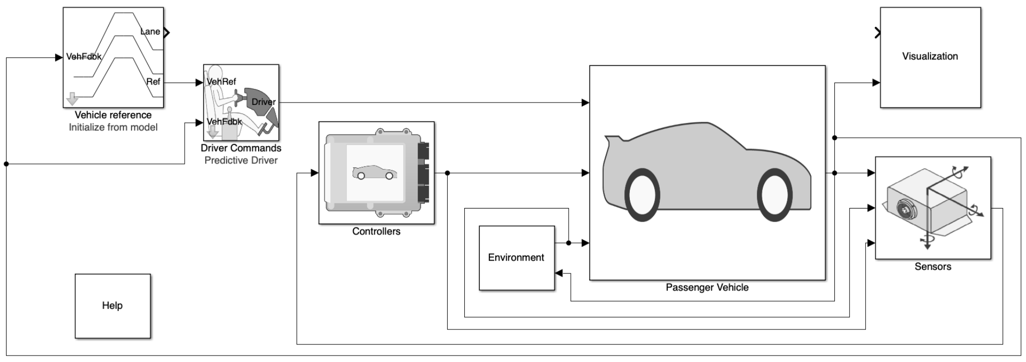


Figure 4.2: Simulink model used for first replay solution.

Figure 4.2 shows the initial feedback control Simulink model used for the Replay simulation. The model is obtained from a modification of the *Double Lane Change Reference Application* given by Matlab. The complete reference is composed both by the trajectory, in terms of x and y coordinates given by SUMO, and the longitudinal speed. The vehicle model can be chosen between a 7 or a 14 degrees of freedom one. The driver model is responsible for the steering, braking and accelerating controls. The visualisation is not relevant and can be discarded. Either way, for Replay the computational time is not relevant, since only the final data will be sent to the driving simulator. Many controllers have been implemented in this model: not only the ones on the acceleration and braking actions but also others responsible for the engine speed and the gearbox. The environment block describes the road on which the vehicle will move (e.g. its friction constant value) and the wind characteristics if present. This block is not necessary for this case study since no wind scenario is considered. Sensors are used to measure all relevant properties of the vehicle and implement the feedback control loop.

Once this complete model has been developed, data acquired by SUMO have been used to test this configuration. Unfortunately, Simulink is not able to follow trajectories in which the vehicle reference speed goes to 0. In this specific scenario, the tyres model returns a singularity and the simulation is immediately stopped. For this reason, this first solution cannot be used.
The second solution coincides with the use of VI-CarRealTime as a third-party software in the configuration. VI-CarRealTime could also be used to obtain some of the variables needed by Simulink to work and substitute the blocks, which gave the error described before. This case has not been considered to avoid further compatibility problems. VI-CarRealTime can be used to implement a complex vehicle model and a driver who will follow the reference trajectory. This is the solution adopted to obtain the final Replay configuration. For this reason, the following section will deeply analyse the vehicle model developed in VI-CarRealTime.

4.1.1. VI-CarRealTime configuration characteristics

This section presents in detail the configuration used in VI-CarRealTime to implement the Replay simulation. It will be discussed its characteristics and the main features of the models used in this software, such as the vehicle and its main components.
VI-CarRealTime is used to obtain the final set of data required to properly use the driving simulator. Three are the main actors of the configurations obtained:

1. SUMO simulation;
2. VI-CarRealTime;
3. DriSMi.

SUMO is the first simulation environment in which a generic simulation is carried out. At the end of the simulation, all data referred to an AV are collected and stored to be used in the following phases. This data can be split into two main components: the reference path and the reference longitudinal speed. VI-CarRealTime is the second piece of the model implemented. The references are given to VI-CarRealTime differently:

- the reference path is a positional reference. It is created a table containing all the subsequent x and y coordinates couples, creating a complete path to be followed. This path is also used to create the visual street that the vehicle will follow in the VI-CarRealTime simulation;
- the speed is given as a time reference. It is not expressed with respect to a specific position, but with respect to a time interval. Specifically, every 0.005 seconds, which

is the sampling time in SUMO, a speed value is given to the driver who will try to follow it as best as possible.

This way, a complete reference state has been created. In VI-CarRealTime it will be implemented a driver who will follow both the speed and position references, replicating the AV movement along all the simulations. The result is a copy of the AV chosen, from which all the information required can be obtained. The driving simulator software and VI-CarRealTime are designed by the same company and this results in complete compatibility. Inside VI-CarRealTime also the sampling time can be chosen and it is set to 0.001 seconds, as required by DriSMi.
In this way, it has been created a complete and closed system able to build the Replay scenario. The driver will follow the reference given and will produce a complete output, which will finally be used in the driving simulator to move the ego car in which passengers will experience the AV guide capabilities.

4.1.2. VI-CarRealTime vehicle model

The vehicle model used in VI-CarRealTime is really complex. A generic four-wheeled vehicle is a model with 14 degrees of freedom and 5 rigid parts: the vehicle chassis (i.g. the sprung mass) and the 4 wheel parts (i.g. the unsprung masses). The suspensions and steering systems are conceptual: with respect to the first one, no linkages or bushings are considered; the second one has no parts for the steering wheel or for the rack. All their properties are described by means of lookup tables. The sprung mass can be rigid or compliant. The unsprung masses define 2 degrees of freedom for every wheel: the vertical travel and the wheel rotation.

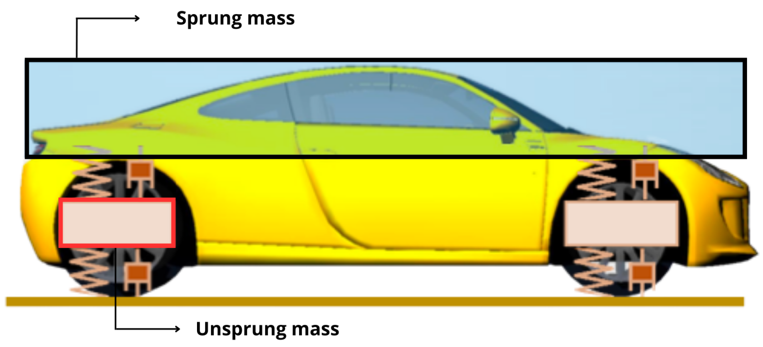


Figure 4.3: Vehicle sprung and unsprung masses in VI-CarRealTime.

If the sprung mass is compliant, the compliances supported are up to 6 DOF. Considering

tires, the number of states depends on the specific tire model used, which in this case study is the Paceijka, described by two states.
The vehicle model is described in terms of commands and functions in a symbolic manipulator adapted to the derivation of multibody equations. The equations of motion are analyzed through a code generator that solves the differential equations of state in explicit form. Finally, numerical integrators are used to obtain the numerical solution. In this thesis work, Runge-Kutta has been considered.
References are important and must be considered with respect to the ones of the driving simulator, in order to obtain the final vehicle in Replay mode and the one in VI-CarRealTime moving in the same direction. Inside VI-CarRealTime two different reference frames are used:

- a global reference frame, which in this case has its origin in $(0, 0, 0)$;
- a local reference frame having the origin in the vehicle's midpoint of its front tires contact patches.

Every wheel has its own reference system, positioned at the wheel centre, with the same orientation as the global reference frame. Finally, also the driver location reference system is considered. Its origin is placed on the rear axle midpoint and the orientation is the same as the local reference frame.

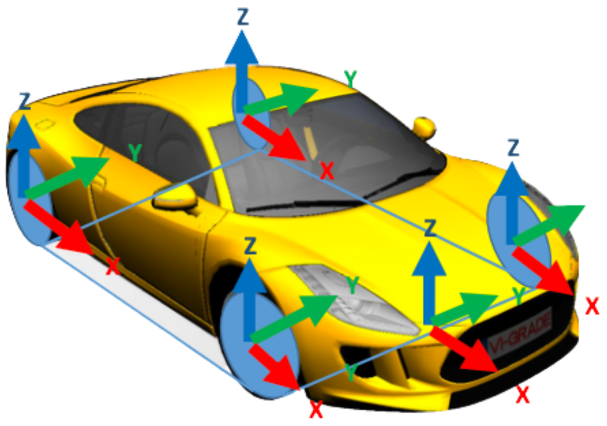


Figure 4.4: Vehicle reference frames. Only the local reference frame and the wheel ones are considered.

Just one sensor point is considered. The sensor will monitor position, velocities and accelerations during the simulation. These values will be used by the driver to follow the reference speed given by the user. All measure quantities are obtained with respect to the global reference frame.
Specifically, the vehicle model used is the Compact car. This model is entirely pre-built

inside VI-CarRealTime and all its characteristics are already defined. Therefore, it is not needed to design all its main components, such as the steering or the suspension systems. This is done since the specific vehicle considered has no relevance: SUMO defines a vehicle through some parameters and almost no physical properties are considered among them. The only important aspect is that the car chosen must be able to follow the speed profile correctly with respect to the reference path and it must be equal to the car used in the driving simulator. Between all saved data, also vehicle complete models and characteristics are saved and given to DriSMi during the final Replay scenario. The following modifications have been considered for the vehicle compact car model:

- the clutch has been substituted by a torque converter. It is much more smooth during gear shifting and ensures better behaviour of the driver during the simulation;
- the transmission has been set to automatic. This was also done to improve the behaviour of the vehicle and make it easier to drive.

The engine is an internal combustion engine. These final characteristics have been chosen to avoid possible interferences during the simulation due to specific vehicle physical properties.

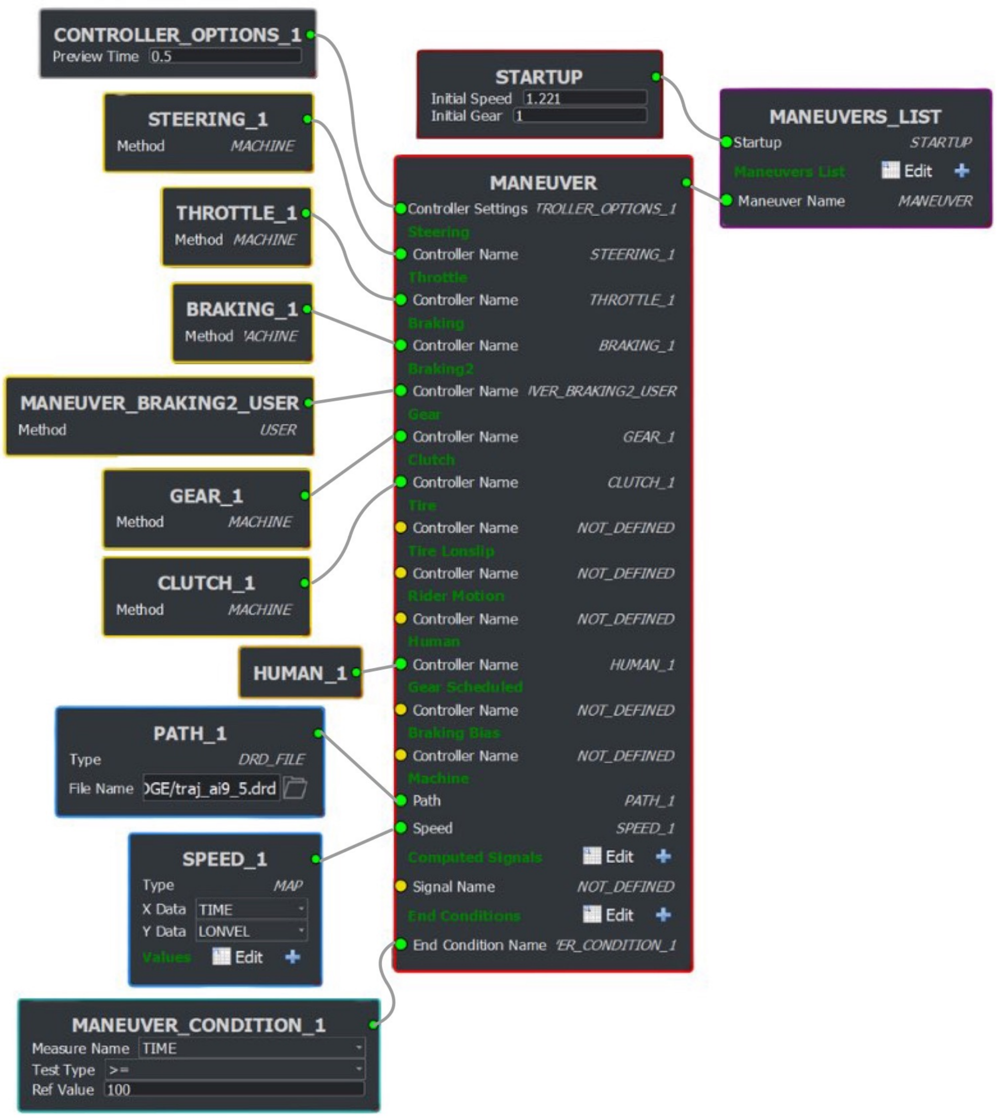
4.2. Replay event implementation

This second section is devoted to the detailed description of the model obtained in VI-CarRealTime. It is presented a comprehensive analysis of the vehicle and the driver's characteristics. Moreover, it is described how all the requirements can be satisfied inside the VI-CarRealTime environment. The vehicle chosen, as cited in section 4.1.2, is a standard compact car. Its main properties are listed in table 4.1.

Property	**Value**	**Unit of measure**
Unladen mass	1383.7	[kg]
Wheelbase	2577.4	[mm]
CG longitudinal position	1021.6	[mm]
CG lateral position	0.0114	[mm]
CG height	563.92	[mm]
Ixx	381956960.0	$[kg \cdot m^2]$
Iyy	1251656000.0	$[kg \cdot m^2]$
Izz	1217101200.0	$[kg \cdot m^2]$

Table 4.1: Main compact car characteristics.

All CG coordinates are defined with respect to the local reference system of the vehicle. The sensor point is positioned at $(-1024.4, 0.1159, 515.43)$. All the other vehicle's properties are preset by VI-CarRealTime and not changed. Once the vehicle has been completely defined, it is possible to create the model responsible for all the replay simulations. The VI-CarRealTime event builder is used, starting from a *FileDriven* event. The complete event is presented in figure 4.5.


CONTROLLER_OPTIONS_1
Preview Time 0.5

STEERING_1
Method MACHINE

THROTTLE_1
Method MACHINE

BRAKING_1
Method 'ACHINE

MANEUVER_BRAKING2_USER
Method USER

GEAR_1
Method MACHINE

CLUTCH_1
Method MACHINE

HUMAN_1

PATH_1
Type DRD_FILE
File Name)GE/traj_ai9_5.drd

SPEED_1
Type MAP
X Data TIME
Y Data LONVEL
Values Edit

MANEUVER_CONDITION_1
Measure Name TIME
Test Type >=
Ref Value 100

STARTUP
Initial Speed 1.221
Initial Gear 1

MANEUVERS_LIST
Startup STARTUP
Maneuvers List Edit
Maneuver Name MANEUVER

MANEUVER
Controller Settings TROLLER_OPTIONS_1
Steering
Controller Name STEERING_1
Throttle
Controller Name THROTTLE_1
Braking
Controller Name BRAKING_1
Braking2
Controller Name IVER_BRAKING2_USER
Gear
Controller Name GEAR_1
Clutch
Controller Name CLUTCH_1
Tire
Controller Name NOT_DEFINED
Tire Lonslip
Controller Name NOT_DEFINED
Rider Motion
Controller Name NOT_DEFINED
Human
Controller Name HUMAN_1
Gear Scheduled
Controller Name NOT_DEFINED
Braking Bias
Controller Name NOT_DEFINED
Machine
Path PATH_1
Speed SPEED_1
Computed Signals Edit
Signal Name NOT_DEFINED
End Conditions Edit
End Condition Name 'ER_CONDITION_1

Figure 4.5: VI-CarRealTime event for the replay simulation.

The event is described by a main manoeuvre block, to which all possible sub-blocks can be linked. All actions during the simulation are controlled by the driver. This is set by choosing yellow blocks with the *Machine* method. As it can be seen from figure 4.5, steering, throttle, braking and gear shifting are referred to this method. Inside each sub-

block, all action properties can be defined, such as smoothing or smoothing time. The controller options block is responsible for the main characteristics of the controller used to move the vehicle. The user can freely set the preview time, the preview length or the delay between the imposed control action and the vehicle response. The human block is used to define the driver skills from novice to robot, which will influence its capabilities to follow the imposed path. In this case study, both path and speed are imposed to obtain the reference trajectory got from SUMO. These two references are crucial to achieving a perfect replication of what the vehicle does in SUMO. The path is imposed as consecutive pairs of coordinates which the driver will follow. The speed is given with respect to the time instant at which the vehicle must reach the reference. Both for speed and path a cubic spline interpolation is used. The acceleration has not been considered, since SUMO is not able to precisely calculate it. The reference path is given to the event creating a *.drd* file, in which all x and y coordinates, the width of the lane and time instants are collected. The reference speed is directly imported in the speed block, defining the two columns which contain the time instants and the speed values in $[m/s]$. The manoeuvre block defines some basic simulation properties, such as the time at which it will stop. Finally, the startup block is used to set the initial conditions, in terms of speed and gear. These values are obtained from the SUMO output.
The event obtained is loaded by VI-CarRealTime before the start of the simulation. Additional parameters to be defined are the initial position of the vehicle, the integration time step and the output time step. To run the final simulation, a *Road Data File* is also obtained, which is responsible for the visual representation of the trajectory. As output files of the simulation, two are required:

- the *.res* file, which contains all simulation details needed by the driving simulator to correctly move the vehicle in the Replay scenario;
- the *.mat* file, which is used to acquire all data and propose a detailed analysis, comparing them to the reference scenario.

The scenario is, therefore, ready to be used to implement the replay simulation. All blocks' properties must be made explicit to have the complete test setup and some vehicle's characteristics must be further analysed.

4.3. Test setup

This section describes the complete test setup of the replay simulation scenario. The setup includes all event properties described in section 4.2 and all the vehicle's subsystems characteristics. Moreover, it also covers all initial conditions to start the simulation and

the specific vehicle chosen to run the test.
Among all vehicles and scenarios considered, it has been taken into account an AV driving inside the 80% of AVs scenario. Specifically, vehicle ai_9 has been chosen. It starts from edge -*E2* and takes the second exit of the roundabout, travelling along edge *E1*, as visually described in figure 4.6.

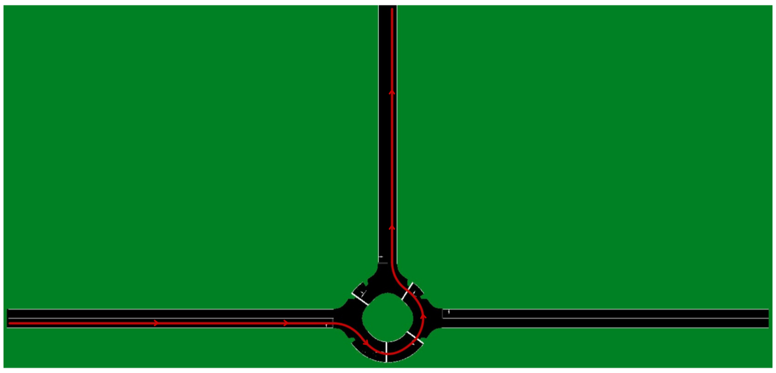
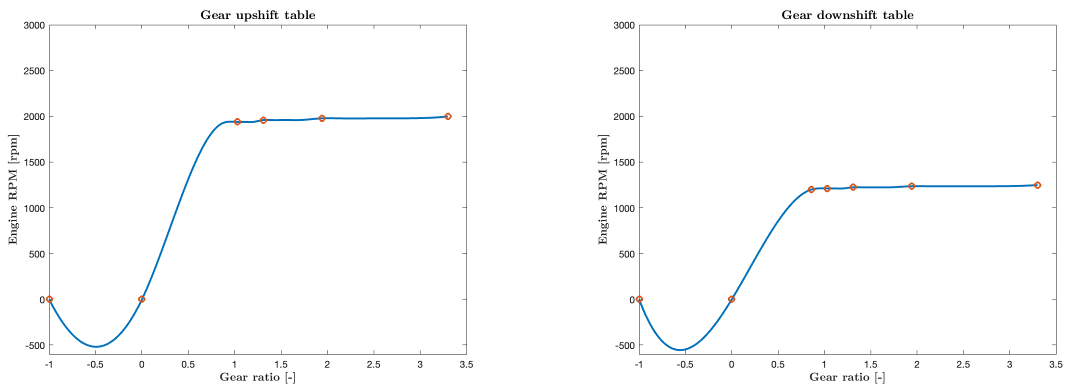


Figure 4.6: Trajectory of vehicle ai_9.

The vehicle starts from point A $(-66.177, -1.6000)$. It is important to consider that the position given by SUMO and the one used by VI-CarRealTime are different: SUMO considers the front axle middle point as the vehicle position, while VI-CarRealTime uses the middle point of the rear axle. For this reason, it is necessary to modify the list of positions given to VI-CarRealTime, considering this translation. The VI-CarRealTime position is also the one considered by the driver to account for the path following and to obtain the error with respect to the trajectory to be followed. To disregard this translation would be to add an error during the simulation and get a driver unable to follow the given trajectory correctly. With respect to the initial speed, this value is obtained by SUMO and it is equal to 1.225 m/s.
The blocks presented in section 4.2 are defined as follows:

- Steering: the smoothing action is activated. When it is enabled a step function is applied to the computed channel value (both open loop and machine) in order to guarantee signal continuity at each manoeuvre startup. The smoothing time is set to 0.1 seconds;
- Braking: initially, it has been considered a braking action from the max possible value to the minimum one. This led to incorrect behaviour of the driver, resulting in a peak braking demand and in a corresponding peak longitudinal acceleration. For this reason, it has been set from 0 to 21%. The scaling factor is equal to 100;

- Gear: the clutch is automatic and the gear shifting mode is set to *comfort*. Furthermore, engine RPMs are used by the driver to understand if up shifting or down shifting. Initially, the gearbox ratios have not been modified;



(a) Gear upshift table. (b) Gear downshift table.

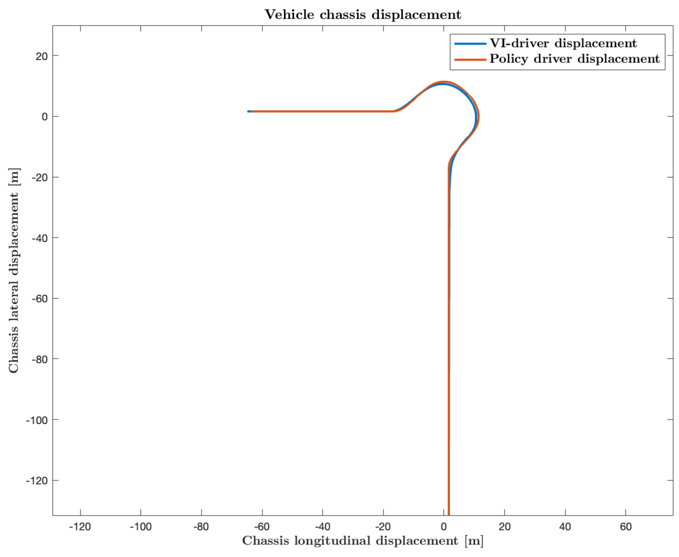
Figure 4.7: Gear upshift and downshift tables used by the driver.

- Driver: it is chosen a professional driver, both to have human and effective driving behaviour;
- Controller: the preview time is set to 0.5 seconds and the minimum preview distance to 0.5 meters. The effect of increasing the preview time is that the controller will produce smoother control actions on the steering but on the other hand the path tracking accuracy will be reduced. Since the main objective is the reference path, it has been chosen a low value of preview time, preserving the path accuracy over the steering action. The longitudinal controller prediction model is enabled. No delay between the imposed control action and the vehicle response is considered.

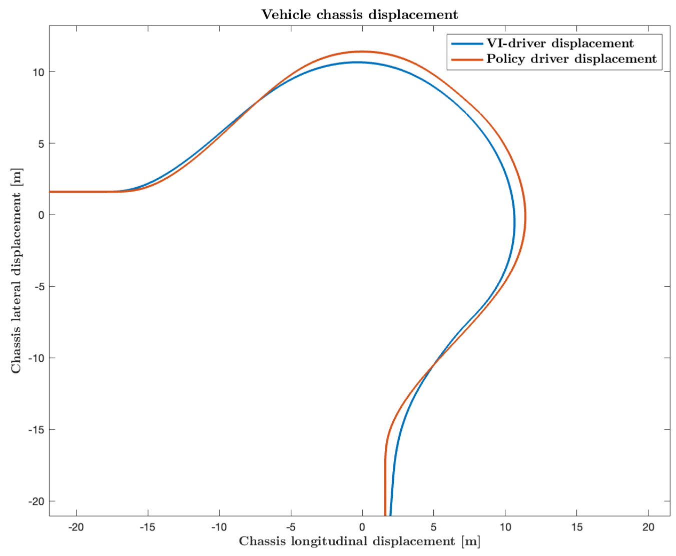
The setup has been completely described. The first replay test can be conducted and they can be used to acquire the first results and to verify the effectiveness of this method. These results will also be crucial to deeply understand AVs' behaviour during the simulation and highlight errors in the policy training.

4.4. Results from the preliminary environment

This section presents the first results obtained from the test on the preliminary environment. The results come from the test setup previously described. Starting from them, the modifications applied will be described to get the final and optimised replay scenario. The results will be commented on from the policy point of view too, emphasising its positive and negative behaviours.



(a) AV displacement comparison.



(b) AV displacement comparison with detail on the circulatory roadway.

Figure 4.8: AV displacement comparison between SUMO and VI-CarRealTime trajectories.

Looking at figure 4.8 it is possible to see the comparison between the vehicle displacement in SUMO driven by the policy and the one resulting from the VI-CarRealTime simulation. Figure 4.8b shows the detail on the circulatory roadway: a big error is obtained on the path follower. The driver in VI-CarRealTime is not able to correctly follow the reference trajectory. This cannot be the responsibility of the software used, since it is generally employed in relevant competitive fields and, therefore, it is built to be able to follow complex paths and speed profiles. The reason for this error must be deeply analysed and also policy behaviour must be considered.

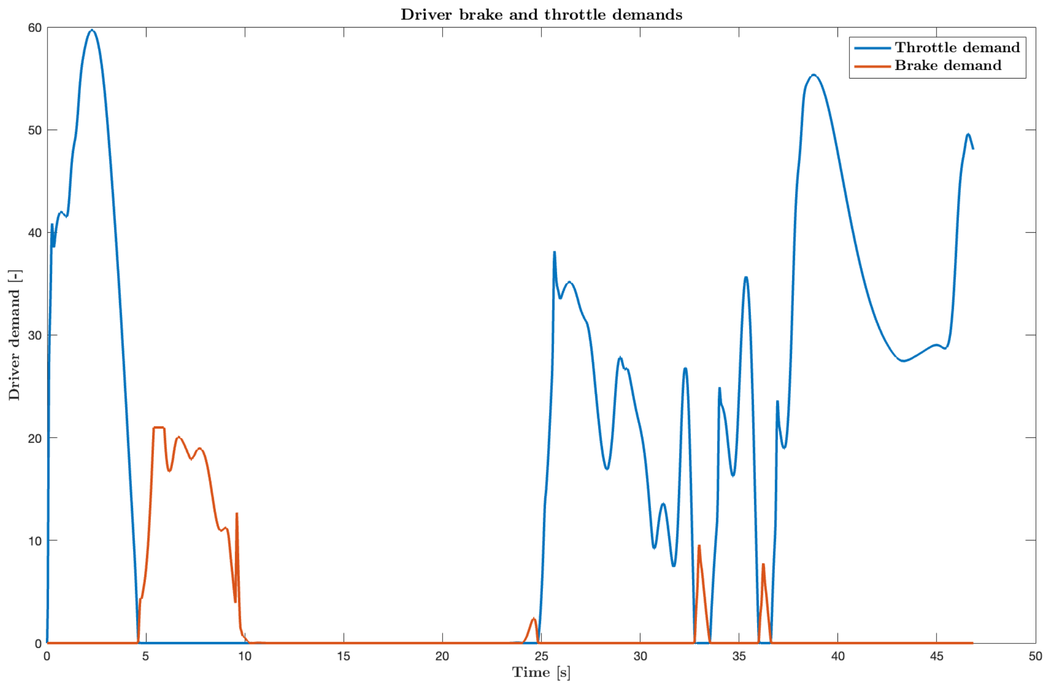


Figure 4.9: VI-CarRealTime driver throttle and braking demands.

Initial considerations are made on the driver braking and throttle demands. Figure 4.9 exhibits the time history of these values: both of them are coherent with the path and speed reference values. The braking action has correctly its maximum at 21% and this ensures no peaks in the longitudinal acceleration, making the vehicle more comfortable for passengers. The throttle demand shows a large variance, which may lower the general comfort.

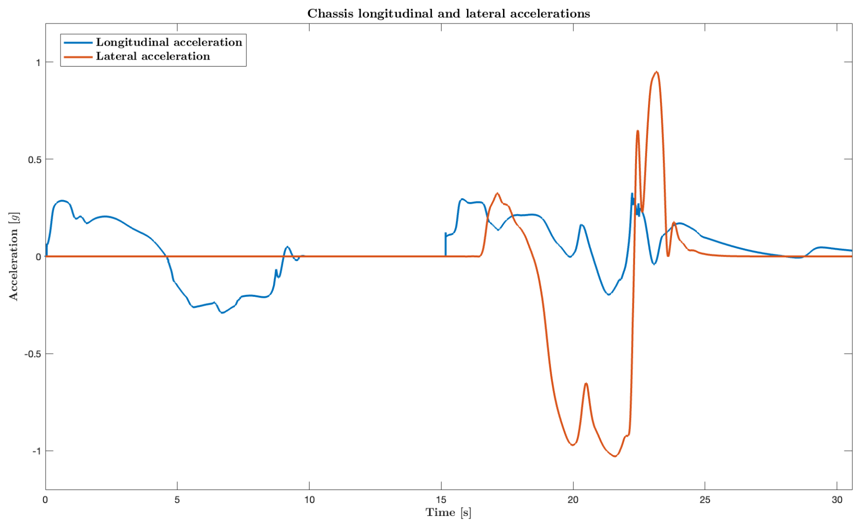


Figure 4.10: Chassis accelerations during the replay simulation.

The chassis accelerations highlight the first problem of policy behaviour. The lateral acceleration has a maximum equal to 0.950 g and a minimum equal to -1.029 g. These values are high and result in general high discomfort. Moreover, lateral acceleration also varies really fast, especially when the vehicle enters and exits the circulatory roadway.

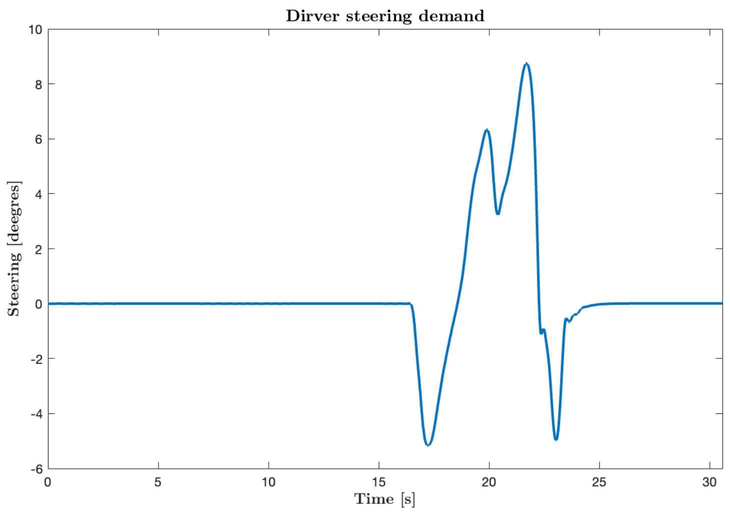


Figure 4.11: Driver steering demands during the replay simulation.

Figure 4.11 is coherent with what has been obtained from the lateral acceleration. The steering demand shows a problem especially when the vehicle exits the roundabout. The driver is not able to follow the path and this results in the wrong behaviour of the steering, which is directly linked to the high and variant lateral acceleration values. This happens for two main reasons:

1. the vehicle travels at a too-high longitudinal speed. The value inside the roundabout ranges from 16.62 km/h to 39.87 km/h. Since the only policy objective is to minimise the travel time, it will try to reach always the maximum allowed speed on every edge of the network, which in this preliminary scenario is set to 40 km/h. No consideration is done for the comfort or the safety of the vehicle. This must be changed by modifying the policy optimisation and objectives;

2. the upshifting and downshifting tables are not well defined and the driver continues to change gears.

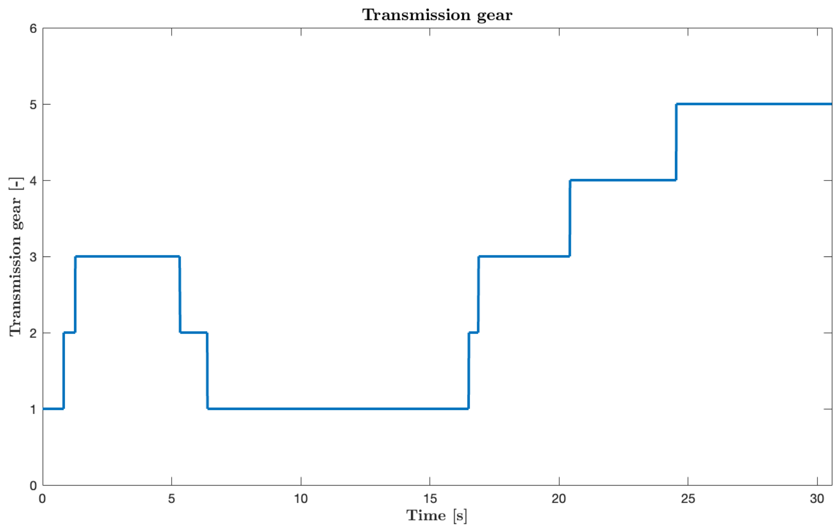


Figure 4.12: Transmissions gear during the replay simulation.

Inside the circulatory roadway, the driver changes gear three times, resulting in much more complexity in following the reference path.

The first problem can be solved in these preliminary tests, by modifying the maximum speed value inside the circulatory roadway. This way it is not needed to train again the policy to understand if this modification affects the driver's behaviour. The limit value for the lateral acceleration is set considering the literature [41] to 0.5 g and the corresponding maximum speed is equal to 19.8 km/h. With regard to the second problem, the two upshifting and downshifting tables are modified and the new ones are presented below.

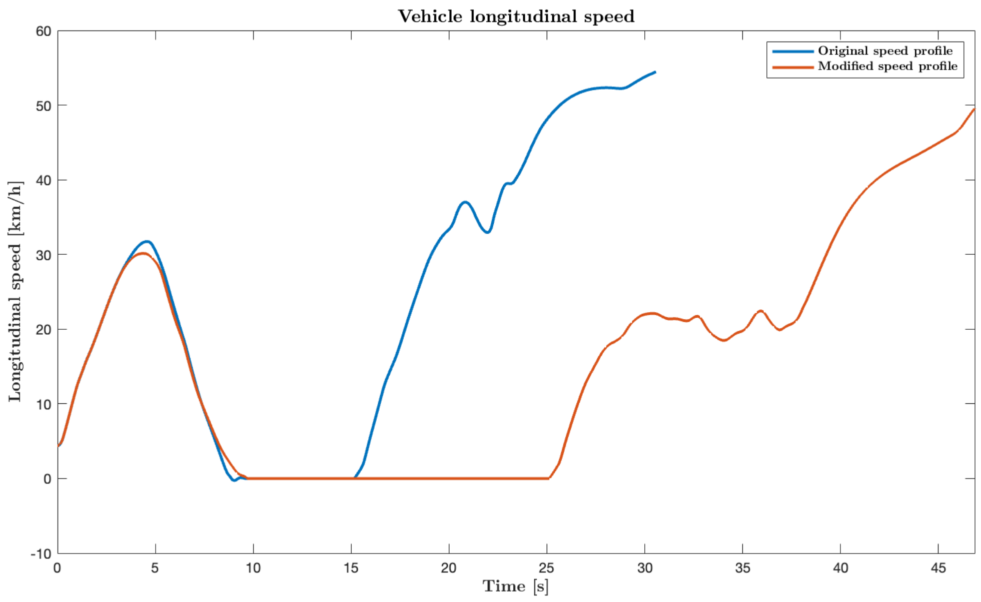
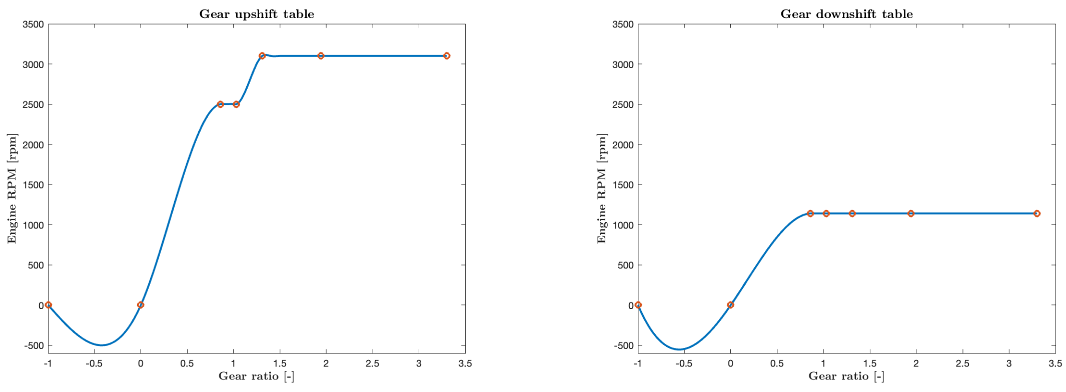


Figure 4.13: original and modified speed profile during the simulation.

Looking at figure 4.13, it is possible to see that the speed is now correctly locked at the limit imposed inside the roundabout. The policy is not able to perfectly maintain the speed limit since it has not been trained to do that. Regardless, the speed effect is clear and the final policy must and will be trained to limit the lateral acceleration, resulting in lower longitudinal velocities and higher comfort.



(a) Modified gear upshift table.

(b) Modified gear downshift table.

Figure 4.14: Modified gear shift tables.

The new gear shift tables completely modify the driver gear selection during the simulation, making it more realistic and constant.

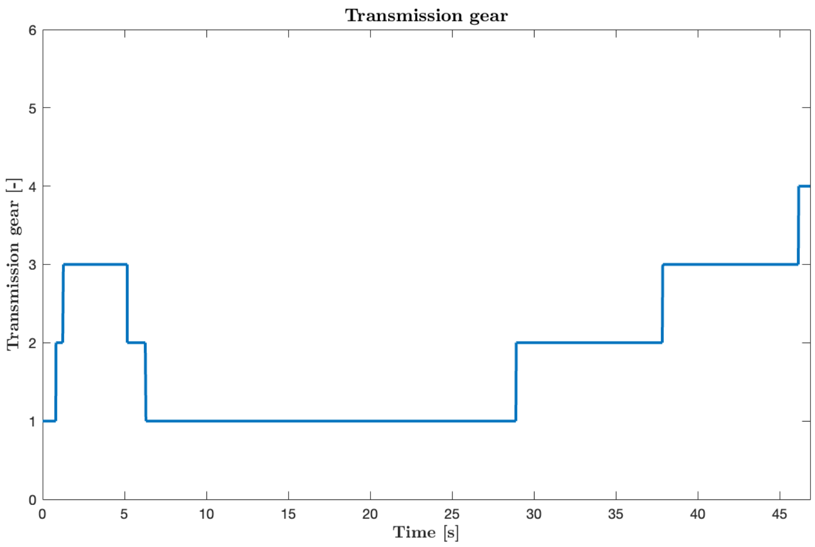
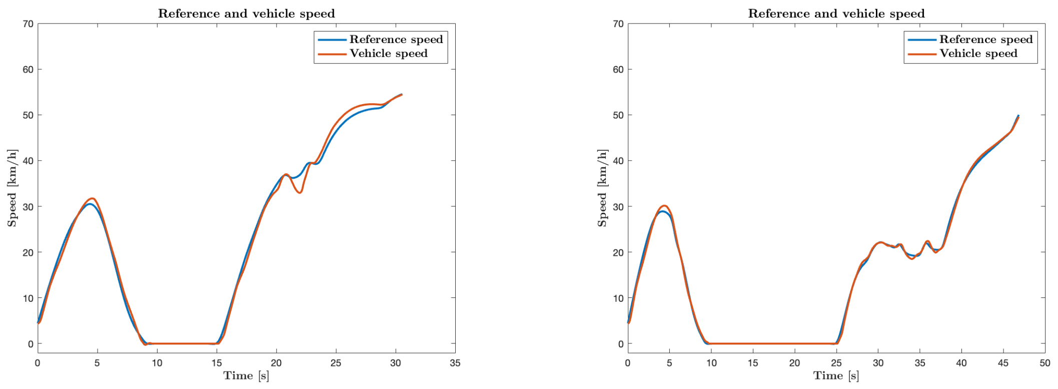


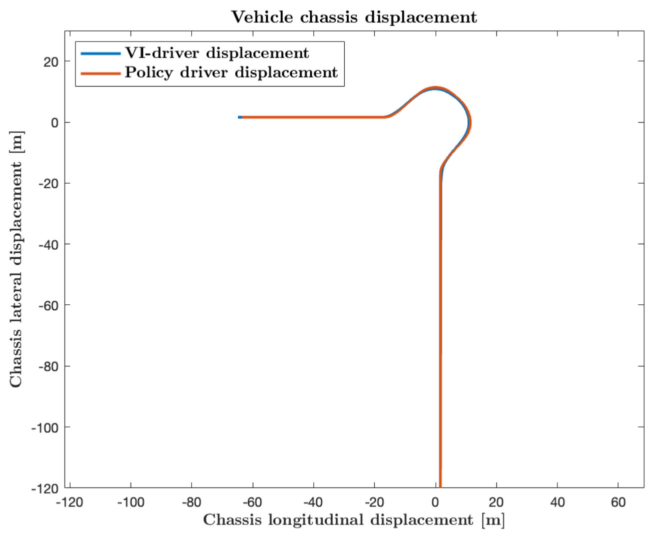
Figure 4.15: Modified transmission gear during the replay simulation.

Now the vehicle changes gear when entering the roundabout and maintains the second gear till it exits the circulatory roadway. The time spent to complete the simulation is different between the original and the modified scenarios since the maximum speed has been changed as previously described. Therefore, the traffic inside the roundabout changes and the vehicle ai_9 must wait more before entering the circulatory roadway. Since the driver does not change gears in the roundabout, it will be more able to follow both the path and the speed references.

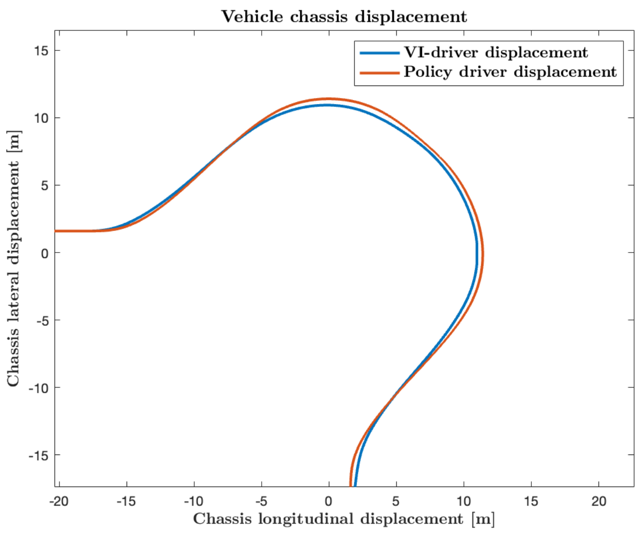


(a) Initial scenario speed comarison. (b) Modified scenario speed comarison.

Figure 4.16: Comaprison between the initial and modified scenario in terms of the difference between the reference and the vehicle speed.



(a) AV displacement comparison for the modified scenario.



(b) AV displacement comparison for the modified scenario with detail on the circulatory roadway.

Figure 4.17: AV displacement comparison between SUMO and VI-CarRealTime trajectories.

As can be seen from figures 4.16 and 4.17 the driver is much more able to follow the reference and, therefore, the initial hypothesis that the error was not caused by the software has been proved. A small difference is still present due to the fact that the AV does not behave as a normal or professional driver would. To completely solve this problem, it is necessary to train again the policy using a much more complex optimisation algorithm which needs to consider many more factors, apart from the travel time inside the roundabout. This will be done for the final simulation scenario.

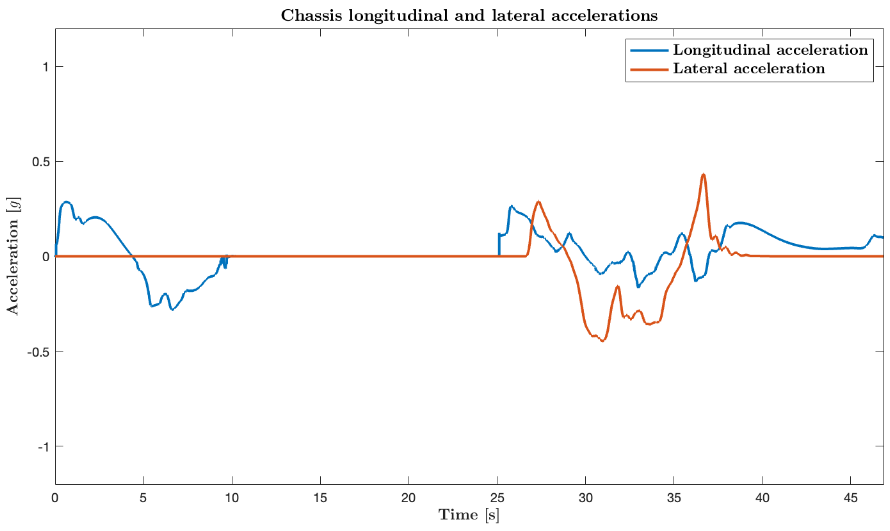


Figure 4.18: Longitudinal and lateral accelerations for the modified scenario.

Finally, it is possible to analyse the longitudinal and lateral accelerations for the modified scenario. They are both much lower with a maximum lateral acceleration equal to 0.435 g and a minimum equal to -0.446 g and, therefore, between the range required. They also change with a much lower ratio, making the vehicle more comfortable.
This final modified scenario has been tested inside the driving simulator, to account for the first impressions and feelings of a passenger inside the AV built. Overall the impression has been positive and the passenger managed to complete several tests. Some discomfort has been denoted, mainly while the vehicle was exiting the roundabout. A huge difference with respect to the original replay simulation has been highlighted by the passenger, who did not succeed to manage more than one test due to the high lateral acceleration and the steering ratio during the replay simulation. The scenario is completed and can be optimised for further simulation environments, making the complete test ready to be used for follow-ups.

5 | Final simulation environment

This chapter summarizes what has been obtained in chapters 2 and 4. It aims to describe the final and complete simulation environment, in which all experimental tests can be carried out. The results of these tests will be crucial not only to verify all design choices but also to validate the policy and its ability to drive inside the roundabout. For this reason, the roundabout must be carefully chosen and designed, starting from real scenarios. Its traffic must be measured too, for the purpose of obtaining a quasi-real simulation, increasing the reliability of data.

5.1. The roundabout

The first and central element of the simulation is the roundabout. The collaboration with the Mobility and Transportation Laboratory of the Department of Design of PoliMi has been necessary to obtain a realistic roundabout design. The following roundabout has been chosen:

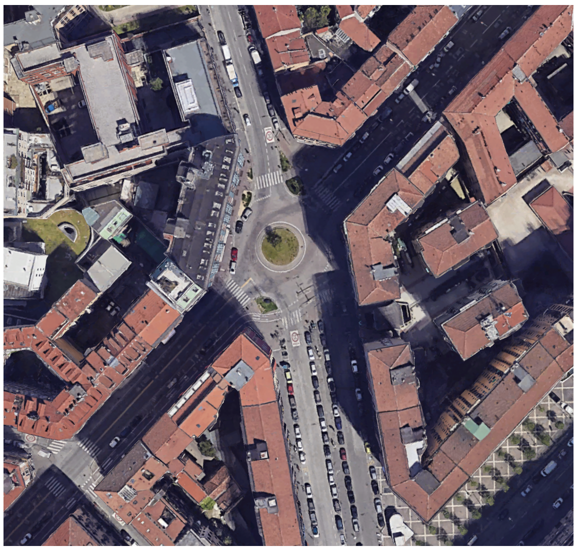


Figure 5.1: The roundabout chosen as a final simulation environment.

This is a four-leg roundabout in Milan. It has just one internal lane and it can be considered a mini-roundabout, as the one used in the preliminary tests. It has been chosen since it shows medium-high traffic, being it a challenging environment for the AV policy. Moreover, it has some important details, which make the simulation more realistic and general. Specifically:

- every leg has pedestrian crosswalks immediately before the entrance of vehicles inside the circulatory roadway;
- two of the legs are central arteries of the city, greatly increasing traffic on the roundabout;
- the roundabout has a standard configuration widely distributed in Europe with significant flows.

The complete structure of the roundabout is shown below.

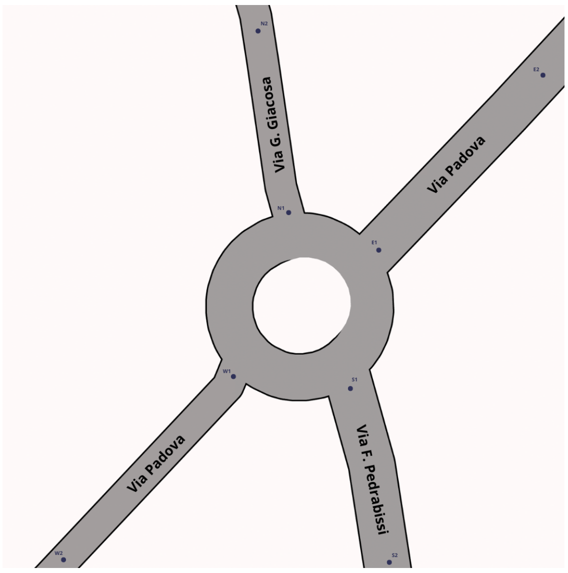
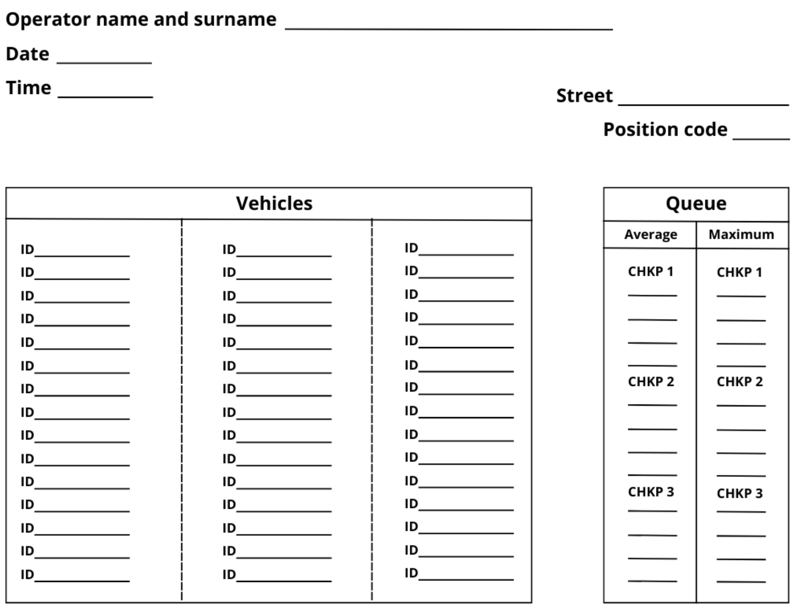


Figure 5.2: The roundabout structure.

Two points are defined for every leg. These points will be used to calibrate the traffic for the simulation environment. Each leg is associated with a letter (i.g. S, E, N, W), which will also be used to name the edges inside the SUMO network. The final network will not have any restricted lanes or pedestrian crosswalks. They are used only to correctly calibrate the traffic. This is done since it is not within the scope of the AI@EDGE project. They could be considered in further phases, adding complexity to the case study. Both SUMO and the driving simulator can deal with pedestrians and specific types of special vehicles, such as the ones of public transportation, creating associated restricted lanes or bus stops.

5.1.1. Roundabout traffic calibration

Once the roundabout has been chosen, it is possible to calibrate the traffic, getting as a final output the number of vehicles and their position during the simulation at the driving simulator. The methodology to complete this process has been given by professors and researchers at the Mobility and Transportation Laboratory of the Department of Design of PoliMi. The first stage is measurement. A group of eight people is used: two per leg, one downstream and one upstream, linked to the eight points obtained in figure 5.2 (i.g. S1, S2, E1, E2, N1, N2, W1, W2). Those who are upstream count who wants to enter the circulatory roadway; those who are downstream count how many vehicles leave the roundabout in that direction. During the survey, the license plates of the vehicles are collected, so as to finally get the route of all registered vehicles. Specifically, the middle 3 numbers for cars are sufficient, for motorcycles last 3 numbers, for bikes is used the abbreviation "BC".
Those upstream must also enter a judgment on average and maximum queue, indicating with an x the position of the registered queue, during a specific survey's stage. To help the operator, checkpoints are listed in the survey sheet. Those downstream of the traffic circle should also enter the number of pedestrians and/or bicycles crossing on their exit at the crosswalks: to report pedestrians it is used an x and for bicycles a B.



Operator name and surname ______________________

Date ________

Time ________

Street ______________

Position code _____

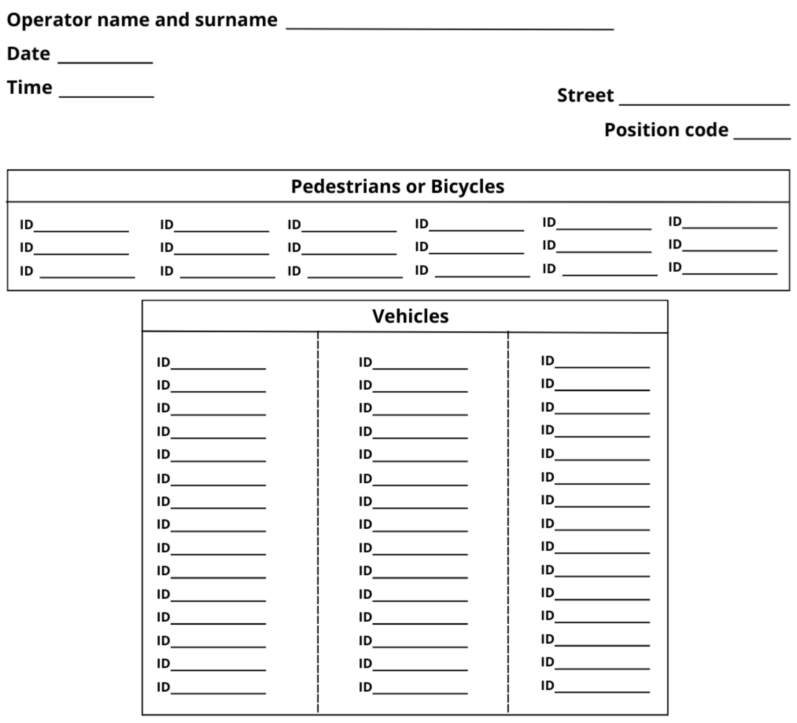
Vehicles		
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________

Queue	
Average	Maximum
CHKP 1	CHKP 1
____	____
____	____
____	____
____	____
CHKP 2	CHKP 2
____	____
____	____
____	____
____	____
CHKP 3	CHKP 3
____	____
____	____
____	____
____	____

Figure 5.3: Template of the sheet used by the upstream operator.

Figure 5.3 shows the template of the sheet compiled by the upstream operator of any leg. The operator uses one or more sheets for every measurement stage. If a new stage begins, a new sheet must be used. The details required are:

- date and time of the measurement stage. The time is defined as the time slot of the measurement taken;
- street and position code. The street may be indicated both with the complete street name or the letter associated. The position code is S2, E2, N2 or W2;
- collected vehicles. Each vehicle must be described by an ID. The ID is the three central numbers of a car plate or the last three numbers of a motorcycle plate. For bicycles, only BC is used. Heavy-duty vehicles must be marked;
- the average and maximum queue. The operator must enter a x in any line between any checkpoints pre-defined. Checkpoints are specifically obtained for each leg and it is known their distance from the entrance inside the circulatory roadway. This distance is the value saved from filling out the format sheet;



Operator name and surname ______________________

Date ________

Time ________

Street ____________

Position code ____

Pedestrians or Bicycles					
ID________	ID________	ID________	ID________	ID________	ID________
ID________	ID________	ID________	ID________	ID________	ID________
ID________	ID________	ID________	ID________	ID________	ID________

	Vehicles	
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________
ID________	ID________	ID________

Figure 5.4: Template of the sheet used by the downstream operator.

As shown in figure 5.4, the downstream operator must report also pedestrians and bicycles, inserting and x or a B, respectively. It is not important their direction, since in the

calibration they will be treated just as crosswalk objects. Also in SUMO, they could be defined as independent flows and, therefore, decoupled from the other ones in the simulation.
The complete calibration of the roundabout has been divided into six intervals, each of ten minutes, for a total of 1 hour of the survey. The intervals are subsequent from one to another. The measurements started at 8:30 A.M. and ended at 9:30 A.M. It has been chosen this time window since the traffic is medium-high in the morning, coinciding with the opening of schools and morning work traffic. Therefore, can be appreciated a general overview of the roundabout traffic scenario.

5.1.2. Calibration results

From the observations carried out the following results have been obtained. These values are crucial to obtain realistic traffic distribution over the four legs of the roundabout. Thanks to this all vehicles inside the simulation can be correctly initialised in SUMO. These values are also used to understand the effect of all the main parameters in SUMO. These parameters can be changed so as to obtain the values to which correspond vehicles that behave in the same way as their twins in reality. Examples of these parameters are the minimum distance between vehicles, the maximum acceleration and deceleration or the impatience with which SUMO-driven vehicles wait before entering the roundabout.

		Queue length							
		E		**N**		**S**		**W**	
Time slot		avg	max	avg	max	avg	max	avg	max
1	8:30 - 8:40	40	89	128	250	17	43	27	91
2	8:40 - 8:50	45	50	50	95	13	26	20	55
3	8:50 - 9:00	25	45	30	60	10	38	7	48
4	9:00 - 9:10	40	45	30	83	13	49	20	63
5	9:10 - 9:20	35	40	30	95	7	38	20	55
6	9:20 - 9:30	50	89	20	60	7	26	20	48

Table 5.1: Queue results from roundabout calibration.

The first values registered are the average and maximum queues for all four legs (i.g. E, S, N, W). They are expressed as the distance between the end of the queue and the point at which the vehicles enter the roundabout. This is obviously an approximate value. To obtain more precise results a GPS system could be used, however, its poor accuracy, especially in such city environments, must be taken into account. In any case, it is not required such high precision for this calibration. From these results, it is possible to obtain the number of vehicles inside the queue by defining the average length of all vehicles' categories.

	Cars	Heavy-duty vehicles
Average length [m]	4,5	12
Minimum gap [m]	1	1,5
Total length [m]	5,5	13,5

Table 5.2: Average length for vehicles' categories.

Heavy-duty vehicles in this case study are represented by public buses and trucks heavier than 3.5 tons. The minimum gap expresses the minimum distance between the vehicle considered and any other subsequent vehicle. Firstly, the percentage of cars with respect to all vehicles is obtained considering:

$$\%_{cars} = n/(n+m) \tag{5.1}$$

5.1: Calculation of the percentage of cars. n stands for the number of cars and m stands for the number of heavy-duty vehicles

Time slot	**E**	**N**	**S**	**W**	**Total**
1	97%	96%	95%	90%	95%
2	94%	94%	89%	89%	92%
3	94%	95%	93%	91%	94%
4	97%	96%	97%	95%	96%
5	94%	94%	94%	90%	94%
6	96%	97%	94%	95%	96%

Table 5.3: Percentages of cars for every leg at every time slot.

From this value, the ratio between cars and heavy-duty vehicles can be obtained.

$$n/m = \%_{cars}/(1 - \%_{cars}) \tag{5.2}$$

5.2: Ratio between the number of cars and heavy-duty vehicles

Time slot	E	N	S	W	Total
1	33,50	21,25	20,50	8,80	18,53
2	15,80	15,80	7,71	8,20	11,50
3	16,75	18,20	13,80	10,25	14,89
4	32,50	24,00	28,50	17,50	25,44
5	15,50	15,60	16,75	9,50	14,41
6	25,33	29,67	16,50	18,00	22,25

Table 5.4: Ratios of cars and heavy-duty vehicles for every leg at every time slot.

It is now possible to obtain the specific number of vehicles inside the queue for every time slot at every leg of the roundabout. To do that the following relation is considered:

$$n_q = \frac{l_q}{(\frac{l_{hv}}{n/m} + l_c)} \tag{5.3}$$

5.3: Calculation of the percentage of cars. n stands for the number of cars and m stands for the number of heavy-duty vehicles. l_q stands for the queue length; l_{hv} stands for the length of heavy-duty vehicles; l_c stands for cars length.

		Number of cars in the queue							
		E		N		S		W	
Time slot		avg	max	avg	max	avg	max	avg	max
1	8:30 - 8:40	6,8	15,1	20,9	40,7	2,8	7,0	3,8	12,9
2	8:40 - 8:50	7,1	7,9	7,9	15,0	1,8	3,6	2,8	7,7
3	8:50 - 9:00	4,0	7,1	4,8	9,6	1,5	5,9	1,0	7,0
4	9:00 - 9:10	6,8	7,6	4,9	13,7	2,2	8,2	3,2	10,0
5	9:10 - 9:20	5,5	6,3	4,7	14,9	1,1	6,0	2,9	7,9
6	9:20 - 9:30	8,3	14,8	3,4	10,1	1,1	4,1	3,2	7,7

Table 5.5: Number of cars inside each queue of the roundabout.

		Number of heavy-duty vehicles in the queue							
		E		**N**		**S**		**W**	
Time slot		avg	max	avg	max	avg	max	avg	max
1	8:30 - 8:40	0,2	0,5	1,0	1,9	0,1	0,3	0,4	1,5
2	8:40 - 8:50	0,4	0,5	0,5	0,9	0,2	0,5	0,3	0,9
3	8:50 - 9:00	0,2	0,4	0,3	0,5	0,1	0,4	0,1	0,7
4	9:00 - 9:10	0,2	0,2	0,2	0,6	0,1	0,3	0,2	0,6
5	9:10 - 9:20	0,4	0,4	0,3	1,0	0,1	0,4	0,3	0,8
6	9:20 - 9:30	0,3	0,6	0,1	0,3	0,1	0,2	0,2	0,4

Table 5.6: Number of heavy-duty vehicles inside each queue of the roundabout.

These are the final values used to calibrate the roundabout. During a generic SUMO simulation, it is possible to retrieve the instantaneous number of vehicles in a queue. SUMO counts all the vehicles with a speed below a threshold. This variable is compared at the end of the simulation with the values obtained by measuring the traffic. Specifically, a few simulations are launched and the average and maximum queue length for any leg is obtained. The SUMO network used in the calibration phase is different with respect to the one used in the driving simulator. This calibration network has also all the pedestrian crosswalks and restricted lanes. It is in fact as close to reality as possible, so that it is comparable with the results obtained by measurements. Below are listed the SUMO parameters considered during calibration:

- jmCrossingGap: minimum distance between the vehicle and the pedestrian that is heading toward the point of conflict of its trajectory with that of the vehicle;
- jmTimegapMinor: minimum time interval for a vehicle to enter an intersection where it does not have the right-of-way, before a vehicle with right-of-way;
- impatience: driver's intent to obstruct a vehicle with the right of way;
- accel: maximum acceleration for the selected vehicle type;
- decel: maximum deceleration for the selected vehicle type;
- tau: minimum time interval between consecutive vehicles;
- actionStepLength: driver reaction time.

Table 5.7 presents the values of the final parameters used in the simulation to train the AV policy and carry out the final tests.

Parameter	Value
jmCrossingGap	1,3545
jmTimegapMinor	1,7792
impatience	0,1182
accel	1,7634
decel	4,2939
tau	1,3472
actionStepLength	0,505

Table 5.7: SUMO calibrated parameters.

5.1.3. SUMO simulation environment

The roundabout's network is built inside SUMO environment using the same tools described in section 2.2. The differences between the real and the virtual scenario are the following:

- no crosswalks are considered, except from a visual point of view;
- the 4 legs of the roundabout have been shortened to a final length of 128 meters;
- the roundabout inner circle has been enlarged to avoid criticalities shown in section 3.4.

With respect to the second difference, it was decided to shorten the legs of the roundabout to lower the maximum number of vehicles simultaneously active in the simulation and respect both the traffic-calibrated flows and the limits on the computational time step.

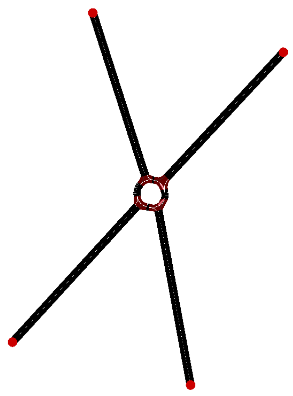


Figure 5.5: Overview of the final network.

The final roundabout is more complex than the first one since more legs are involved and their directions are realistic. Legs are considered completely straight while approaching the roundabout to avoid additional levels of complexity in creating trajectories. The circulatory roadway is narrower.

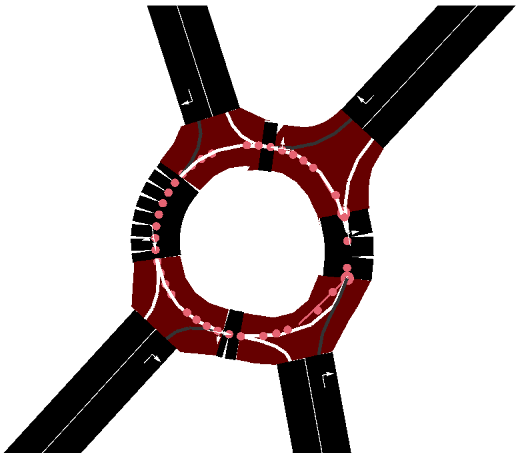


Figure 5.6: Circulatory roadway of the final network.

Figure 5.6 shows the circulatory roadway in detail. All internal junctions, edges and lanes are visible. Specifically, every leg is composed of the following junctions and edges, following the nomenclature given during the traffic calibration and described in section 5.1.1:

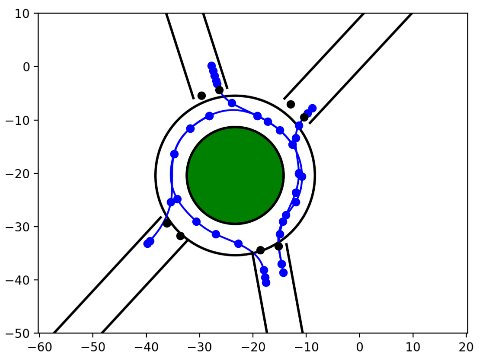
- S_0 and S_R, linked by edges s and -s in the direction of arrival and exit, respectively;
- E_0 and E_R, linked by edges e and -e in the direction of arrival and exit, respectively;
- N_0 and N_R, linked by edges n and -n in the direction of arrival and exit, respectively;
- W_0 and W_R, linked by edges w and -w in the direction of arrival and exit, respectively;

In this final network, all edges have the maximum speed set at 50 km/h since it is the limit imposed by law on city roads in Italy and the new policy must be able to limit the speed if necessary as described in section 4.4.
All trajectories have been interpolated considering the second interpolation algorithm for two reasons:

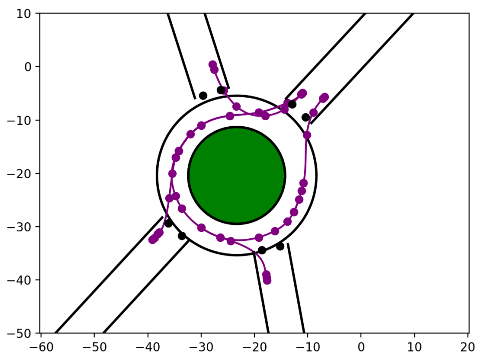
- the number of vehicles in the simulation is much higher than the one for the preliminary tests, resulting in more vehicles to be handled by the algorithm inside the circulatory roadway;
- the quality of the obtained complete trajectories is preferred, having to achieve a more natural motion of the vehicles and thus a better perception of the simulation

environment by the participants, partly sacrificing positional accuracy compared to that of SUMO.

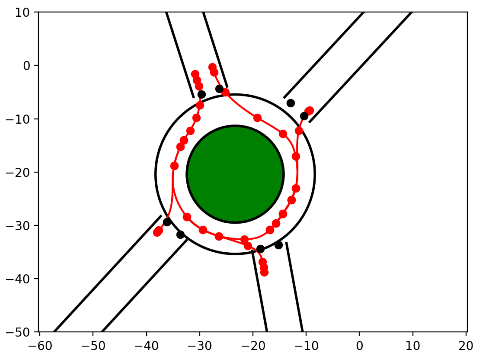
The trajectories obtained are shown below.



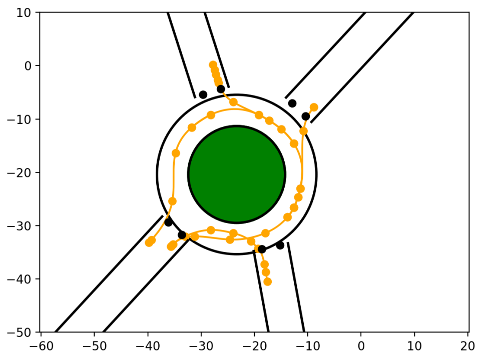
(a) First leg trajectories.



(b) Second leg trajectories.



(c) Third leg trajectories.



(d) Fourth leg trajectories.

Figure 5.7: All final network interpolated trajectories.

Finally, turn signals are implemented throughout the simulation. As described in section 3.4, signal information is retrieved by SUMO and sent to DriveSim to display it, considering only the exiting branches of the circulatory roadway.
The network is, therefore, completely implemented and can be used to carry out the final tests and validate the simulation environment and policy training.

5.2. Final tests

Final tests have a similar structure to the one described for preliminary tests. Participants will run two times from every leg of the roundabout, considering two different traffic distributions in terms of AVs and human-driven vehicles; specifically, 20% AVs or 80% AVs are taken into account. The order in which the two scenarios are presented to users is kept equal for all the roundabout's legs, but it is unknown. As there are more legs and greater traffic, participants will remain in the simulation environment for a longer time than in the preliminary network, allowing users to be able to fully evaluate the performance of the simulation environment.

Test	Scenario	Starting point
1	20% of human-driven vehicles and 80% of AVs	Leg 1
2	80% of human-driven vehicles and 20% of AVs	Leg 1
3	20% of human-driven vehicles and 80% of AVs	Leg 2
4	80% of human-driven vehicles and 20% of AVs	Leg 2
5	20% of human-driven vehicles and 80% of AVs	Leg 3
6	80% of human-driven vehicles and 20% of AVs	Leg 3
7	20% of human-driven vehicles and 80% of AVs	Leg 4
8	80% of human-driven vehicles and 20% of AVs	Leg 4

Table 5.8: Final tests simulations order.

Participants will perform all simulations at DriSMi, exactly the same as for the preliminary tests, with the only difference being the exit from the circulatory roadway. They will start from fixed points equal for all of them from every leg, going down the straight, entering the roundabout and exiting at the third available exit. Once they are on the exit straight, they can stop the simulation whenever they prefer. The full test can be completed in a total of about 30 minutes, including all the downtime between changing the various scenarios and starting legs. All participants will arrive 15 minutes in advance with respect to the start of the simulations. They will receive all the necessary information to carry out the experiment, listed below:

- the configuration of the entire test will be communicated as shown in Table 5.8, omitting the specific difference between the two scenarios;
- they will be asked to focus on the differences in safety, crossing time, and realism of the simulation;

- they will be instructed with respect to how to complete the questionnaire.

5.2.1. Final questionnaire

It was chosen to present the questionnaire in the same manner and with the same questions as described in section 3.2 for the preliminary tests. In this way, it is possible to ensure the comparison between the results of the two tests carried out. This is also done since the policy used, at least the one of the first final tests, is the same as the preliminary one. The primary objective of the RL policy controlling the AVs remains the minimisation of crossing time, however, being able to work within a simulation environment optimized by the results obtained from the initial trials allows to add realism to the simulation and increases the reliability of the collected results. The only difference concerning the preliminary tests is the number of starting points the participants will begin the simulations from. This is beneficial to have a more extensive understanding of the simulation environment but, on the other hand, it could introduce a greater difficulty in understanding the questions and thus errors in the answers. The questionnaire will, therefore, validate all modifications applied to the simulation to make it as much similar to reality as possible. All suggested modifications in section 3.4 may be applied for further tests on new policies and simulation environments. Finally, to prove in a more detailed and effective way the reduction of crossing time of function of autonomous vehicle penetration, the same equation presented in section 3.2 is used to obtain the average crossing time. In this case, the starting time is set when the vehicle enters the simulation and the final time is obtained when the vehicle leaves the network. Accordingly, it is presented a more comprehensive description of the crossing time which considers also the waiting time during the simulation. Table 5.9 presents the results achieved.

	0% of Avs	**20% of Avs**	**80% of Avs**
Average crossing time [s]	56.26	54,49	49,01
Maximum crossing time [s]	87,53	83,32	79,66
N. vehs completed the simulation [-]	35	39	41
Reduction of crossing time	ref.	3,15%	12,88%

Table 5.9: Average crossing time as a function of AVs penetration.

The AVs are able to reduce the crossing time of the roundabout even considering the whole network from the beginning of the starting edge to the end of the final edge. The reduction is obviously lower than the one obtained for the circulatory roadway only.

5.2.2. Final tests results

The results of the final tests are presented in the same way as the preliminary ones. This, again, ensures the comparability of the two simulation environments and the validation of the modifications implemented. Moreover, participants were chosen from people outside PoliMi who were not informed about the driving simulator or its operation. An attempt is made to obtain results that are as general as possible; on the other hand, the general inexperience of the users must be considered and, in some cases, can compromise the results. In this way, also the simulator's ability to be an effective tool for collecting data, such as those needed for this research, is tested. In total, 14 participants attended the tests but only 10 completed them. The others felt sick during the simulations. This is due both to the users' inexperience at DriSMi and to the difficulty of the roundabout, within which there are narrow radius curves. Finally, the roundabout obtained for this final simulation environment turns out to be perceived as largely more realistic and, for this reason, the results obtained are certainly of greater value in this field of research.
The first section is the one with reference to the judgment of the individual scenarios.

First Question	Quantity
Completely agree	0
Agree	6
Neither agree nor disagree	0
Disagree	3
Completely disagree	1

(a) Results of the first question of the second section of the final tests questionnaire.

Second Question	Quantity
One vehicle	0
More than one vehicle	4

(b) Results of the second question of the second section of the final tests questionnaire.

Table 5.10: Perception of compliance with the rules of right of way in the roundabout, considering the scenario with 80% of AVs for the final tests.

First Question	Quantity
Completely agree	0
Agree	2
Neither agree nor disagree	0
Disagree	6
Completely disagree	2

(a) Results of the first question of the second section of the final tests questionnaire.

Second Question	Quantity
One vehicle	3
More than one vehicle	5

(b) Results of the second question of the second section of the final tests questionnaire.

Table 5.11: Perception of compliance with the rules of right of way in the roundabout, considering the scenario with 20% of AVs for the final tests.

Tables 5.10 5.11 shows better behaviour of vehicles in the 80% AVs scenario in respecting the rules of right of way. Still, in some simulations, more than one vehicle was not compliant with the precedence logics. Overall, the new simulation environment proves to be better than the preliminary one. Modifications applied have been effective in limiting the consequences of a bad environment design. The perfect environment cannot be achieved and would be far from reality. Furthermore, it is necessary to emphasize the participants' inexperience, which in some cases led to incorrect behaviour, such as very low travel speeds due to the novelty of perceived sensations of acceleration and braking or a very large distance from the next vehicles.

Table 5.12 presents the results of the last question of the second section referring to the perception of traffic behaviour entering the circulatory roadway.

Third Question	Quantity
Completely agree	0
Agree	1
Neither agree nor disagree	2
Disagree	5
Completely disagree	2

(a) Results of the third question of the second section of the final tests questionnaire, referred to 80% AV scenario.

Third Question	Quantity
Completely agree	0
Agree	2
Neither agree nor disagree	0
Disagree	6
Completely disagree	2

(b) Results of the third question of the second section of the final tests questionnaire, referred to 20% AV scenario.

Table 5.12: Perception of traffic in the final roundabout scenario.

Traffic inputting the roundabout has been perceived in the same way for the two simulation scenarios, without any visible difference. Considering a future version of the questionnaire, this question can be extended to show also minor differences in behaviour.

The last section of the questionnaire refers to the direct comparison of the two scenarios. Also in this case, results have been sorted considering scenario 1 the simulation with 20% of AVs and scenario 2 the simulation with 80% of AVs. Looking at table 5.13 it is possible to highlight:

- the traffic smoothness has been overall perceived equally between the two scenarios. Furthermore, it is possible to underline a slight perception in favour of the second scenario, in accordance with the policy used and its purpose. With respect to the result obtained in section 5.2, the reduction is difficult to be perceived by participants who used DriSMi for the first time. In fact, they focus on other aspects, such as their safety or in general the feelings they experience during the simulation. In order to have a complete demonstration of this aspect, it would be needed to consider experienced drivers and longer simulation with denser queues;
- participants felt generally safer in the second scenario where vehicles respected the most the rules of right of way. AVs have been able to orchestrate the traffic so as to obtain a safer environment, acting as a whole and not just like single entities in the network;
- the second scenario has been significantly preferred. This is one of the most important results of this questionnaire and final tests. Users felt better with more AVs in the network, proving the validity of the study and, more in general, of this future trend. Having vehicles communicating with each other and affecting the whole system becomes an advantage also for unconnected vehicles which commit fewer accidents and behave accordingly and thus also in contact with other vehicles in the simulation environment.

TRAFFIC SMOOTHNESS	
First question	**Quantity**
Traffic in scenario 1 was significantly smoother than in scenario 2	1
Traffic in scenario 1 was partially smoother than in scenario 2	3
Traffic in scenario 2 was partially smoother than in scenario 1	4
Traffic in scenario 2 was significantly smoother than in scenario 1	1
I perceived no difference in the smoothness of traffic in the two scenarios	1

(a) Comparison of perception of traffic's smoothness in the roundabout.

TRAFFIC SAFETY	
Second question	**Quantity**
In scenario 1 I felt significantly safer than in scenario 2	0
In scenario 1 I felt partially safer than in scenario 2	2
In scenario 2 I felt partially safer	2
In scenario 2 I felt significantly safer than in scenario 1	6
I perceived no difference between the 2 scenarios in the feeling of safety	0

(b) Comparison of perception of safety in the roundabout.

GLOBAL PREFERENCE	
Third question	**Quantity**
I significantly preferred scenario 1 to scenario 2	0
I partially preferred scenario 1 to scenario 2	3
I partially preferred scenario 2 to scenario 1	3
I significantly preferred scenario 2 to scenario 1	4
I can't say which of the two scenarios I preferred	0

(c) Global preference between the two scenarios.

Table 5.13: Anwers of the third questionnaire's section, referring to the scenarios comparison for the final tests.

With respect to the last question, in preparation for more extensive testing, it will be necessary to further investigate the reason for the preference, being in some cases unrelated to both safety and speed of travel.

Finally, table 5.14 presents the coherence analysis of the most interesting participants' answers. This analysis is also relevant to both validate the observations made in this section and propose further modifications to the questionnaire.

Participant ID	Comments on their answers
1	In general they show consistency among responses. They feel safer in the scenario where vehicles respected the right of way and rate the one with more aggressive vehicles as smoother. The only response unrelated to the others is the last one, which denotes the need to specify the type of preference.
2	The direct evaluation of the two scenarios turns out to be the same. The differences, in this case, are minimal, but a well-defined perception of safety is denoted. This demonstrates the need to investigate more what happens during the judging of the individual scenario.
3	Again, the judgment of the individual scenariosis identical. A general consistency is denoted by considering the direct comparison of the scenarios. The participant prefers the smoother and safer scenario.
4	The participant demonstrates consistency in responses, both in direct comparison and in judging individual scenarios.

Table 5.14: Coeherence analysis on participants' answers for the final tests. Green highlights coherence, yellow not complete coherence, and red incoherence.

The analysis highlights a complete coherence of participants, higher than the one registered for preliminary tests. This may be due to the improved simulation environment which allowed them to have a more complete perception of the two scenarios. Users spent more time in the simulation with realistic traffic distributed over the four legs of the roundabout. Also, the coherence for all the other users has been verified.

6 | Conclusions and future developments

This thesis work aimed to investigate the creation of a Digital Twin useful for assessing the presence of Autonomous Vehicles within a city roundabout. AVs were considered both at the macroscopic level, as part of a larger system and in communication with each other and the infrastructure, and at the microscopic level, taking into consideration the effect on human passengers. The first important result has been the creation of a stable and functioning communication system between Flow, SUMO and the driving simulator. This enabled the development of a preliminary simulation environment in which it has been tested the system performance and reliability. This configuration has been shown to be capable of handling such a complex simulation scenario within which are present AVs, vehicles driven by driver models and, more importantly, a human user which introduces variability and unpredictability, features necessary to make the Digital Twin resemble the real condition. Through preliminary testing, it was then possible to create the final simulation environment and obtain the following crucial results:

1. the policy has proven to be able to handle different environment geometries. This is an important result since in a real scenario it should be able to travel in different cities and states. Thanks to edge computing it will also be able to learn different logics and rules using the data made available by other vehicles in the network;

2. the AVs were able to handle a variety of driver behaviours, being tested with various and different participants, both by simulator experience and more general driving experience. It denotes, in some cases, the difficulty of the policy to understand the human driver's intentions, e.g. if the driver has a too-low travel speed or they keep a too-high distance from following vehicles. In these examples, the policy took decisions which may be considered dangerous by human users. Further versions of the DRL algorithm must be trained with more participants so as to acquire a better knowledge of these scenarios;

3. for the most part, participants felt safer and preferred the scenario with a higher

percentage of Autonomous Vehicles. This turns out to be one of the most important findings of this research, underscoring the validity of this solution for the future of mobility. On the other hand, a pronounced perception of a shorter travel time was not demonstrated. This can be attributed to the traffic, which unless it is very high does not allow a high perception of this variable, and to the driving style and experience of human drivers, who, by maintaining high safety distances from the following vehicles, could not appreciate a faster entry of Autonomous Vehicles inside the circulatory roadway. Regarding the latter consideration, for future tests, it will be necessary to give participants more time to understand the operation of the simulator and thus feel safe during the simulations, eventually being able to appreciate the reduction obtained;

4. the *Replay* simulation has been crucial to completely understand the policy performances, also considering safety and comfort. It will be an essential tool in order to complete comprehensive testing for the AI@EDGE project. It can also be used to improve the trajectories algorithms implementation, adding a much more precise calculation of yaw angle;

5. the traffic calibration led to a better performing scenario which participants felt was real and contributed to more reliable and valuable results.

Future developments may consider new DRL algorithms to account for more relevant KPIs, such as the reduction of fuel consumption and pollutants produced by vehicles and higher safety and comfort for the human driver, considering limits on lateral and longitudinal acceleration. It is also crucial to understand the effect of cooperative AVs in the network. Specifically, the roundabout had an uneven distribution of traffic on the four legs, leading to widely varying queues. This can be reduced, for example, by training the policy and having Autonomous Vehicles interact more as a group than as individuals. Moreover, the *Replay* scenario may be used more and more in collaboration with policy training in order to optimize the behaviour of Autonomous Vehicles and their interaction with humans and thus becoming a real solution that can be used in real scenarios.

Finally, tests on a much larger population of participants will have to be completed in order to validate the results obtained in these initial tests and also propose a more accurate analysis of the responses.

Bibliography

[1] E. G. M. 017. Mobile edge computing (mec); deployment of mobile edge computing in an nfv environment. February 2018.

[2] 5GAA. An assessment of lte-v2x (pc5) and 802.11p direct communications technologies for improved road safety in the eu. 2017.

[3] F. Al-Dhief, N. Sabri, N. M. Abdul Latiff, N. N. Nik Abd Malik, M. K. Abd Ghani, M. Mohammed, R. Al-Haddad, Y. Dawood, M. Ghani, and O. Ibrahim Obaid. Performance comparison between tcp and udp protocols in different simulation scenarios. *International Journal of Engineering Technology*, 7:172–176, 01 2018.

[4] M. Ali, M. Mailah, Tang, , and H. Hing. Path navigation for mobile robot in a road roundabout setting. 12 2011.

[5] M. Ali, M. Mailah, and T. Hing. Path planning of mobile robot for autonomous navigation of road roundabout intersection. 6:203–211, 01 2012.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 2017.

[7] S. R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige. Vehicular networks for collision avoidance at intersections. *Society for Automotive Engineers (SAE) World Congress*, 2011.

[8] S. R. Azimi, G. Bhatia, R. R. Rajkumar, and P. Mudalige. Intersection management using vehicular networks. *Society for Automotive Engineers (SAE) World Congress*, 2012.

[9] A. Bakibillah, M. Kamal, Susilawati, and C. Tan. The optimal coordination of connected and automated vehicles at roundabouts. *Proceedings of the SICE Annual Conference,*, page 1392–1397, 2019.

[10] L. C. Bento, R. Parafita, and U. Nunes. Intelligent traffic management at intersections supported by v2v and v2i communications. In *2012 15th International*

IEEE Conference on Intelligent Transportation Systems, pages 1495–1502, 2012. doi: 10.1109/ITSC.2012.6338766.

[11] Y. Bichiou and H. A. Rakha. Developing an optimal intersection control system for automated connected vehicles. *IEEE*, 2018.

[12] S. Biswas, R. Tatchikou, and F. Dion. Vehicle-to-vehicle wireless communication protocols for enhancing highway traffic safety. *IEEE Communications Magazine*, 44 (1):74–82, 2006. doi: 10.1109/MCOM.2006.1580935.

[13] K. Bucsuházy, E. Matuchová, R. Zůvala, P. Moravcová, M. Kostíková, and R. Mikulec. Human factors contributing to the road traffic accident occurrence. *Transportation Research Procedia*, 45:555–561, 2020. ISSN 2352-1465. doi: https://doi.org/10.1016/j.trpro.2020.03.057. URL `https://www.sciencedirect.com/science/article/pii/S2352146520302192`. Transport Infrastructure and systems in a changing world. Towards a more sustainable, reliable and smarter mobility.TIS Roma 2019 Conference Proceedings.

[14] "CCAM". Ccam - european partnership, 2022. URL `https://www.ccam.eu/`.

[15] T. Chen and N. Nikaein. Towards software defined 5g radio access networks, 2016.

[16] L. G. Cuenca, J. E. Puertas, F. Andrés, and N. Aliane. Autonomous driving in roundabout maneuvers using reinforcement learning with q-learning. *Electronics*, 2019.

[17] S. Davidović, V. Bogdanović, N. Garunović, Z. Papic, and D. Pamucar. Research on speeds at roundabouts for the needs of sustainable traffic management. *Sustainability*, 13:399, 01 2021. doi: 10.3390/su13010399.

[18] A. Demba and D. P. F. Möller. Vehicle-to-vehicle communication technology. *IEEE*, 2018.

[19] N. Distefano and S. Leonardi. *Rotatorie Stradali:manuale di pianificazione, progettazione e gestione*. EPC editore, 2021.

[20] P. E and M. M. Human factors in the causation of road traffic crashes. *Eur J Epidemiol*, 2000. doi: 10.1023/a:1007649804201.

[21] P. Frenger and R. Tano. More capacity and less power: How 5g nr can reduce network energy consumption. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5, 2019. doi: 10.1109/VTCSpring.2019.8746600.

[22] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin,

and A. Kousaridas. A tutorial on 5g nr v2x communications. *IEEE Communications Surveys Tutorials*, 23(3):1972–2026, 2021. doi: 10.1109/COMST.2021.3057017.

[23] L. García Cuenca, J. Sanchez-Soriano, E. Puertas, J. Fernandez Andrés, and N. Aliane. Machine learning techniques for undertaking roundabouts in autonomous driving. *Sensors*, 19(10), 2019. ISSN 1424-8220. doi: 10.3390/s19102386. URL `https://www.mdpi.com/1424-8220/19/10/2386`.

[24] S. Gyawali, S. Xu, Y. Qian, and R. Q. Hu. Challenges and solutions for cellular based v2x communications. *IEEE*, 2020.

[25] S. Hakak, T. R. Gadekallu, P. K. R. Maddikunta, S. P. Ramu, Parimala, and M. L. Chamitha De Alwis. Autonomous vehicles in 5g and beyond: A survey. *Vehicular Communications*, 2022.

[26] P. Hang, C. Huang, Z. Hu, Y. Xing, and C. Lv. Decision making of connected automated vehicles at an unsignalized roundabout considering personalized driving behaviours. *arXiv*, 2021.

[27] N. Hassan, K.-L. A. Yau, and C. Wu. Edge computing in 5g: A review. *IEEE Access*, 7:127276–127289, 2019. doi: 10.1109/ACCESS.2019.2938534.

[28] H. Hong, Y. Y. Kim, R. Y. Kim, and W. Ahn. An effective wide-bandwidth channel access in next-generation wlan-based v2x communications. *Applied Sciences*, 10(1), 2020. ISSN 2076-3417. doi: 10.3390/app10010222. URL `https://www.mdpi.com/2076-3417/10/1/222`.

[29] S. P. Hoogendoorn and R. Hoogendoorn. Generic calibration framework for joint estimation of car-following models by using microscopic data. *Transportation Research Record*, 2010.

[30] Y. Hou, Y. Zhao, K. F. Hulme, Y. Y. Shan Huang, A. W. Sadek, and C. Qiao. An integrated traffic-driving simulation framework: Design, implementation, and validation. *Transportation Research Part C Emerging Technologies*, 2014.

[31] F.-H. Hsu, Y.-L. Hwang, C.-Y. Tsai, W.-T. Cai, C.-H. Lee, and K. Chang. Trap: A three-way handshake server for tcp connection establishment. *Applied Sciences*, 6 (11), 2016. ISSN 2076-3417. doi: 10.3390/app6110358. URL `https://www.mdpi.com/2076-3417/6/11/358`.

[32] J. Hummer, J. Milazzo, B. Schroeder, and K. Salamati. Potential for metering to help roundabouts manage peak period demands in the united states. *Transportation Research Record: Journal of the Transportation Research Board*, page 56– 66, 2014.

[33] M. C.-C. (i2CAT), E. C. C. (i2CAT), N. di Pietro (ATH), A. I. (ATH), M. C. (ATH), O. A. (INRIA), J. P. (INRIA), S. S. (CNAM), F. B. (EAB), N. L. (EAB), P. Ödling (ULUND), C. C. (FBK), A. W. (FBK), L. G. (ATOS), G. A. (8Bells), S. K. (8Bells), A. A. (ITL), B. M. (SRS), B. M. (SPI), M. M. (CRF), M. R. (AERO), N. S. (DFKI), G. L. (ICCS), B. A. (RISE), G. M. (POLIMI), and N. P. M. (TIM). D2.2 preliminary assessment of system architecture, interfaces specifications, and techno-economic analysis. *CoRR, vol. abs/1710.05465*, 2017.

[34] S. International". Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, 2021. URL `https://www.sae.org/standards/content/j3016_202104/`.

[35] G. Jacquemart. Modern roundabout practice in the united states. *Number Project 20-5 FY 1996*, February 1998.

[36] K. Jang, E. Vinitsky, B. Chalaki, L. B. Ben Remer, A. Malikopoulos, and A. Bayen. Simulation to scaled city: Zero-shot policy transfer for traffic control via autonomous vehicles. *ArXiv*, 2018.

[37] Y. M. Jang, J.-C. Cano, K. Yang, and Y.-J. Choi. Enabling technologies towards next generation mobile systems and networks. *Mobile Information Systems*, 2016: 1–2, 01 2016. doi: 10.1155/2016/9805636.

[38] N. A. Johansson, Y.-P. E. Wang, E. Eriksson, and M. Hessler. Radio access for ultra-reliable and low-latency 5g communications. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 1184–1189, 2015. doi: 10.1109/ICCW.2015.7247338.

[39] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper_files/paper/2013/file/ac1dd209cbcc5e5d1c6e28598e8cbbe8-Paper.pdf`.

[40] N. Ketkar. *Stochastic Gradient Descent*, pages 113–132. Apress, Berkeley, CA, 2017. ISBN 978-1-4842-2766-4. doi: 10.1007/978-1-4842-2766-4_8. URL `https://doi.org/10.1007/978-1-4842-2766-4_8`.

[41] O. Kuba1 and J. Jagelcak1. Lateral acceleration of passenger vehicle in roundabouts in term of cargo securing. *IOP Conf. Ser.: Mater. Sci. Eng.*, 2022. doi: 10.1088/1757-899X/1247/1/012037.

[42] X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 983–992. PMLR, 16–18 Apr 2019. URL `https://proceedings.mlr.press/v89/li19c.html`.

[43] Y. Li. Deep reinforcement learning: An overview. *arXiv*, 2018.

[44] Y. Liu, J. Peng, J. Kang, A. M. Iliyasu, D. Niyato, and A. A. A. El-Latif. A secure federated learning framework for 5g networks. *IEEE Wireless Communications*, 27 (4):24–31, 2020. doi: 10.1109/MWC.01.1900525.

[45] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, and Y.-P. Fl. Microscopic traffic simulation using sumo. 2018.

[46] P. Maheshwary, K. Bhattacharyya, B. Maitra, and M. Boltze. A methodology for calibration of traffic micro-simulator for urban heterogeneous traffic operations. *Journal of Traffic and Transportation Engineering*, pages 507–512, 2020.

[47] D. W. Matolak. V2v communication channels: State of knowledge, new results, and what's next. In M. Berbineau, M. Jonsson, J.-M. Bonnin, S. Cherkaoui, M. Aguado, C. Rico-Garcia, H. Ghannoum, R. Mehmood, and A. Vinel, editors, *Communication Technologies for Vehicles*, pages 1–21, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37974-1.

[48] V. Milanés, J. Villagrá, J. Godoy, J. Simó, J. Pérez, and E. Onieva. An intelligent v2i-based traffic management system. *IEEE*, 2012.

[49] A. Montella, S. Turner, S. Chiaradonna, and D. Aldridge. International overview of roundabout design practices and insights for improvement of the italian standard. *Canadian Journal of Civil Engineering*, pages 1215–1226, 2013.

[50] A. Pakgohar, R. S. Tabrizi, M. Khalili, and A. Esmaeili. The role of human factor in incidence and severity of road crashes based on the cart and lr regression: a data mining approach. *Procedia Computer Science*, 3:764–769, 2011. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2010.12.126. URL `https://www.sciencedirect.com/science/article/pii/S1877050910005016`. World Conference on Information Technology.

[51] L. Qian, Z. Luo, Y. Du, and L. Guo. Cloud computing: An overview. In M. G. Jaatun, G. Zhao, and C. Rong, editors, *Cloud Computing*, pages 626–631, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-10665-1.

[52] F. F. Saccomanno, F. Cunto, G. Guido, and A. Vitale. Comparing safety at signalized intersections and roundabouts using simulated rear-end conflicts. *Transportation Research Record*, 2078(1):90–95, 2008. doi: 10.3141/2078-12. URL `https://doi.org/10.3141/2078-12`.

[53] M. Sewak. *Policy-Based Reinforcement Learning Approaches: Stochastic Policy Gradient and the REINFORCE Algorithm*, pages 127–140. 06 2019. ISBN 978-981-13-8284-0. doi: 10.1007/978-981-13-8285-7_10.

[54] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* 2018.

[55] D. Tollner, H. Cao, and M. Zöldy. Artificial intelligence based decision making of autonomous vehicles before entering roundabout. *2019 IEEE 19th International Symposium on Computational Intelligence and Informatics*, 2019.

[56] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *CoRR, vol. abs/1710.05465*, 2017.

[57] X. Yang, X. Li, , and K. Xue. A new traffic-signal control for modern roundabouts: method and application. *IEEE Transactions on Intelligent Transportation Systems*, pages 282–287, 2004.

[58] H. Zhang, Yu, and T. *Taxonomy of Reinforcement Learning Algorithms.* Springer, Singapore, 2020.

[59] S. Zhang. An overview of network slicing for 5g. *IEEE Wireless Communications*, 26(3):111–117, 2019. doi: 10.1109/MWC.2019.1800234.

[60] M. Zhu, X. Wang, A. Tarko, and S. Fang. Modeling car-following behavior on urban expressways in shanghai: A naturalistic driving study. *Transportation Research Part C: Emerging Technologies*, 93:425–445, 2018. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2018.06.009. URL `https://www.sciencedirect.com/science/article/pii/S0968090X18308635`.

[61] S. Šurdonja, D. Aleksandra, and S. Babić. Optimization of roundabout design elements. *Tehnicki Vjesnik*, 20:533–539, 05 2013.

A | Appendix A

Extended code for Terminal 1.

```
env_name = TestEnv

# ADAPT THESE PATHS TO THE SPECIFIC COMPUTER USED
# This fisrt path referes to the checkpoint of the policy used
PATH_TO_SAVED_MODEL = "/Users/lore/Desktop/POLICY/POLICY_FILES/checkpoint_1"
# This second path refers to the network chosen for the simulation
LuST_dir = "/Users/lore/Desktop/POLICY/POLICY_FILES/RB_shorten"
USE_INFLOWS = True
inflow = InFlows()

# probability of a vehicle appearing every second of the simulation
# use this only for probabilities - We do not use it since we want a
    deterministic scenario
#INFLOW_PARAMS = {
#    'human': 0.2,
#    'ai': 0.1
#}

# create some default parameters parameters
env_params = EnvParams()
initial_config = InitialConfig()
vehicles = VehicleParams()
#lane_change_model = SUMOLaneChangeParams(model= )

### COLD START VEHICLES ###
# the following two lines determine how many vehicles will already be on the
    network when the simulation starts
vehicles.add("f_0.0", acceleration_controller=(IDMController, {}),
    num_vehicles=2)
vehicles.add('ai', acceleration_controller=(RLController, {}), num_vehicles=3)
```

```


class TemplateNetwork(Network):

    def specify_routes(self, net_params):
        return {"f_0.0_0": ["s", "se", "-e"],
                "f_4.0_0": ["e", "en", "-n"],
                "f_8.0_0": ["n", "nw", "-w"],
                "f_12.0_0": ["w", "ws", "-s"],
                "ai_0": ["w", "ws", "se", "en", "nw", "-w"],
    }
    pass

if USE_INFLOWS:

    # EDGE S ----------------------------------------------------------
    # ----------------------------------------------------------------
    inflow.add(
        veh_type="ai",
        edge='s',
        #probability=INFLOW_PARAMS['ai']
        vehs_per_hour = 27,
        route = "routef_0.0_0_0",
    )


ADDITIONAL_NET_PARAMS = {
    # length of the ring road
    "max_num_vehicles": 36,
}

sim_params = SUMOParams(render=True, sim_step=0.005, num_clients=2)

if USE_INFLOWS:
    net_params = NetParams(template={
            "net": os.path.join(LuST_dir, "RB_padova_WS_short.net.xml")
        }, inflows=inflow)
else:
    net_params = NetParams(template={
            "net": os.path.join(LuST_dir, "RB_padova_WS_short.net.xml"),
        })

initial_config = InitialConfig(edges_distribution=["e", "s", "n", "w"],
    spacing = "uniform")

flow_params = dict(
    exp_tag='marl-centralized-ai-edge-polimi-paper-draft-2',
    env_name=TestEnv,
    network=TemplateNetwork,
    simulator='traci',
    sim=sim_params,
    env=env_params,
    net=net_params,
    veh=vehicles,
    initial=initial_config,
    #lane_change = lane_change_model
)
```

B | Appendix B

Extended code for Terminal 2.

```
impostazioni_SUMO = {'salva_risultati': 1, # salva i dati su un file di testo: 1 si, 0 no
                   'informazioni_a_video': 2} # cosa si visualizza a video: 1 tutte le informazioni, 2 solo cili e tempo di calcolo, 0 nulla

# Impostazioni scenario e veicoli
scenario = {'id_ego': 'f_0.0_0', # id della ego car in SUMO
           'id_vei_worldsim': {'f_0.0_1',"f_1.0_0", 'f_1.0_1', "f_2.0_0", "f_3.0_0", "f_4.0_0", "f_5.0_0", "f_6.0_0", "f_7.0_0", "f_8.0_0",
                            },  # elenco degli id delle macchine in SUMO da inviare a worldsim.
           'yaw_offset': 90, # offset rotazione di imbardata tra SUMO e worldsim
           'n_step_iniziali': 4000, # numero di passi da simulare all'inizio della simulazione per creare il traffico
           'ego_init': (251.939, 277.538, 180), # posizione iniziale della ego (solo per test senza trasmissione dati)
           'x_offset': 0,# offset tra l'origine dello scenario SUMO e quella dello scenario Worldsim in direzione x
           'y_offset': 0, # offset tra l'origine dello scenario SUMO e quella dello scenario Worldsim in direzione x
           'veicoli_attivi': [], # veicoli da inviare a worldsim effettivamente presenti in SUMO al passo attuale di integrazione
           'veicoli_non_attivi': []} # veicoli da inviare a a worldsim non presenti in SUMO al passo attuale di integrazione
# dati veicoli (usare come etichetta l'id veicolo): distanza tra la sala anteriore e il paraurti e angolo di imbardata iniziale (convenzione SUMO, gradi)
```

```
scenario['veicoli']={}

# Needed for every vehicle in the scenario ----------------
scenario['veicoli']['f_0.0_0']= {'L_bumper': 0.849, 'angle_iniziale': 349.571,

# Inserisco i valori necessari per il calcolo delle traiettorie di ogni veicolo
for i in scenario['veicoli'].keys():
    scenario['veicoli'][i]['delta_prec'] = 0
    scenario['veicoli'][i]['steer_prec'] = 0
    scenario['veicoli'][i]['control'] = 0
    scenario['veicoli'][i]["points_couples"] = [0,0]

# Denominazione veicoli.
# Gli id dei veicoli devono essere uguali tra questo software e SUMO. In
    worldsim c' una definizione degli id diversa. Per far corrispondere i
    veicoli, ogni veicolo in questo software
# collegato a una porta del client UDP che mantiene per tutta la simulazione.
    Sul cuncurrent, ogni porta del client UDP  mappata univocamente in un
    veicolo di worldsim

# impostazioni server e client UDP
impostazioni_UDC = {'UDC_IP_SELF': '192.168.100.77', #indirizzo ip di questa
    macchina sulla rete del simulatore
                   'UDC_IP_CUNCURRENT': '192.168.100.22', # UDC_IP_CUNCURRENT:
                       indirizzo ip del cuncurrent
                   'UDC_PORT_SELF': 30001, # deve essere uguale alla porta da
                       cui riceve il cuncurrent nella configurazione della IO
                   'time_out': 10, # tempo di attesa prima di chiudere la
                       connessione se non ricevo dati
                   'n_veicoli_worldsim': len(scenario['id_vei_worldsim']),
                   'n_variabili_veicolo_worldsim': 5, # numero di parametri da
                       inviare a worldsim per ogni veicolo
                   'n_variabili_ego': 3} # numero di parametri ricevuti da
                       worldsim per la ego car
impostazioni_UDC['buffer_size'] = 8 * (1 +
    impostazioni_UDC['n_variabili_ego']) # dimensione del buffer di ricezione
    della ego (il primo byte  il flag per sapere lo stato della simulazione)
impostazioni_UDC['codifica_ricezione'] = '<' + str(1 +
    impostazioni_UDC['n_variabili_ego']) + 'd'
```

```
impostazioni_UDC['codifica_invio'] = '<' +
   str(impostazioni_UDC['n_variabili_veicolo_worldsim']) + 'd'
# elenco porte di ricezione sul cuncurrent nella configurazione della IO (ogni
   id veicolo  linkato univocamente a una porta)
impostazioni_UDC['lista_porte_CUNCURRENT']={}
# Needed for every vehicle displayed during the simulation
impostazioni_UDC['lista_porte_CUNCURRENT']['f_0.0_0']=30001

veh_id = list(scenario['id_vei_worldsim'])
veh_id.append("f_0.0_0")
speed = ["Speed"]
acc = ["acc"]
data_vehicles = dict.fromkeys(veh_id)

for i in data_vehicles:
    data_vehicles[i] = dict.fromkeys(speed, [])
    data_vehicles[i]["acc"] = []
    data_vehicles[i]["cicli"] = []
    data_vehicles[i]["lateral_speed"] = []
    data_vehicles[i]["distance_travelled"] = [0]
    data_vehicles[i]["previous_value"] = 0
    data_vehicles[i]["angle_difference"] = [0]
    data_vehicles[i]["previous_value_angle"] = 0
    data_vehicles[i]["acc_lat"] = [0]


#%%


# dati della rete stradale di SUMO riparametrizzati in modo da avere
   traiettorie fluide (da fare per ogni scenario)
# Il file matlab leggi_rete_SUMO.m trasforma la rete SUMO un questo formato
   (attualmente implementati solo rettilinei e tratti di circonferenza)
# tipi di tratto implementati:
# rettilineo: angolo sterzo, imbardata
# curva (arco di cerchio): coordinate centro, raggio, angolo sterzo, imbardata
   di riferimento
interpola_corsie={} # numero di punti per la media pesata sull'angolo di sterzo

```

```
interpola_corsie["buffer"] = {'buffer_imbardata': 60, # numero di punti per la
   media pesata sull'imbardata
                'buffer_sterzo': 50}
# CORSIE - Needed for all trajectories
interpola_corsie[":1_1bis"] = {"traiettoria":
   [-14.3433,-38.6232,-14.62967,-36.97232,-15.0,-31.4,-13.8,-27.8,-12.0,-23.6,
-11.4,-20.0732,-11.4,-20.0]}

# Needed for all trajectories
x_1_1 = np.concatenate((interpola_corsie[":1_1_straight"]
["xnew"],interpola_corsie[":1_1bis"]["interpolata"](interpola_corsie[":1_1bis"]
["xnew"]),interpola_corsie[":1_1"]["interpolata"]
(interpola_corsie[":1_1"]["xnew"]),interpola_corsie[":1_1_straight_2"]["xnew"]))
y_1_1 = np.concatenate((interpola_corsie[":1_1_straight"]["interpolata"]
(interpola_corsie[":1_1_straight"]["xnew"]),interpola_corsie[":1_1bis"]
["xnew"],interpola_corsie[":1_1"]["xnew"],interpola_corsie[":1_1_straight_2"]
["interpolata"](interpola_corsie[":1_1_straight_2"]["xnew"])))

complete_paths = {}
complete_paths[":1_1_complete"] = {}
complete_paths[":1_1_complete"]["x"] = x_1_1
complete_paths[":1_1_complete"]["y"] = y_1_1

for i in complete_paths.keys():
    complete_paths[i]["distance_column"] = [0]

# Corrections needed for all trajectories
for i in complete_paths.keys():
    for j in range(1, len(complete_paths[i]["x"])):
        distance = math.sqrt((complete_paths[i]["x"][j]-
            complete_paths[i]["x"][j-1])**2 + (complete_paths[i]["y"][j] -
            complete_paths[i]["y"][j-1])**2)

        if i == ":1_1_complete":
            if complete_paths[i]["y"][j] <= -8.53129 and
                complete_paths[i]["y"][j] >= -33.8372:
                distance = distance*1.0070 # controllare

        cum_distance = complete_paths[i]["distance_column"][j-1] + distance
        complete_paths[i]["distance_column"].append(cum_distance)
    complete_paths[i]["distance_column"] =
        np.asarray(complete_paths[i]["distance_column"])

starting_points = {}
starting_points["s"] = [10.7470,-174.6645]
starting_points["e"] = [82.6385,95.8339]
starting_points["n"] = [-71.8703,125.9246]
starting_points["w"] = [-134.1964,-141.5983]

# %%

# buffer per media mobile su dati veicoli da SUMO
buffer={'buffer_x': {},
        'buffer_y': {},
        'buffer_delta': {},
        'n_elementi': {}}

# inizializzo varibile di archiviazione dati
stringa={}
for vei_id in scenario['id_vei_worldsim']:
    stringa[vei_id]={'x': 0., # posizione x del veicolo letta da SUMO
        (riferimento SUMO)
                     'y': 0., # posizione y del veicolo letta da SUMO
                        (riferimento SUMO)
                     'angle': 0., # angolo di ibardata del veicolo letto da SUMO
                        (convenzioni SUMO), in gradi
                     'distance':0.,#distanza percorsa cdal veicolo
                     'v': 0., # modulo della velocit del veicolo letta da SUMO
                        (valore inviato)
                     'RoadID': 0., # ID della corsia su cui si trova il veicolo
                     'x_interp': 0., # posizione x del veicolo interpolata
                        (riferimento SUMO)
                     'y_interp': 0., # posizione y del veicolo interpolata
                        (riferimento SUMO)
                     'angle_interp': 0., # angolo di ibardata del veicolo
                        calcolato dalle posizioni (convenzioni SUMO), in gradi
                     'delta': 0., # angolo di sterzo stimato del veicolo, in
                        radianti
```

```
                   'delta_mediato': 0., # angolo di sterzo stimato del veicolo e mediato sul buffer, in radianti (valore inviato)
                   'x_asse': 0., # posizione x della sala anteriore del veicolo (riferimento SUMO)
                   'y_asse': 0., # posizione y della sala anteriore del veicolo (riferimento SUMO)
                   'x_ws': 0., # posizione x della sala anteriore del veicolo (riferimento worldsim, valore inviato)
                   'y_ws': 0., # posizione y della sala anteriore del veicolo (riferimento worldsim, valore inviato)
                   'angle_ws': 0.} # angolo di imbardata del veicolo (convenzione worldsim, valore inviato), in radianti

# campi della variabile "stringa" che si vogliono salvare nel file di testo
da_salvare=['x',
           'y',
           'distance',
           'v',
           'RoadID',
           'x_interp',
           'y_interp',
           'angle_interp',
           'delta',
           'delta_mediato',
           'x_asse',
           'y_asse',
           'x_ws',
           'y_ws',
           'angle_ws']

# nome file di salvataggio
nome_file_salvataggio='dati_new.txt'

# Debug senza connessione
test = {'Test': True, # se True non viene attivata la connessione e si pu testare solo la parte di SUMO, se False va normalmente
        'Cicli': 20000} # numero cicli di test

# INIZIO DEFINIZIONE FUNZIONI
```

```

def interpolazione_corsie():
    # interpolo la posizione attuale del veicolo in modo che le curve siano pi
       fluide.
    # al momento sono disponibili solo il rettilineo (nessuna interpolazione)
       o l’interpolazione delle curve come archi di circonferenza analitici

    for vei_id in scenario['veicoli_attivi']:

        distance = stringa[vei_id]['distance'] +
           scenario['veicoli'][vei_id]["init_distance"]
        chosen_path = scenario['veicoli'][vei_id]['id_tra']

        xnew = np.interp(distance,
           complete_paths[chosen_path]["distance_column"],
           complete_paths[chosen_path]["x"])
        ynew = np.interp(distance,
           complete_paths[chosen_path]["distance_column"],
           complete_paths[chosen_path]["y"])

        stringa[vei_id]['x_interp']= xnew
        stringa[vei_id]['y_interp']= ynew
        stringa[vei_id]['delta'] =
           np.arctan2(scenario['veicoli'][vei_id]["points_couples"][1] -
           stringa[vei_id]['y_interp'],
           scenario['veicoli'][vei_id]["points_couples"][0] -
           stringa[vei_id]['x_interp'])

        # Considerando che arctan2 va da -180 a 180 --> riporto a da -90 a 90
        if scenario['veicoli'][vei_id]["points_couples"][1] -
           stringa[vei_id]['y_interp'] < 0 and
           scenario['veicoli'][vei_id]["points_couples"][0] -
           stringa[vei_id]['x_interp'] < 0:
               stringa[vei_id]['delta'] = stringa[vei_id]['delta'] + np.pi

        if scenario['veicoli'][vei_id]["points_couples"][1] -
           stringa[vei_id]['y_interp'] >= 0 and
           scenario['veicoli'][vei_id]["points_couples"][0] -
           stringa[vei_id]['x_interp'] < 0:
            stringa[vei_id]['delta'] = stringa[vei_id]['delta'] - np.pi

        # Prendo il valore assoluto dell'angolo ottenuto
        stringa[vei_id]['delta'] = abs(stringa[vei_id]['delta'])

        value = stringa[vei_id]['delta']
        stringa[vei_id]['delta'] = abs(stringa[vei_id]['delta'] -
            scenario['veicoli'][vei_id]['delta_prec'])
        scenario['veicoli'][vei_id]['delta_prec'] = abs(value)

        scenario['veicoli'][vei_id]["points_couples"][0] =
            stringa[vei_id]['x_interp']
        scenario['veicoli'][vei_id]["points_couples"][1] =
            stringa[vei_id]['y_interp']



def esegui_step_SUMO(x, y, angle, speed):
    muovi_veicolo(scenario['id_ego'], x, y, angle, speed) # posiziono la ego
        dove desiderato


def leggi_veicolo():
    scenario['veicoli_attivi']=traci.vehicle.getIDList() # veicoli attualmente
        attivi nella simulazione SUMO
    scenario['veicoli_attivi']=set(scenario['veicoli_attivi']) # trasformo da
        tuple a set
    scenario['veicoli_attivi']=scenario['veicoli_attivi'].difference(
    {scenario['id_ego']}) # tolgo la ego
    scenario['veicoli_non_attivi']=scenario['id_vei_worldsim'].difference(
    scenario['veicoli_attivi']) # veicoli non attualmente attivi in SUMO

    for vei_id in scenario['veicoli_attivi']:
            traci.vehicle.subscribe(vei_id, [tc.VAR_POSITION, tc.VAR_SPEED,
                tc.VAR_ROAD_ID , tc.VAR_ANGLE, tc.VAR_DISTANCE])

    values = traci.vehicle.getAllSubscriptionResults()

    for vei_id in scenario['veicoli_attivi']:

        stringa[vei_id]['x']= values[vei_id][66][0]
        stringa[vei_id]['y']= values[vei_id][66][1]
        stringa[vei_id]['angle']= values[vei_id][67] # leggo l'angolo di
           imbardata
        stringa[vei_id]['v']= values[vei_id][64] # leggo la velocit del veicolo
        stringa[vei_id]['RoadID']= values[vei_id][80] # leggo il ROAD ID
        stringa[vei_id]['distance']= values[vei_id][132] # leggo distanza
           percorsa dal veicolo

def muovi_veicolo(vei_id, x, y, angle, speed):
    traci.vehicle.moveToXY(vei_id, "", -1, x, y, angle, keepRoute=2) #
       posizione desiderata della ego (keepRoute=2 impone la posizione a
       prescindere che sia sulla strada)
    traci.vehicle.setSpeed(vei_id, speed) # velocit ego (al momento  posta
       uguale a zero, va capito cosa fare)


def converti_coordinate_ego(dati):
    x = dati[1] + scenario['veicoli']['f_0.0_0']['L_bumper']*np.cos(dati[3]) -
       scenario['x_offset'] # riporto le varibili nel sistema SUMO e calcolo
       la posizione del centro del bumper anteriore
    y = dati[2] + scenario['veicoli']['f_0.0_0']['L_bumper']*np.sin(dati[3]) -
       scenario['y_offset']
    angle = -dati[3] / pi * 180 + scenario['yaw_offset'] # riporto l'angolo di
       imbardata nelle convenzioni di SUMO
    speed = 0 #dati[4]

    return x, y, angle, speed

def converti_coordinate_veicoli_worldsim():
    for vei_id in scenario['veicoli_attivi']:
        # riporto le coordinate x e y alla sala anteriore
        sigma = (-stringa[vei_id]['angle_interp'] + scenario['yaw_offset']) /
           180 * pi # riporto l'angolo dalle convenzioni di SUMO a quelle
           standard usate anche da Worldsim
        x = stringa[vei_id]['x_interp'] -
           scenario['veicoli'][vei_id]['L_bumper'] * np.cos(sigma) # calcolo
           le coordinate del centro della sala anteriore a partire dal centro
```

```
          del bumper anteriore
        y = stringa[vei_id]['y_interp'] -
          scenario['veicoli'][vei_id]['L_bumper'] * np.sin(sigma)

        stringa[vei_id]['x_asse']=x
        stringa[vei_id]['y_asse']=y
        stringa[vei_id]['x_ws']=x + scenario['x_offset']
        stringa[vei_id]['y_ws']=y + scenario['y_offset']
        stringa[vei_id]['angle_ws']=sigma

def crea_socket_e_connessione():
    indirizzo_self = (impostazioni_UDC['UDC_IP_SELF'],
        impostazioni_UDC['UDC_PORT_SELF'])
    indirizzo_client={}
    for i in impostazioni_UDC['lista_porte_CUNCURRENT']: # creo per ogni
        veicolo una tuple con IP cuncurrent e porta dedicata ala veicolo
        indirizzo_client[i]=(impostazioni_UDC['UDC_IP_CUNCURRENT'],
            impostazioni_UDC['lista_porte_CUNCURRENT'][i])

    # creo un socket per inviare dati per ogni veicolo e lo chiamo server
        (questa macchina)
    s_server = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
    s_server.bind(indirizzo_self)
    s_server.settimeout(impostazioni_UDC['time_out'])

    # creo un socket per ricevere i dati e lo chiamo client (cuncurrent)
        QUESTE ANDRANNO LINKATE AI VEICOLI
    s_client={}
    for i in impostazioni_UDC['lista_porte_CUNCURRENT']:
        s_client[i]=socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
        s_client[i].settimeout(impostazioni_UDC['time_out'])

    print('UP sockets opened')
    print(s_server)
    for i in s_client:
        print(s_client[i])

    return s_server, s_client, indirizzo_client
```

```
def ricevi_dati(s):
    data, address = s.recvfrom(impostazioni_UDC['buffer_size']) # leggo tutto il buffer
    converteddata = struct.unpack(impostazioni_UDC['codifica_ricezione'], data) # converto i dati (attenzione a big o little endian)

    return converteddata

def invia_dati(s_client, indirizzo_client):
    for vei_id in scenario['id_vei_worldsim']:
        da_inviare=[stringa[vei_id]['x_ws'],
                    stringa[vei_id]['y_ws'],
                    stringa[vei_id]['angle_ws'],
                    stringa[vei_id]['v'],
                    stringa[vei_id]['delta_mediato']]
        msg1 = struct.pack(impostazioni_UDC['codifica_invio'], *da_inviare)
        s_client[vei_id].sendto(msg1, indirizzo_client[vei_id]) # invio i dati, ogni porta ha lo stesso id del veicolo corrispondente

def scorri_buffer(x,v): # butta il primo elemento del buffer, fa scorrere indietro gli altri elementi e mette il dato nell'ultimo elemento del buffer
    v[0:-1]=v[1:]
    v[-1]=x
    return v

def ciclo_SUMO(x, y, angle, speed):
    # passo di simulazione SUMO - muove solamente il veicolo
    esegui_step_SUMO(x, y, angle, speed)

    # leggo i dati dei veicoli attivi
    leggi_veicolo()
    # interpolo i dati da SUMO
    interpolazione_corsie()

    # aggiorno i buffer e calcolo imbardata e angolo sterzo
    for vei_id in scenario['veicoli_attivi']:
        buffer['buffer_x'][vei_id]=scorri_buffer(stringa[vei_id]
        ['x_interp'],buffer['buffer_x'][vei_id])
        buffer['buffer_y'][vei_id]=scorri_buffer(stringa[vei_id]
        ['y_interp'],buffer['buffer_y'][vei_id])
        buffer['buffer_delta'][vei_id]=scorri_buffer(stringa[vei_id]
        ['delta'],buffer['buffer_delta'][vei_id])
        buffer['n_elementi'][vei_id]=buffer['n_elementi'][vei_id]+1

        # imbardata
        if buffer['n_elementi'][vei_id]>interpola_corsie["buffer"]
        ['buffer_imbardata']:
            if stringa[vei_id]['v']>1e-3:
             # se il veicolo  fermo non aggiorno l'angolo di imbardata perch
                non riuscirei a calcolarlo
                # calcolo l'angolo di imbardata come variazione della direzione
                   del vettore spostamento
                stringa[vei_id]['angle_interp']=np.arctan2(buffer['buffer_x']
                [vei_id][-1]-buffer['buffer_x'][vei_id][0],
                                                        buffer['buffer_y']
                                                        [vei_id][-1]-
                                                        buffer['buffer_y']
                                                        [vei_id][0])*180/pi
        else:
            stringa[vei_id]['angle_interp']=scenario['veicoli'][vei_id]
            ['angle_iniziale']
        # angolo di sterzo mediato sul buffer (serve per evitare salti nella
           visualizzazione)
        stringa[vei_id]['delta_mediato']=np.mean(buffer['buffer_delta'][vei_id])

    # impongo i dati dei veicoli non attivi
    for vei_id in scenario['veicoli_non_attivi']:
        for i in stringa[vei_id].keys():
            stringa[vei_id][i]=0.

def scivi_dati_su_file(num): # salvo su un file la variabile 'stringa'
    v=''
    for vei_id in scenario['id_vei_worldsim']:
        v=v+vei_id+' '
        for var in da_salvare:
            v=v+str(stringa[vei_id][var])+' '
    v=v+'\n'
    num.write(v)
```

```

for vei_id in scenario['id_vei_worldsim']:
        # inizializzo i buffer per tutti i veicoli
        buffer['buffer_x'][vei_id]=np.zeros(interpola_corsie["buffer"]
        ['buffer_imbardata'])
        buffer['buffer_y'][vei_id]=np.zeros(interpola_corsie["buffer"]
        ['buffer_imbardata'])
        buffer['buffer_delta'][vei_id]=np.zeros(interpola_corsie["buffer"]
        ['buffer_sterzo'])
        buffer['n_elementi'][vei_id]=0

        # inizializzo l'angolo di imbardata
        stringa[vei_id]['angle_interp']=stringa[vei_id]['angle']

# INIZIO COMUNICAZIONE
#%%

# Setting the port on the second client
cmd = "ps aux | grep SUMO"
stdoutdata = subprocess.getoutput(cmd)
count = 0
index = 0
valore = 1
for i in stdoutdata.split():
    if i == "--remote-port" and valore == 1:
        index = count
        valore = 2
    count +=1

print("--remote-port: " + stdoutdata.split()[index+1])
#PORT = int(stdoutdata.split()[index+1])
PORT = 53141

# Insert the local host ip
traci.init(PORT,tc.DEFAULT_NUM_RETRIES,"127.0.0.1")
traci.setOrder(2) # number can be anything as long as each client gets its own
    number

for vei_id in scenario['id_vei_worldsim']:

        # inizializzo i buffer per tutti i veicoli
        buffer['buffer_x'][vei_id]=np.zeros(interpola_corsie["buffer"]
        ['buffer_imbardata'])
        buffer['buffer_y'][vei_id]=np.zeros(interpola_corsie["buffer"]
        ['buffer_imbardata'])
        buffer['buffer_delta'][vei_id]=np.zeros(interpola_corsie["buffer"]
        ['buffer_sterzo'])
        buffer['n_elementi'][vei_id]=0

        # inizializzo l'angolo di imbardata
        stringa[vei_id]['angle_interp']=stringa[vei_id]['angle']

if not test['Test']:
    # inizializzo connessione.
    # server: questo pc
    # client: cuncurrent
    s_server, s_client, indirizzo_client = crea_socket_e_connessione()

ciclo = 0 # ciclo di calcolo
flag = 1 # flag   il canale VI_Drivesim.Outputs.Vicrt.Satus (3=pausa,
    1=simula, altro=stop)
marker = 0
initial_angle = 0
if impostazioni_SUMO['salva_risultati']:
    num = open(nome_file_salvataggio, 'w') # creo il file di salvataggio,
        sovrascrive senza chiedere


# Inizializzo posizione della Ego
x = 88.500
y = 96.859
angle = 222.897
speed = 0
valore = 0
tempi = []
steps = []
tempo_tot = 0

```

```python
while traci.simulation.getMinExpectedNumber() >= 0 and flag == 1 or flag == 3:

    ciclo = ciclo + 1 # conto i cicli
    inizio = time() # tengo traccia della durata di ciascun ciclo
    if test['Test']: # se sto facendo il test del solo SUMO, rimango nel ciclo per il numero di cicli di calcolo richiesti nei dati
        if ciclo<test['Cicli']:
            flag=1
        else:
            flag=2

    else:
        dati_ego = ricevi_dati(s_server) # Ricevo i dati della ego dal cuncurrent
        flag = dati_ego[0] # flag per vedere quando uscire dal ciclo e chiudere il server. Il primo valore ricevuto  lo status
        x, y, angle, speed = converti_coordinate_ego(dati_ego) # Trasformo le coordinate della ego dal sistema worldsim a quello SUMO


    if flag == 1 and ciclo >= 100: # se la simulazione  attiva sul cuncurrent eseguo il ciclo di SUMO, atrimenti non faccio nulla
        ciclo_SUMO(x, y, angle, speed) # Sposto solamente il veicolo
        converti_coordinate_veicoli_worldsim() # converto i dati da SUMO nelle convenzioni di worldsim

        if not test['Test']: # se non sto facendo il test del solo SUMO, trasmetto i dati
            invia_dati(s_client, indirizzo_client)

        if impostazioni_SUMO['salva_risultati']:
            scivi_dati_su_file(num) # salvo i dati del ciclo nel file di testo

    if impostazioni_SUMO['informazioni_a_video']==1:
        print(ciclo, flag, stringa, time() - inizio)
    elif impostazioni_SUMO['informazioni_a_video']==2:
        print(ciclo, time() - inizio)
        tempi.append(time()- inizio)
        steps.append(ciclo)
        tempo_tot += time() - inizio


    if flag == 1 or flag == 3:
        traci.simulationStep()


if impostazioni_SUMO['salva_risultati']:
        num.close() # chiduo il file di testo di salvataggio dei dati

if not test['Test']:
    # termino server e client
    s_server.close()
    for s in s_client:
        s_client[s].close()


traci.close(False)
```

List of Figures

List of Tables

Nomenclature

Symbol	Explanation
3GPP	Third Generation Partnership Project
5G	Fifth Technology
AI	Artificial Intelligence
AIF	Artificial Intelligence Framework
AV	Autonomous vehicle
BSM	Basic Safety Message
CAM	Cooperative Awareness Message
CCAM	Connective, Cooperative and Automated Mobility
DENM	Decentralised Environmental Notification Messages
DRL	Deep Reinforcement Learning
DS	Driving simulator
DSRC	Dedicated short-range communications
eMBB	enhanced Mobile Broadband
FL	Federated learning
GSM	Global System for Mobile communications
I/O	Input/Output
I2V	Infrastructure-to-vehicle communication
ICD	Inscribed Circle Diameter
ITS	Intelligent Transportation Systems
MEC	Multi-access Edge Computing
ML	Machine Learning
MOE	Measure of Effectiveness
MPC	Model Predictive Control
MTS	Microscopic traffic simulator

ProSe	proximity service
RL	Reinforcement Learning
SAE	Society of Automotive Engineers
SAECU	Safety Application Electronic Control Unit
SDN	Software-defined networking
SUMO	Simulation of Urban MObility
TCP	Transmission control protocol
UDP	User datagram Protocol
URLCC	Ultra-Reliable Low Latency Communication
V2I	Vehicle-to-infrastructure communication
V2V	Vehicle-to-vehicle communication
V2X	vehicle-to-everything
VICN	Vehicle's Internal Communication Network