Autonomous Vehicle Driving

Final Project – 2022/23

Camilla Spingola	0622701698	c.spingola@studenti.unisa.it
Mattia Marseglia	0622701697	m.marseglia1@studenti.unisa.it
Vito Turi	0622701795	v.turi3@studenti.unisa.it
Sica Ferdinando	0622701794	f.sica24@studenti.unisa.it

UNIVERSITÀ DEGLI STUDI
DI SALERNO

Table of Contents

Introduction

Autonomous driving represents a transformative technology poised to revolutionise the way we perceive transportation. With the promise of safer roads, increased efficiency, and enhanced mobility, autonomous vehicles have captured the imagination of researchers, engineers, and enthusiasts alike. While fully autonomous vehicles capable of navigating all road conditions and scenarios without human intervention are not yet a reality, significant progress has been made towards achieving this ambitious goal. The field of autonomous driving encompasses a broad range of technologies, spanning from perception and decision-making algorithms to control systems and human-machine interfaces. Sensing technologies such as LiDAR (Light Detection and Ranging), cameras, and radar that play a crucial role in capturing the surrounding environment, while machine learning algorithms process and interpret this wealth of sensor data to make informed decisions in real-time.

One of the key challenges in autonomous driving is the development of robust perception systems that can accurately detect and classify objects in the environment. This includes identifying and tracking vehicles, pedestrians, traffic signs and traffic lights, among other elements. By leveraging deep learning techniques, computer vision algorithms have made significant strides in achieving high accuracy and reliability in object recognition, enabling vehicles to make informed decisions based on their surroundings.
Another critical aspect of autonomous driving is path planning and control. Once the perception system has gathered information about the environment, the vehicle must determine the optimal trajectory to follow and execute precise manoeuvres accordingly. This involves considering factors such as traffic rules, road geometry and dynamic obstacles to ensure safe and efficient navigation. Advanced control algorithms, such as model predictive control and reinforcement learning, have been employed to enable autonomous vehicles to navigate complex scenarios and adapt to changing conditions.

The development and testing of autonomous driving algorithms necessitate extensive validation and verification processes. While real-world testing is crucial, it can be time-consuming, expensive and potentially dangerous. This is where simulators like CARLA became important, providing a realistic virtual environment and allowing researchers to iterate rapidly, evaluate different algorithms, and test edge cases without putting human lives at risk or incurring significant costs. CARLA is an open-source simulator specifically designed for autonomous driving research. CARLA offers a wide range of configurable parameters, including weather conditions, traffic density, and sensor noise, allowing users to create diverse and challenging scenarios. Moreover, CARLA's open-source nature encourages collaboration and knowledge sharing within the autonomous driving community. Researchers can build upon existing work, contribute to the simulator's development, and collectively advance the field of autonomous driving.

To ensure consistency and fairness, the CARLA Leaderboard employs standardized metrics for evaluating autonomous driving performance. These metrics cover various aspects, including perception accuracy, decision-making quality, trajectory planning and overall driving behaviour. By utilizing these metrics, participants can gain valuable insights into their algorithms' strengths and weaknesses, enabling them to refine and improve their autonomous driving solutions.

Autonomous driving represents a paradigm shift in transportation, holding the potential to enhance road safety, improve traffic efficiency, and revolutionize personal mobility. The CARLA simulator serves as a vital tool in the development and evaluation of autonomous driving algorithms.

Executive Summary

Project's purpose

The goal of the project is to develop an autonomous driving algorithm capable of successfully completing the scenarios included in the CARLA Leaderboard. Specifically, five main routes have been selected as the scoring routes, while an additional five extra routes have been chosen to test the product developed and address key challenges. These extra routes also provide optional situations that can be handled by the algorithm but are not necessary for the main goal. The selection of these routes aims to create a comprehensive evaluation framework that covers a wide range of driving scenarios. By including both the main routes, which represent typical driving scenarios and the extra routes, which introduce additional complexities and challenges, the project aims to thoroughly assess the algorithm's performance and identify areas for improvement.

To be more precisely the main routes serve as the core evaluation benchmarks, encompassing diverse environments such as urban areas, highways, and intersections. By successfully navigating through these main routes, the algorithm demonstrates its competency in handling common driving scenarios.

In contrast, the extra routes offer a more specialized evaluation of the algorithm's capabilities. These routes introduce unique challenges, such as complex traffic patterns or specific driving regulations. By including these extra routes, the project aims to test the algorithm's adaptability and robustness in handling unexpected and less common situations. This helps uncover potential weaknesses and areas requiring further refinement. While not mandatory for evaluation, successfully managing these optional situations can demonstrate the algorithm's advanced capabilities and its potential for real-world applications.

Operational design domain (ODD)

The Operational Design Domain (ODD) is a crucial aspect to consider when developing autonomous vehicle systems. It defines the specific operating conditions, environments, and scenarios in which the autonomous vehicle is designed to function safely and reliably. In our project, the ODD is strongly related to the driving scenarios analyzed in the assigned route. The assigned routes encompass a combination of urban, suburban and highway roads, presenting a diverse set of challenges and considerations. Urban streets present tight turns, intersections, and varying speed limits. Suburban roads typically have higher speed limits. The assigned routes expose the autonomous vehicle to a range of environmental conditions. Those conditions were: sunny, night and raining, also during the night. Regarding the specific conditions of night and day, they do not significantly impact the performance of our algorithm because we rely on perfect perception, acquiring data directly from the simulator, which provides precise information about the position and velocity of every object, barring any issues related to the simulator itself. Therefore, our algorithm remains unaffected by variations in lighting conditions. On the other hand, when it comes to weather conditions, our developed controller has not encountered any significant challenges in navigating through different environments. As a result, we have not implemented any specific adaptations or variations in our algorithm to address rainy weather situations. The assigned routes experience varying traffic patterns. Urban streets are characterized by heavy traffic, frequent stops and interactions with pedestrians and cyclists. Suburban roads typically have moderate traffic volumes and may encounter frequently hard breakings and some parked vehicles at the edge of the road. Both involve variable-moving traffic, with considerations for lane changing, as for example in overtaking, and maintaining safe distances from other vehicles. Each road type within the assigned routes has specific speed limitations. Urban streets have lower speed limits due to the presence of pedestrians and potential congestion. Suburban roads typically have higher speed limits, balancing safety and efficient travel. The assigned routes are equipped with a variety of road signage, including stop signs, traffic lights, speed limit signs and warning signs that must be respected. The assigned

routes feature different infrastructure elements, such as pedestrian crossings, bikers crossing, sidewalks, parked vehicles, rough road, dedicated lanes, traffic warnings and construction cone. Synthesizing, the operational design domain of our autonomous vehicle project is shaped by the characteristics of the assigned routes.

We can confidently say that our self-driving vehicle is capable of operating on different roads even in rainy conditions; it has not been tested on unpaved or country roads where stability may be compromised. Any static objects on the road are always recognized so that they can be overtaken; the junctions were managed and carried out in a safe way. In terms of environmental conditions, sunny, rainy and nocturnal environments were tested. Road signs are partially used; in the routes on which we tested the vehicle there weren't large quantities of road signs so the only sign that is actually used are the stop sign and the traffic lights; for the speed limits we take this information directly from CARLA from the road we are traveling on and as regards the warning signs for any obstructions on the road, we go directly to see the distance from the objects on the road without analysing the information contained in the road sign. The road conditions on which the autonomous driving algorithm was tested are varied, but it is important to specify that the robustness of our algorithm is as minor as the test routes are different from the one we have used.

Simulation time step and client-server synchrony

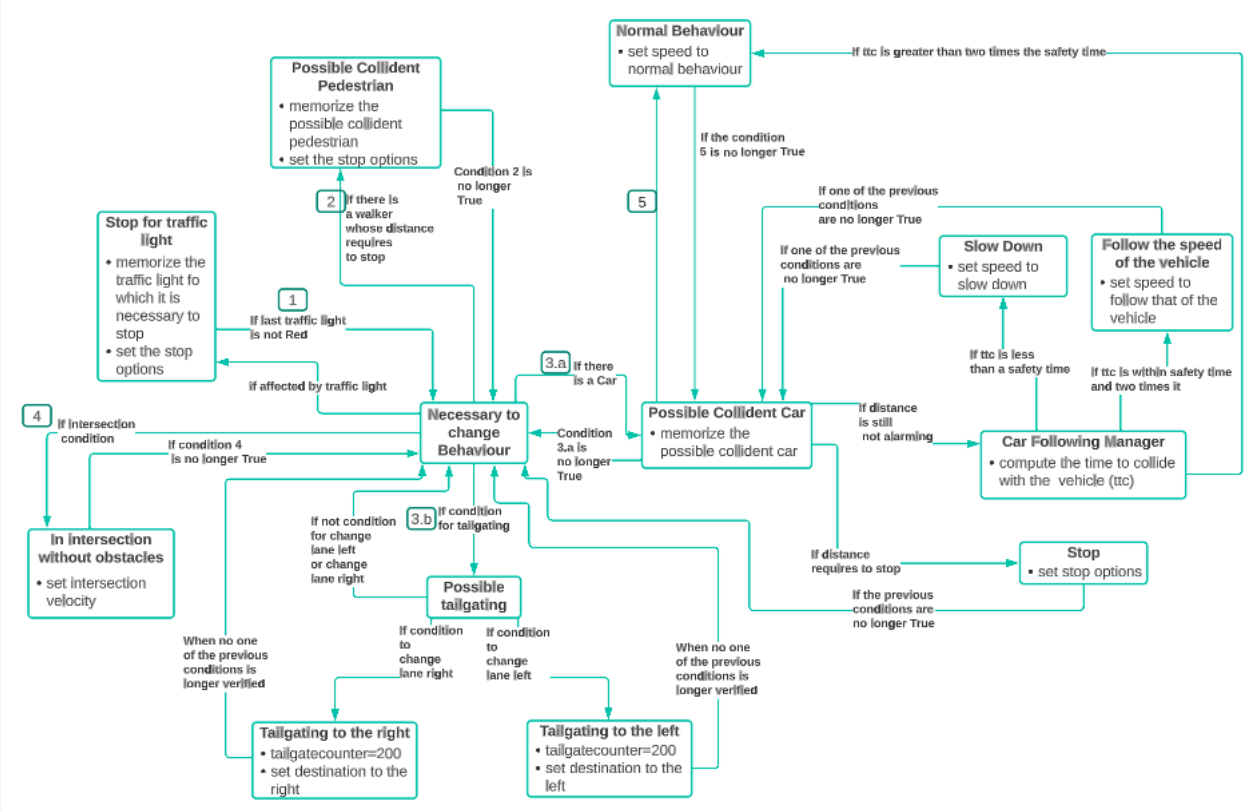
Simulation time step and synchronization play a crucial role in the accurate and reliable simulation of autonomous vehicle behaviour in the Carla platform. In this section, we will discuss the selection of an appropriate simulation time step and synchronization mode between the client and server components in Carla. The simulation time step determines the granularity at which the simulation progresses in Carla. It represents the discrete interval at which updates, such as vehicle dynamics, perception and control are calculated and applied. Choosing an appropriate time step is crucial for achieving a balance between computational efficiency and accuracy of the simulated vehicle dynamics and its interactions with the environment. In the Carla platform, the simulation consists of two main components: the client, which typically runs the autonomous driving algorithms, and the server, responsible for the simulation environment. Synchronization between the client and server ensures that the simulation progresses consistently and accurately. The synchronization mechanism enables the exchange of data and commands between the client and server, ensuring that the client receives up-to-date information about the simulation environment, including the positions and states of other vehicles, traffic signals and pedestrians. This allows the client to make informed decisions based on the most recent state of the simulation. Choosing an appropriate synchronization mode is essential to minimize delays and ensure the client's perception and control algorithms can operate in real-time. Additionally, synchronization influences the fidelity of the simulation, as it affects the accuracy of the perceived environment and the responsiveness of the controlled vehicle. In our project, we carefully considered the simulation time step and synchronization mode in Carla to strike a balance between computational efficiency and simulation accuracy. The parameters for choosing the time step and the synchronization mode between client and server could be set in the CARLA leaderboard, in the ***leaderboard_evaluator*** file, which manages the loading of a new CARLA world and the sending of data. For the simulation step, we have chosen to have a fixed time-step because when the time step is variable the update of the world is carried out when the data from the server is available, therefore, precisely, in a variable manner and completely dependent on the specifications of the machine on which it is running, creating anomalous and non-repeatable behaviours. This thing can cause problems: if we design a control algorithm we expect to receive data every certain time, therefore we cannot guarantee this "time" to do the control. For this reason the fixed time step is used, so although the server proceeds slower than the real world, the system adapts the simulation to achieve the desired behaviours. So we basically tell the

server that we need the data every fixed time step, that in our case is 0.05 seconds. Once this is set, on every call to the server, even if it's done every 10ms, the server's (world's) delay won't affect our experimentation. Regarding the client-server synchrony, if the operation is asynchronous, the client and server are independent. If the control signal doesn't come, the world moves forward anyway so the server will move forward. Synchronous mode is where the server after producing the results for the client, will stop and wait for the control signal. This second mode is slower but allows us to eliminate the processing time of the client and server machines, so we have chosen this one.

Baseline

As an initial starting point of the project, we were provided with a baseline that included the essential functionalities for navigating the routes, albeit without handling most of the complex driving situations. This baseline algorithm was capable of basic operations such as stopping and starting at traffic lights (yellow excluded), following vehicles ahead without collision (if they are detected on the same road, this check is not always enough) and managing pedestrian crossings when they were perfectly in front of the vehicle and detected on time. However, it lacked the ability to handle several critical scenarios, including bikers, overtaking, intersections, stop signs, narrow passages, emergency braking and obstacles on the road.

In the following it is possible to see a high-level schema representing the logical flow of different situations managed into the baseline. This schema served us to understand the conditions checked into this first algorithm and also the consequent implementations present.



It is important to clarify that everything has been graphed with a level of detail such as to be a summary picture of the implementation provided to us, in order to start diagnosing the issues with the baseline through code analysis, before observing the execution of different routes.

In the diagram, next to the main conditions, there are some incremental IDs. These IDs, associated with a specific condition, ensure that the transition along the specific arrow can only occur if the conditions

associated with lower numerical IDS are not satisfied. Therefore, they serve to establish a hierarchical order among the possible conditions that denote a behavioral change in our vehicle.

The baseline algorithm served as a foundation for our development process, providing a starting point from which we could build and enhance the autonomous driving capabilities. Our goal was to improve the algorithm's performance and extend its functionality to effectively handle the diverse challenges presented by the CARLA Leaderboard scenarios. One of the key areas of improvement was the detection and interaction with bikers. The baseline algorithm did not have the capability to accurately detect bikers on the road. Our goal was to enhance the “perception” system to effectively detect and respond to bikers, ensuring safe navigation and smooth interaction with these vulnerable road users.
Another critical aspect we aimed to address was the handling of overtaking maneuvers. The baseline algorithm did not include the ability to overtake. Developing this capability required advanced perception, decision-making and trajectory planning algorithms to assess the surrounding traffic, identify appropriate opportunities for overtaking, and execute the maneuver smoothly, efficiently and safely.
Furthermore, intersections and stop signs presented additional complexities. The baseline algorithm did not have the necessary functionality to handle these scenarios, including correctly interpreting traffic rules and making safe decisions when approaching and crossing intersections. Our focus was on implementing robust algorithms that could analyze the traffic conditions, interpret the intentions of other vehicles and navigate intersections while ensuring safety and adherence to traffic regulations.
Narrow passages and tight spaces, such as road construction zones or narrow lanes, were also areas that required improvement. The baseline algorithm lacked the capability to navigate through these challenging environments, which required precise control and maneuvering.
Emergency braking and obstacle avoidance were crucial aspects that needed to be addressed as well. The baseline algorithm did not possess the ability to detect and respond to sudden obstacles or hazardous situations on the road, which could lead to accidents or collisions. Developing an effective emergency braking system and implementing algorithms for efficient obstacle detection and avoidance were fundamental objectives of our project.

So the initial baseline provided us a starting point, but it lacked the necessary capabilities to handle a wide range of complex driving scenarios. Our project focused on enhancing the algorithm's functionalities to address the challenges related to bikers, overtaking, intersections, stop signs, narrow passages, emergency braking and obstacle avoidance. By improving these aspects, we aimed to create a more advanced and capable autonomous driving algorithm that could successfully navigate the CARLA Leaderboard routes and handle the various situations encountered on the road. A detailed and better explained analysis of the routes to establish the problems and the module that have to be implemented to increase the algorithm’s performance is made in the next paragraphs relative to the route analysis.

Routes analysis

In this part of the report, we present our scores (driving score, route completion and infraction penalty) obtained on the basic and extra routes; we will analyse the score of the baseline with the controlled provided in it. The scope is to evaluate the performance and the critical issues of the baseline in order to apply countermeasures to mitigate these problems. Once we analysed the basic level of functionalities given by the baseline, we then focused on enhancing the system's performance through fine-tuning of the controller and the autonomous driving algorithms.

BASELINE – BASIC CONTROLLER – BASIC ROUTES

	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4
Route Completion	100%	12.14%	100%	100%	70.97%
Outside Route Lanes	0.36%	0%	0%	0%	0%
Collisions	20	1	1	0	3
Running Red Light	0	0	0	0	0
Running Stop	1	0	1	0	1
Min Speed	100.2%	100%	112.71%	114.88%	106.43%
In Route	✓	✓	✓	✓	✓
Agent Blocked	✓	✓	✓	✓	×
Scenario Timeout	9	1	0	0	1
Timeout	✓	×	✓	✓	✓

GLOBAL DRIVING SCORE: 32.26

GLOBAL ROUTE COMPLETION: 76.62

GLOBAL INFRACTION PENALTY: 0.41

In this first table we have the results with the baseline on the basic routes with the controller given with the project; this controller gives a lot of problems that are being analysed in the next tables, where we explain every single infraction noted during this first simulation, the cause of the infraction and a possible solution that has to be implemented during the development of the project.

ROUTE 0 PROBLEMS

	Number	Why	Solution
Collisions with bikers	3	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with stationary vehicles	7	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	4	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with static object	2	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy

Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Collisions with moving vehicles	4	Some curves are done too wide, so we collide with the vehicles on the other lane; another case of collision is that during a braking our vehicle doesn't swerve and collide with vehicles	Correct the controller to avoid wide curves and implement a different behaviour when the vehicle has to brake, in a way that it continues to swerve
Outside route percentage	0.36%	The invasion of other lane is caused by a not tuned controller	Implement a good control system

DRIVING SCORE: 8.8e-05

ROUTE COMPLETION: 100

INFRACTION PENALTY: 1e-6

ROUTE 1 PROBLEMS

	Number	Why	Solution
Collisions with static objects	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Collisions with bikers	1	Bikers not recognized on the road	Recognize them and implement a surpass policy

DRIVING SCORE: 5.5237

ROUTE COMPLETION: 12.14

INFRACTION PENALTY: 0.455

ROUTE 2 PROBLEMS

	Number	Why	Solution
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Collisions in junction	1	In the junction we pass when it's not possible and collide with a vehicle	Implement a good logic for the junctions

DRIVING SCORE: 47.21664

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.472166

ROUTE 3 PROBLEMS

	Number	Why	Solution

DRIVING SCORE: 100.0

ROUTE COMPLETION: 100

INFRACTION PENALTY: 1.0

ROUTE 4 PROBLEMS

	Number	Why	Solution
Collisions with moving vehicles	2	Some curves are done too wide, so we collide with the vehicles on the other lane; the other case of collision is that during a braking our vehicle doesn't swerve and collide with vehicles	Correct the controller to avoid wide curves and implement a different behaviour when the vehicle has to brake, in a way that it continues to swerve
Collisions with stationary vehicles	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal

DRIVING SCORE: 8.584531

ROUTE COMPLETION: 70.97

INFRACTION PENALTY: 0.12096

Basic routes analysis summary

After this first step of analysis, we have seen that the first thing to do to achieve better results is to implement a well-tuned controller; it can give the possibility to avoid a lot of collision made into these basic routes. In the specific we can summarize the solutions that have to be implemented:

- Tune the controllers to avoid wide curves and have a more reactive behaviour.
- Recognize the bikers and eventually their surpass policy.
- Recognize vehicles at the edge of the road and eventually their surpass policy.
- Recognize static object on the road and eventually their surpass policy.
- Recognize stop signals and implement a policy to stop and wait the necessary time.
- Implement a way to swerve also when the vehicle is braking.
- Manage the narrow lanes.

- Implement a good logic for the junctions.

The implementation of a good controller can avoid collisions with moving vehicles during a curve and a more robust behaviour in the junction. Recognition and overtaking of the bikers avoids every collision with them and also a possible scenario timeout; obviously a surpass policy has to be implemented in a safe-way respecting the speed limits of the road. Recognition of vehicles at the edge of the road is really important and constitutes the cause of a lot of collisions had during these simulations. Static objects also have to be recognized to avoid the collision and eventually surpassed. The stop signals are not recognized and constitutes a small part of the obtained infractions and at the end the swerve during a brake is necessary to avoid strange behaviours and an invasion of the other lanes.

BASELINE – BASIC CONTROLLER – EXTRA ROUTES

In this part of the report we 'll do an analysis of the optional routes 8,9,10,11 and 12. Obviously these routes are not too important for the project, so we will see the problems that are present using the basic controller and if there are "new" problems different from the ones resulting from the basic routes that, if they will be fixed, will constitute a new "basis" on which start for analysing the new problems of the optional routes.

	ROUTE 8	ROUTE 9	ROUTE 10	ROUTE 11	ROUTE 12
Route Completion	76.32%	79.15%	34.41%	32.1%	100%
Outside Route Lanes	0.03%	0%	0.6%	0.9%	0%
Collisions	13	4	4	2	3
Running Red Light	0	0	0	0	0
Running Stop	1	0	0	1	0
Min Speed	77.87%	142.86%	96.48%	101.38%	68.06%
In Route	✓	✓	✓	✓	✓
Agent Blocked	✓	×	×	×	✓
Scenario Timeout	5	2	0	0	2
Timeout	×	✓	✓	✓	✓

GLOBAL DRIVING SCORE: 5.91

GLOBAL ROUTE COMPLETION: 64.40

GLOBAL INFRACTION PENALTY: 0.14

In this table we have the results with the baseline on the extra routes with the controller given with the project; we have seen new type of problems, also with the old ones, that are being analysed in the next tables.

ROUTE 8 PROBLEMS

	Number	Why	Solution
Collisions in junction	3	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with static object	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Collisions with bikers	2	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with stationary vehicles	6	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	1	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Outside route percentage	0.03	When the car is braking, it doesn't swerve so it goes straight in the direction had at the start of the braking	Implement a different behaviour when the vehicle has to brake, in a way that it continues to swerve

DRIVING SCORE: 0.011729

ROUTE COMPLETION: 76.32

INFRACTION PENALTY: 0.000154

ROUTE 9 PROBLEMS

	Number	Why	Solution
Collisions during parking exit	1	We don't control, going out from a parking, if there are vehicles passing on the road	Check the vehicles that are passing and stop if there isn't the necessary space to exit
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with moving vehicles	1	Some curves are done too wide, so we collide with the vehicles in the other lane	Implement a good controller to avoid wide curves
Collisions with pedestrians	1	The pedestrian on the edge of the road is not recognized	Recognize the pedestrians also when they are on the edge of the road and avoid them

DRIVING SCORE: 4.188618

ROUTE COMPLETION: 79.15

INFRACTION PENALTY: 0.05292

ROUTE 10 PROBLEMS

	Number	Why	Solution
Collisions during parking exit	1	We don't control, going out from a parking, if there are vehicles passing on the road	Check the vehicles that are passing and stop if there isn't the necessary space to exit
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with moving vehicles	1	This collision is done because we had a previous one in the junction	Manage junctions
Collisions with pedestrians	1	The pedestrian on the edge of the road is not recognized	Recognize the pedestrians also when they are on the edge of the road and avoid them
Outside route percentage	0.6%	The route invasion is done because we had a previous collision in junction	Manage junctions

DRIVING SCORE: 3.654926

ROUTE COMPLETION: 34.41

INFRACTION PENALTY: 0.106217

ROUTE 11 PROBLEMS

	Number	Why	Solution
Collisions during parking exit	1	We don't control, going out from a parking, if there are vehicles passing on the road	Check the vehicles that are passing and stop if there isn't the necessary space to exit
Collisions with moving vehicles	1	Some curves are done too wide, so we collide with the vehicles in the other lane	Implement a good controller to avoid wide curves
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Outside route percentage	0.6%	The route invasion is done because the controller does a too wide curve	Implement a good controller to avoid wide curves

DRIVING SCORE: 15.269967

ROUTE COMPLETION: 32.1

INFRACTION PENALTY: 0.4757

ROUTE 12 PROBLEMS

	Number	Why	Solution
Collisions during parking exit	1	We don't control, going out from a parking, if there are vehicles passing on the road	Check the vehicles that are passing and stop if there isn't the necessary space to exit
Collisions with stationary vehicles	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions with static object	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy

DRIVING SCORE: 6.430928

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.064309

Routes extra analysis summary

After analysing the extra routes, we have seen that the first thing to do, as seen previously, is to implement a well-tuned controller; it can give the possibility to avoid a lot of collision made into these routes and the basic ones. In the specific, we have seen a lot of problems already seen in the basic routes; the only new problems are the exit from a parking and the recognition of a pedestrian, also on the edge of the road.

Baseline 2.0

In this section, we present the results obtained by executing the baseline with the inclusion of well-tuned longitudinal and lateral controllers. Our goal was to evaluate the performance of the finalized controllers, including both the optimized Stanley controller and the PID controller for lateral control. The outcome of this evaluation will guide us in determining which of these controllers is the most effective for our autonomous driving system. It is important to note that the controllers used in this evaluation are the final versions that have undergone extensive tuning and refinement. The evaluation of the baseline with the tuned controllers provides valuable insights into the system's capabilities and highlights any areas for improvement. By comparing the performance of the Stanley lateral controller and the PID lateral controller, we aim to determine which controller offers superior trajectory tracking, stability, and responsiveness in various driving scenarios.

Longitudinal control

In the field of autonomous vehicle driving, achieving precise control over both longitudinal and lateral motion is crucial for ensuring safe and efficient operation. Longitudinal control focuses on regulating the speed and acceleration of the vehicle, while lateral control involves maintaining the desired trajectory and lateral position. Longitudinal control is responsible for regulating the speed and acceleration of the vehicle, allowing it to follow a desired velocity profile. This control aspect is essential for maintaining a safe distance from preceding vehicles, ensuring smooth transitions between speed levels, and achieving comfortable ride experiences. For the longitudinal control, we have chosen a PID controller that is a widely used control algorithm for both longitudinal and lateral control in autonomous driving. It provides a feedback mechanism to continuously adjust control inputs based on the error between the desired state and the measured state. The PID controller consists of three main components: Kp term generates a control input proportional to the current error, which is the difference between the desired state and the measured state. The proportional gain determines the strength of this response. Ki term integrates the error over time and is responsible for eliminating steady-state errors. It provides long-term correction by accumulating the error and generating a control input based on the integral gain. Kd term anticipates the future error trend by considering the rate of change of the error. It provides damping and stability to the system by generating a control input proportional to the derivative gain. The sets of values chosen for PID longitudinal controller are: Kp = 0.37, KI = 0.7 and KD = 0.024, with a dt of 0.05.

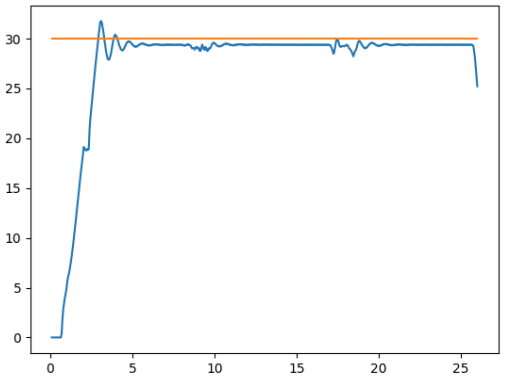


Figure 1. Speed.txt

Lateral control

Lateral control involves maintaining the desired trajectory and lateral position of the vehicle. It allows the autonomous vehicle to navigate through curves, follow lane markings, and perform manoeuvres such as lane changes. Effective lateral control is essential for ensuring accurate vehicle positioning, enhancing stability, and enabling safe and efficient navigation in complex traffic scenarios. For this type of controller, we have evaluated both a Stanley controller and a PID one. Regarding the PID, in the process of its implementation, we identified the need for different parameter sets based on the vehicle's speed. We recognized that the dynamics of the vehicle varied significantly at different velocities, requiring separate parameter tuning to ensure stability in both low and high-speed scenarios. Therefore, we established two sets of parameters for the lateral PID controller. One set was designed for speeds below 45 km/h, while the

other set was tailored for speeds above 45 km/h. The set for the lower speed is: Kp = 0.45, KI = 0.4 and KD = 0.001, with a dt of 0.06; the set for higher speed is: Kp = 0.45, KI = 0.4 and KD = 0.001, with a dt of 0.08. By utilizing distinct parameter sets, we could optimize the control response for different speed ranges, maintaining stability and accuracy in various driving conditions. This need was not necessary for the Stanley on which we found a good couple of curvature gain (ks) and the velocity gain (kv) that had a good behaviour also with the different velocities encountered in the routes. Stanley controller is a lateral control algorithm that is widely used for autonomous driving applications, especially in the context of path tracking and following. It combines a kinematic vehicle model with a control law that aims to minimize the lateral error between the vehicle's position and a reference trajectory. The Stanley controller achieves this by adjusting the vehicle's heading angle and steering control input. The sets of values chosen for Stanley controller are KV = 3.9 and KS = 1.0 with a dt of 0.02. The only variable parameter for the Stanley controller is the lookahead distance; it is an essential parameter that influences the path tracking behaviours of the autonomous vehicle. The lookahead distance determines how far ahead the controller anticipates the desired path and adjusts the steering angle accordingly. A larger lookahead distance allows the vehicle to anticipate and react to upcoming curves or changes in the desired trajectory earlier. This can help the vehicle achieve smoother and more precise path tracking, especially in scenarios with high-speed manoeuvres or sharp turns. On the other hand, a shorter lookahead distance may result in a more reactive control behaviours, with the vehicle adjusting its trajectory closer to the actual curve. During the tuning process of the Stanley lateral controller, we carefully considered the lookahead distance to optimize the path tracking performance. We conducted multiple experiments with different lookahead distances and evaluated the resulting trajectory accuracy, smoothness, and stability. By systematically adjusting the lookahead distance, we aimed to find the optimal value that achieved a balance between anticipatory control and responsiveness. The values of lookahead distance chosen are 0 for velocities under 5 km/h, 1 for velocities between 5 and 20 km/h and 1.5 for greater velocities.

Routes analysis

BASELINE – PID FINAL CONTROLLER – BASIC ROUTES

As said before, we quickly realized that the controller was not performing as expected and was causing several issues. Therefore, we experimented with different tuning values to improve the controller's performance. We continued to fine-tune both the longitudinal and lateral controllers until we reached an optimal configuration. In this section we report the results we have had using the PID as lateral controller and longitudinal one. In addition to fine-tuning the PID controller for both longitudinal and lateral control, we also anticipated the need for different controller parameters at different speeds, as said before.

	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4
Route Completion	100%	100%	100%	100%	100%
Outside Route Lanes	0%	0.13%	0%	0%	0%
Collisions	21	16	0%	1	5
Running Red Light	0	0	0	0	0
Running Stop	1	2	0	0	4

Min Speed	101.35%	52.85%	113.47%	114.2%	143.44%
In Route	✓	✓	✓	✓	✓
Agent Blocked	✓	✓	✓	✓	✓
Scenario Timeout	2	4	0	0	1
Timeout	✓	✓	✓	✓	✓

GLOBAL DRIVING SCORE: 32.31

GLOBAL ROUTE COMPLETION: 100.00

GLOBAL INFRACTION PENALTY: 0.32

In this first table we have analysed the problems of the mandatory routes using the baseline and our well-tuned controller. At this point we have only implemented our controllers (PID for lateral and longitudinal) and the set of values for high speed. As we can see, the routes completion is increased but the collisions in the different routes have changes and, in some cases, also increased, because we reach new part of the routes that could not be reached with the basic controller. Next we analyse the problems of the single routes to see if there are new tasks to add in the summary obtained at the end of the first simulation.

ROUTE 0

	Number	Why	Solution
Collisions with bikers	7	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with stationary vehicles	7	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	6	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with static object	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal

DRIVING SCORE: 0.000753

ROUTE COMPLETION: 100

INFRACTION PENALTY: 8e-6

ROUTE 1

	Number	Why	Solution
Collisions with bikers	6	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with stationary vehicles	4	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	4	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with static object	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Collisions with pedestrians	1	The pedestrian on the edge of the road is not recognized	Recognize the pedestrians also when they are on the edge of the road and avoid them
Stop signal disrespected	2	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Outside route percentage	0.13%	When the car is braking, it doesn't swerve so it goes straight in the direction had at the start of the braking	Implement a different behaviour when the vehicle has to brake, in a way that it continues to swerve

DRIVING SCORE: 0.002124

ROUTE COMPLETION: 100

INFRACTION PENALTY: 2.1e-5

ROUTE 2

	Number	Why	Solution
Junction		The junctions are not managed well, the vehicle almost collided during their execution	Recognize junctions and have a cautious behaviour to avoid collisions

DRIVING SCORE: 100

ROUTE COMPLETION: 100

INFRACTION PENALTY: 1.0

ROUTE 3

	Number	Why	Solution
Collisions with stationary vehicle	1	Our vehicle recognized the car in front and started stopping, but at the end continued to go with slow speed colliding with it	Implement a way to copy the behaviour of the car in front of us to have a cautious and secure drive

DRIVING SCORE: 60.0

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.6

ROUTE 4

	Number	Why	Solution
Collisions with stationary vehicle	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	2	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with pedestrians	1	The pedestrian has been recognized too late	Recognize the pedestrians also with more anticipation to have a cautious behaviour
Collisions with bikers	1	Bikers not recognized on the road	Recognize them and implement a surpass policy
Stop signal disrespected	4	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal

DRIVING SCORE: 1.507506

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.015075

As we can see from the tables, with the implementation of a well-tuned controller we have improved performance in some routes, improving then the global score. In particular we can notice that in route 0 we have avoided a collision thanks to the fact that the curves are done very well now without having wide trajectory; in fact a collision with the basic controller was caused by a wide curve of the car. This is the only improvement obtained into the first route. Regarding second route (route 1) the route completion now arrives to 100% instead of the 12.14% obtained with the old controller; because of this, the collisions are increased and we have also 2 stops signal disrespected thanks to the fact that new part of this route have been discovered. The route 2 has improved but this is only casually happened because the controller gives better stability and follows better the target speed, so we avoided the collision in the junction and we didn't

have a stop signal disrespected, but the junction and stop policy still have to be implemented in a good way. The route 3 already gave good results with the basic controller, so we didn't achieve improvements on that one. Route 4 reached the 100% completion, the collisions are increased because new part of the route were explored, same for the stop signals disrespected. In general, we reached what expected without discovering new type of problems different from the ones revealed from the first analysis done with a basic controller. Now we will analyse the results on the extra routes with the well-tuned controller.

BASELINE – PID CONTROLLER – ROUTE EXTRA

	ROUTE 8	ROUTE 9	ROUTE 10	ROUTE 11	ROUTE 12
Route Completion	76.43%	100%	100%	100%	100%
Outside Route Lanes	0%	0%	0.39%	0%	0%
Collisions	14	3	6	3	3
Running Red Light	0	0	0	0	1
Running Stop	1	0	0	2	0
Min Speed	77.07%	146.25%	154.75%	103.94%	53.76%
In Route	✓	✓	✓	✓	✓
Agent Blocked	✓	✓	✓	✓	✓
Scenario Timeout	6	1	1	1	2
Timeout	✗	✓	✓	✓	✓

GLOBAL DRIVING SCORE: 4.71

GLOBAL ROUTE COMPLETION: 95.29

GLOBAL INFRACTION PENALTY: 0.05

ROUTE 8

	Number	Why	Solution
Collisions in junction	2	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with stationary vehicle	3	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	4	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with bikers	1	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with static objects	4	Static objects, as the traffic warning and the construction	Recognize static objects that can't be trampled and

		cone in this route, are not recognized	implement a stop and surpass policy
Outside route percentage	0.07%	The incident in the junction causes an invasion of the other lane	Resolve the collision in the junction
Stop signal disrespected	2	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal

DRIVING SCORE: 0.004899

ROUTE COMPLETION: 76.43

INFRACTION PENALTY: 6.4e-05

ROUTE 9

	Number	Why	Solution
Collisions with moving vehicles	1	During the exit from the parking, we don't control if there are vehicles	Recognize vehicle that are passing near in the road
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with pedestrians	1	The pedestrian on the edge of the road is not recognized	Recognize the pedestrians also when they are on the edge of the road and avoid them

DRIVING SCORE: 15.12

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.1512

ROUTE 10

	Number	Why	Solution
Collisions with moving vehicles	1	During the exit from the parking, we don't control if there are vehicles	Recognize vehicle that are passing near in the road
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with pedestrians	3	The pedestrian on the edge of the road is not recognized	Recognize the pedestrians also when they are on the edge of the road and avoid them
Collisions with stationary vehicle	1	There are some vehicles on the road that are collided and stopped with different orientation	Recognize vehicles also if they invade partially the road

Outside Route percentage	0.39%	The incident in the junction causes an invasion of the other lane	Resolve the collision in the junction

DRIVING SCORE: 1.836533

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.018365

ROUTE 11

	Number	Why	Solution
Collisions with moving vehicles	2	During the exit from the parking, we don't control if there are vehicles; the other one is caused by a stop signal disrespected	Recognize vehicle that are passing near in the road; recognize the stop signals
Collisions with stationary vehicles	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Stop signal disrespected	2	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Red signal disrespected	1	The vehicle stopped after passing the red light	Implement a correct policy for traffic lights

DRIVING SCORE: 3.013971

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.03014

ROUTE 12

	Number	Why	Solution
Collisions with moving vehicles	1	During the exit from the parking, we don't control if there are vehicles	Recognize vehicle that are passing near in the road
Collisions with stationary vehicles	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions with static object	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Red signal disrespected	1	The vehicle stopped after passing the red light	Implement a correct policy for traffic lights

DRIVING SCORE: 3.58378

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.035838

With the implementation of a well-tuned controller, route extra also have better performance. In the route 8 we have about the same scores as the ones with the basic controller because the collisions were not caused by the controller. The route 9 with the final controller reaches 100% completion and doesn't block colliding because of the wide curves. Also in the route 10 the completion now arrives at 100%; the collisions increased because new part of the route were explored. Route 11 reached the 100% completion and doesn't invade outside lanes. At the end the route 12 doesn't have great improvements respect to the basic controller simulation.

BASELINE – STANLEY FINAL CONTROLLER – BASIC ROUTES

We experimented with different tuning values to improve the Stanley controller's performance. In this section we report the results we have had using the Stanley as lateral controller and PID as longitudinal one. In addition to fine-tuning the Stanley controller for lateral and the PID for longitudinal control, we also anticipated the need for different control system at different speeds; for the Stanley lateral controller we decided to use different values only for lookahead distance, as explained before in the paragraph on Lateral control.

	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4
Route Completion	100%	100%	100%	100%	100%
Outside Route Lanes	0%	0%	0%	0%	0.68%
Collisions	26	8	0%	5	4
Running Red Light	0	0	0	0	0
Running Stop	1	0	1	0	4
Min Speed	109.12%	39.56%	110.68%	115.48%	130.93%
In Route	✓	✓	✓	✓	✓
Agent Blocked	✓	✓	✓	✓	✓
Scenario Timeout	6	3	0	0	1
Timeout	✓	✗	✓	✓	✓

GLOBAL DRIVING SCORE: 19.444113

GLOBAL ROUTE COMPLETION: 90.704

GLOBAL INFRACTION PENALTY: 0.194935

In this first table we have analysed the problems of the mandatory routes using the baseline and our well-tuned controller with Stanley as lateral and PID as longitudinal. At this point we have only implemented our controllers and the different lookahead distances for different speeds. As we can see, the routes completion is increased but the collisions in the different routes have changes and, in some cases, also increased, because we reach new part of the routes that could not be reached with the basic controller. Next we analyse the problems of the single routes to see if there are new tasks to add in the summary obtained at the end of the first simulation.

ROUTE 0

	Number	Why	Solution
Collisions with bikers	8	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with stationary vehicles	5	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	4	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with static object	2	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Collisions with moving vehicles	6	Some curves are done too wide, so we collide with the vehicles on the other lane; another case of collision is that during a braking our vehicle doesn't swerve and collide with vehicles	Correct the controller to avoid wide curves and implement a different behaviour when the vehicle has to brake, in a way that it continues to swerve
Collisions in junctions	1	In the junction we pass when it's not possible	Implement a good logic for the junctions

DRIVING SCORE: 0.0

ROUTE COMPLETION: 100

INFRACTION PENALTY: 1.4e-05

ROUTE 1

	Number	Why	Solution
Collisions with bikers	3	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with stationary vehicles	2	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	2	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with static object	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy

DRIVING SCORE: 0.284626

ROUTE COMPLETION: 53.52

INFRACTION PENALTY: 0.005318

ROUTE 2

	Number	Why	Solution
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal

DRIVING SCORE: 80.0

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.8

ROUTE 3

	Number	Why	Solution
Collisions with stationary vehicle	5	Our vehicle recognized the car in front and started stopping, but at the end continued to go with slow speed colliding with it	Implement a way to copy the behaviour of the car in front of us to have a cautious and secure drive

DRIVING SCORE: 12.96

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.1296

ROUTE 4

	Number	Why	Solution
Collisions with stationary vehicle	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	2	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Stop signal disrespected	4	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Collisions in junctions	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Outside Route percentage	0.68%	The incident in the junction causes an invasion of the other lane	Resolve the collision in the junction

DRIVING SCORE: 3.975925

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.039759

As we can see from the tables, with the implementation of a well-tuned controller we have improved performance in some routes, improving then the global score. In general, we reached what expected without discovering new type of problems different from the ones revealed from the first analysis done with a basic controller. Now we will analyse the results on the extra routes with the well-tuned controller using the Stanley.

BASELINE – STANLEY FINAL CONTROLLER – EXTRA ROUTES

	ROUTE 8	ROUTE 9	ROUTE 10	ROUTE 11	ROUTE 12
Route Completion	81.34%	100%	96.22%	100%	100%
Outside Route Lanes	0.13%	0%	1.76%	0.41%	0%
Collisions	14	5	4	6	5
Running Red Light	0	0	0	0	0
Running Stop	1	0	0	2	0
Min Speed	77.05%	154.9%	132.5%	92.81%	68.22%
In Route	✓	✓	✓	✓	✓
Agent Blocked	✓	✓	×	✓	✓
Scenario Timeout	6	1	2	1	2
Timeout	×	✓	✓	✓	✓

GLOBAL DRIVING SCORE: 2.690646

GLOBAL ROUTE COMPLETION: 95.512

GLOBAL INFRACTION PENALTY: 0.027224

ROUTE 8

	Number	Why	Solution
Collisions in junction	2	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with stationary vehicle	6	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions when blocked	3	When we collide with stopped vehicles, we stay blocked, and the other vehicles collide with us	Surpass the vehicles to avoid blocked conditions
Collisions with bikers	2	Bikers not recognized on the road	Recognize them and implement a surpass policy
Collisions with static objects	1	Static objects, as the traffic warning and the construction cone in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Outside route percentage	0.13	The incident in the junction causes an invasion of the other lane	Resolve the collision in the junction
Stop signal disrespected	1	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal

DRIVING SCORE: 0.005655

ROUTE COMPLETION: 81.34

INFRACTION PENALTY: 7e-05

ROUTE 9

	Number	Why	Solution
Collisions with moving vehicles	3	During the exit from the parking, we don't control if there are vehicles	Recognize vehicle that are passing near in the road
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with pedestrians	1	The pedestrian on the edge of the road is not recognized	Recognize the pedestrians also when they are on the edge of the road and avoid them

DRIVING SCORE: 5.4432

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.054432

ROUTE 10

	Number	Why	Solution
Collisions with moving vehicles	2	During the exit from the parking, we don't control if there are vehicles	Recognize vehicle that are passing near in the road
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Collisions with pedestrians	1	The pedestrian on the edge of the road is not recognized	Recognize the pedestrians also when they are on the edge of the road and avoid them
Outside Route percentage	1.76%	The incident in the junction causes an invasion of the other lane	Resolve the collision in the junction

DRIVING SCORE: 4.012129

ROUTE COMPLETION: 96.22

INFRACTION PENALTY: 0.041697

ROUTE 11

	Number	Why	Solution
Collisions with moving vehicles	4	During the exit from the parking, we don't control if there are vehicles	Recognize vehicle that are passing near in the road
Collisions with stationary vehicles	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Stop signal disrespected	2	The stop policy is not implemented	Implement a stop policy for stop and wait when there is this signal
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions
Outside Route percentage	0.41%	The brake without swerve causes an invasion of the other lane	Implement the logic to swerve during a brake

DRIVING SCORE: 1.507758

ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.015078

ROUTE 12

	Number	Why	Solution
Collisions with moving vehicles	2	During the exit from the parking, we don't control if there are vehicles	Recognize vehicle that are passing near in the road
Collisions with stationary vehicles	1	Vehicles stopped at the edge of the road are not recognized	Recognize those vehicles and implement a surpass policy
Collisions with static object	1	Static objects, as the traffic warning in this route, are not recognized	Recognize static objects that can't be trampled and implement a stop and surpass policy
Collisions in junction	1	In the junction we pass when it's not possible	Implement a good logic for the junctions

DRIVING SCORE: 2.484489

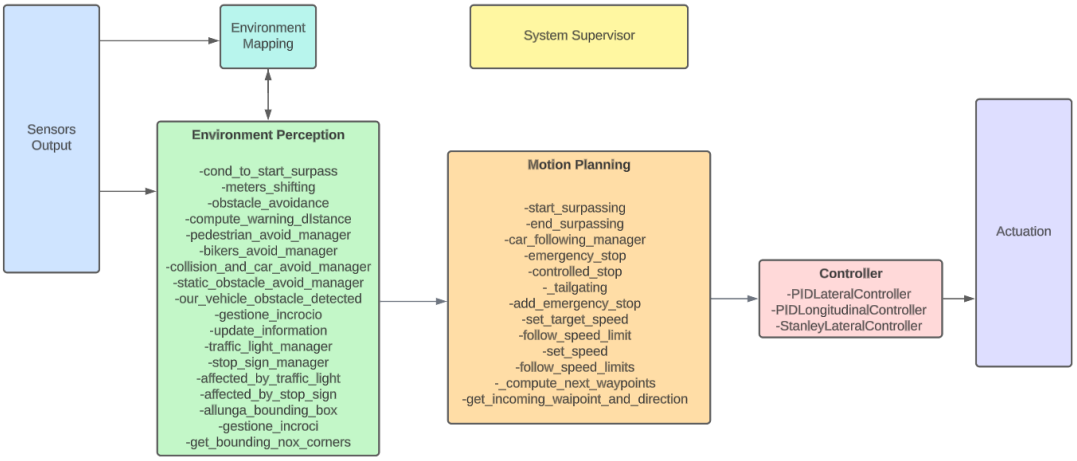
ROUTE COMPLETION: 100

INFRACTION PENALTY: 0.024845

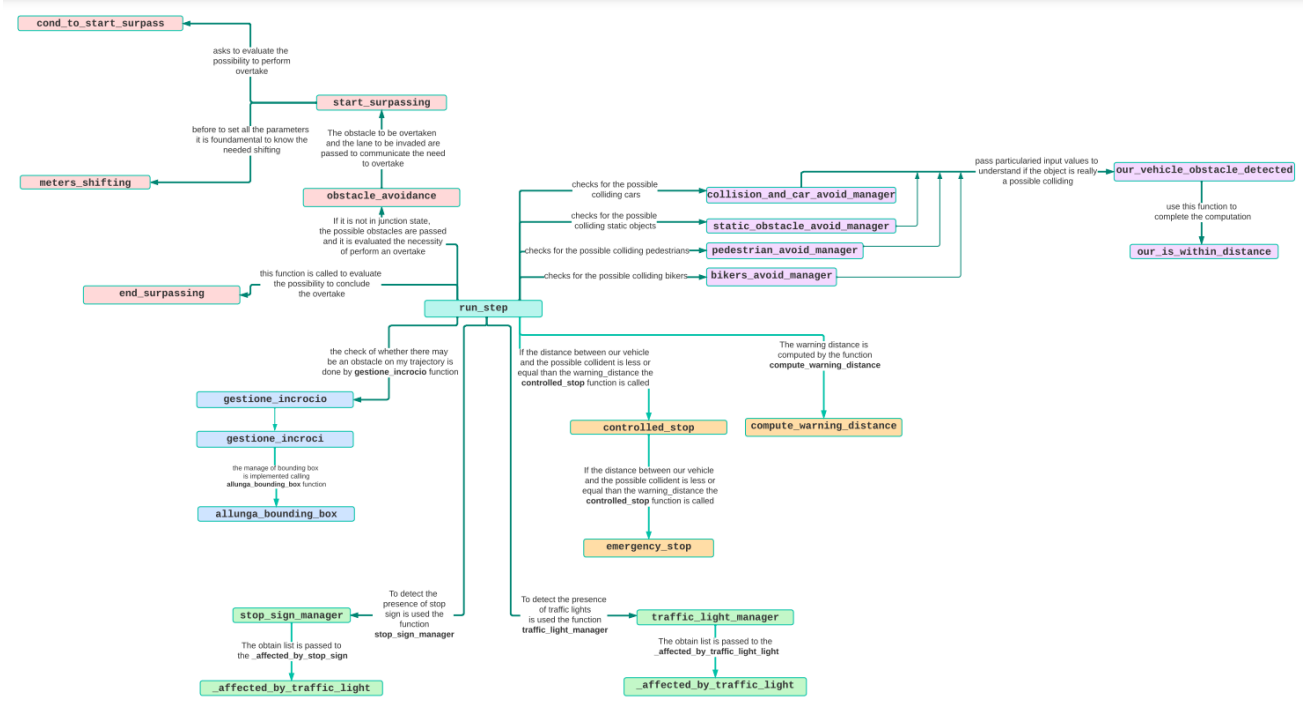
Final Project

In this section of the report, we will discuss the final version of the project that, starting from the previously described baseline, has undergone the modifications and improvements we had planned. We will present the various modules that we have developed or improved based on existing implementations. For each module, we will provide a detailed architecture visualization and an accurate description.a

Below is a general overview of the overall architecture, illustrating how our implemented functions fit into the broader architecture of an autonomous vehicle driving system. Specifically, our implemented functions are primarily located in the environment perception and motion planning sections. In the environment perception section, we have incorporated all the functions used to obtain information about other vehicles or any other objects present in the scene, as well as information about the road itself. In the motion planning section, we have included all the functions necessary for managing the definition of the path for our vehicle, taking into account any deviations dictated by obstacles of any kind.

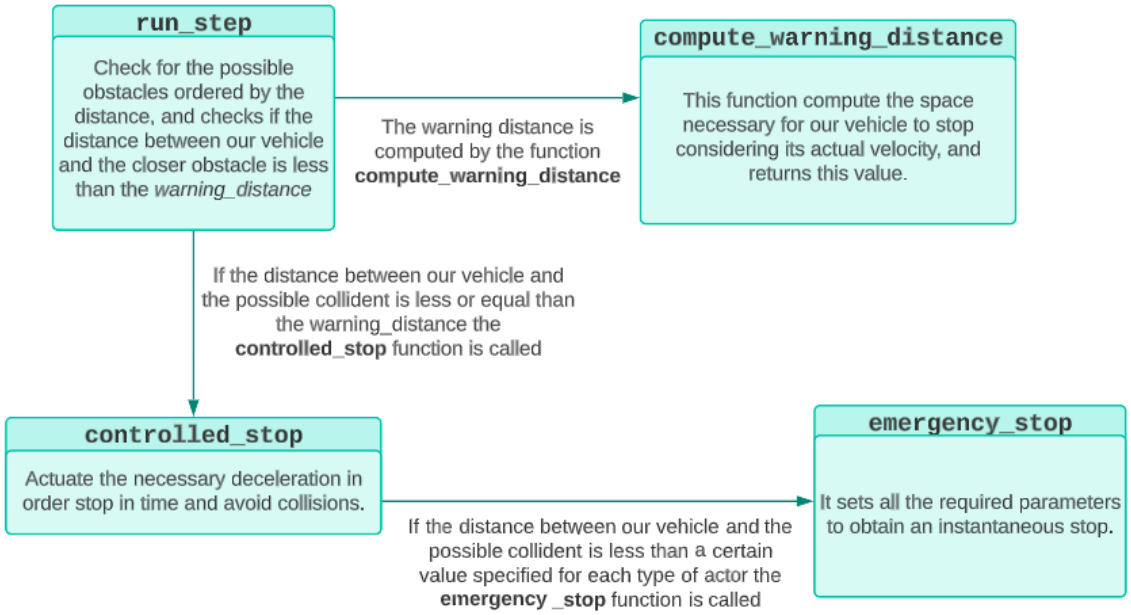


Below, a high-level diagram will be presented, representing the interconnections between the various modules, as well as the various supporting functions used.



Brake management

This section of the report contains a description of all the modules involved in the braking control of our vehicle in case of needs arising, for example, from the risk of collision with other objects. Specifically, compared to what was already present in the baseline, where the command to stop the vehicle was given in case of potential collisions, efforts have been made to make improvements that would avoid abrupt and overly early braking. The modules involved for this purpose are ***compute_warning_distance***, ***controlled_stop*** and ***emergency_stop***. So, the objective of this modules is to control the deceleration and stopping behaviour of our vehicle based on the distance from the potential obstacle, ensuring a safe and controlled stop, considering the current speed, the distance, and applying appropriate speed reduction to avoid collisions. The relationship that interconnects the various modules will be explained in the following and is clear from the below schema.

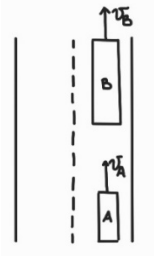


With respect to the schema reported above the only exception is represented by the traffic light that directly uses the function emergency stop without actuating the controlled stop. This implementation has been made in this way because the red light of the traffic light is something not predictable unless history of signal sequence is kept, so when the light is red it is necessary to perform a hard brake.

Compute warning distance

This function is used to calculate the distance of the potential obstacle relative to our vehicle, from which it is necessary to start decelerating in order to brake in time without colliding with the possible colliding object. In particular, it is obtained the possible colliding vehicle's speed in m/s, then it is computed the relative velocity of our vehicle respect to that of the possible colliding, where the relative velocity is that represented in the following schema.

Relative speed



$$\boldsymbol{v_{Arelativa} = v_A - v_B}$$

At this point, starting from the formulas of uniformly accelerated motion applied to our vehicle for which:

$$v_{relative\_final}{}^2 = v_{relative\_initial}{}^2 + 2a(s - s_0)$$

and considering that the final velocity that we expected to reach is equal to zero, because we want to perform a deceleration, and that also s_0 is equal to zero, it is obtained the distance starting from which it is necessary to decrease the speed:

$$distance_{to_{start_{stop}}} = \frac{v_{relative_{initial}}{}^2}{2a}$$

considering as acceleration the maximum allowable deceleration for a general real vehicle, which is equal to 3.86m/s^2.

The value $distance_{to_{start_{stop}}}$ is the return value of this function.

It is important to highlight that the calculation performed within this function can easily be replaced by a much simpler calculation, explained below:

$$distance_{to_{start_{stop}}} = (v_{relative}/10)^2$$

where $v_{relative}$ in this case is computed in Km/h while $distance_{to_{start_{stop}}}$ is computed in meters, so there is no concordance of measurements units.

So according to this empirical formulation, the distance at which starting to decelerate is equal to the actual velocity divided by 10 and then raised to the power of two.

Although this empirical formulation is much simpler to implement and completely equivalent to the implementation in the ***compute_warning_distance*** function, the function has still been implemented in a way that provides physical support for the obtained result.

In the following it is provided the demonstration that the two formulations are equivalent:

$$distance_{to_{start_{stop}}} = \left(\frac{v_{relative_{\frac{km}{h}}}}{10}\right)^2 = \frac{(v_{relative_{\frac{km}{h}}})^2}{100} = \frac{(v_{relative_{\frac{m}{s}}} * 3.6)^2}{100} = \frac{(v_{relative_{\frac{m}{s}}})^2 * 3.6^2}{100} =$$

$$= \frac{(v_{relative_{\frac{m}{s}}})^2 * 12.96}{100} = \frac{(v_{relative_{\frac{m}{s}}})^2}{7.716} \sim \frac{(v_{relative_{\frac{m}{s}}})^2}{2 * a}$$

considering that $a = 3.86$ and $2 * a = 7.72$.

So, because of this reason these two ways are used interchangeably in the code.

Controlled Stop

The ***controlled_stop*** function is responsible for managing the controlled stop behavior of a vehicle. This function is designed to be, also in this case, as closely adherent to reality as possible, achieving smooth and controlled braking, just as it should be in normal driving conditions, with the exception of emergency situations.

It takes in input as parameters the obstacle to avoid and due to which it is necessary to slow down, the actual distance from this and also the minimum allowed distance between our vehicle and the obstacle, below which it is necessary to give a stop command almost instantly.

In this function it is first checked if the distance between our vehicle and the obstacle is less or not then the minimum allowed distance, in positive case an emergency stop is triggered by calling the ***emergency_stop*** function, ensuring that the vehicle halts immediately to avoid collision. If the actual distance is greater than the minimum allowed distance it means that there is the necessity to slow down but not to abruptly brake. At this point, we need to find the right decrease in speed, which is the appropriate deceleration, in order to ensure that our vehicle will be able to brake in time without colliding with the obstacle.

For our vehicle, also in this case, a uniformly accelerated motion has been assumed, considering as the acceleration the maximum allowed deceleration for a vehicle, which is generally considered equal to 3.86m/s^2. At this point, the current speed of our vehicle (*norm_velocity*) has been evaluated as the norm of the vector representing the velocity of our vehicle.

Then the target speed value has been set equal to the actual speed, from which a certain quantity, referred to as '***speed_to_sub***', is subtracted. This quantity corresponds, in its most general formulation, to:

$$speed_{to_{sub}} = \frac{(3.86 * norm_{velocity})}{distance}$$

This quantity has been formulated in this way considering the fact that 3.86 m/s^2 is the maximum possible deceleration of a normal vehicle, but it was not sufficient to only reduce by 3.86 m/s^2 because this reduction might not always be achieved, resulting in an inaccurate estimation of the braking time. Considering that the action of the controller becomes more aggressive as the error relative to the desired parameter increases, it was ensured that this quantity is proportional to the maximum allowed deceleration but additionally, it is directly proportional to the current velocity and inversely proportional to the distance from the obstacle. This means that it increases as the distance decreases while keeping the velocity constant, and vice versa, it decreases as the distance decreases while keeping the velocity constant. A similar concept can be applied when keeping the distance fixed and varying the velocity.

The above statement represents a general formulation, but then there were several factors that needed to be taken into account. Specifically, for speeds above 50 km/h, an additional multiplicative factor is added to the $speed_{to_{sub}}$ quantity, as it is necessary to make a more pronounced deceleration in those situations. Conversely, if the speed of our vehicle dropped below 2 km/h, it was ensured that this speed is maintained until a definitive stop is required, in order to avoid frantic deceleration and acceleration activities. Once the target speed has been determined through all the aforementioned methods, it is set within the local

planner and its ***run_step*** method is executed. The resulting control information is returned as the output of this function.

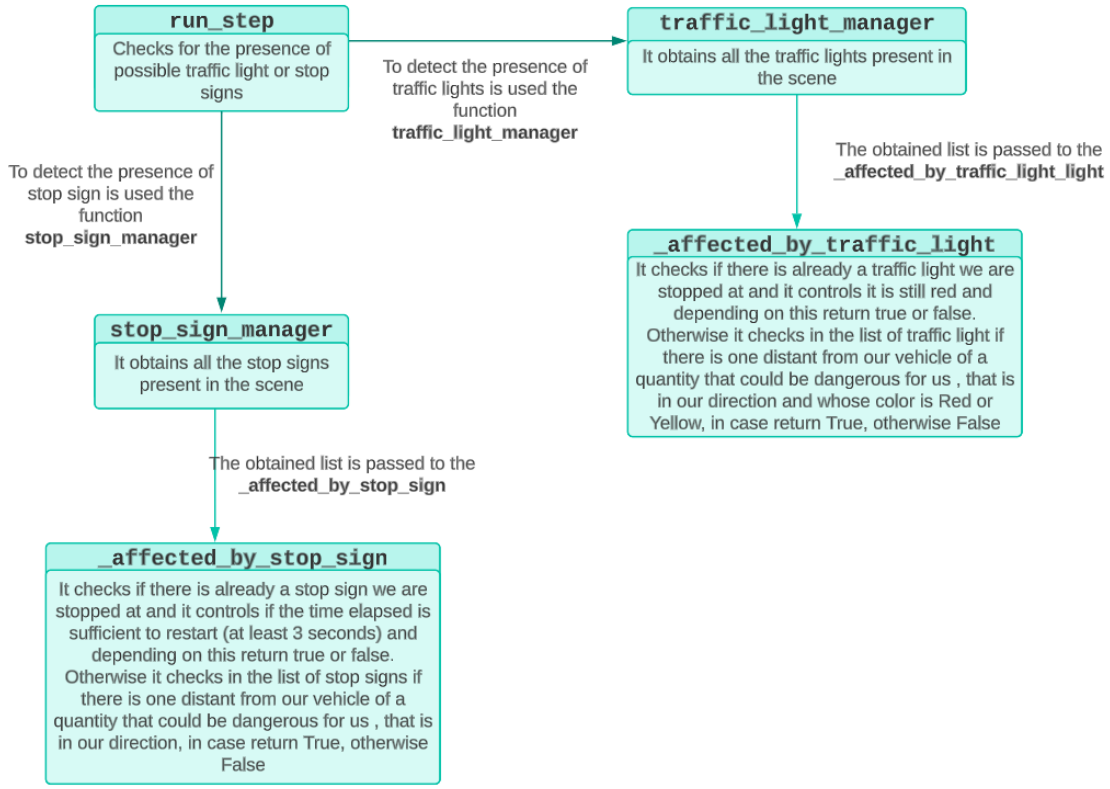
In the following it is possible to see a video representing how our vehicle performs a controlled stop when the bikers are detected, before starting the overtake. In particular it is possible to see that performing a cautious deceleration helps our vehicle to not completely stopped before to start overtaking, maintaining a minimum speed that helps it in the maneuver. this link.

Emergency stop

An important modification was made to the ***emergency_stop*** function with respect to that contained in the Baseline. In fact, we noticed that this function allowed setting the target speed to zero and the brake value to the maximum possible (0.5). However, the issue was that this function did not continue to apply lateral control, resulting in no steering input being sent when the vehicle was in emergency stop mode. As a result, the vehicle would abruptly brake but continue moving along its path. This behavior was not desirable, so we decided to add steering control even during emergency braking. This is particularly crucial in curves where it is essential to maintain the ability to follow the road while applying maximum braking force. To do this we add another function in the local planner in order to apply the lateral control also in these situations of ***emergency_stop***. By doing this, we ensure that the vehicle continues to follow its waypoints even when braking abruptly.

Here it is possible to see how work the brake management in emergency situations like that of hard brake of the vehicle in front of us this link.

Vertical Signal Management



Vertical signage management is a crucial aspect of autonomous driving systems, as it involves the detection, interpretation and response to various traffic signs and signals present on the road. These signs play a vital role in regulating traffic flow, ensuring safety and providing essential information to drivers. In our project, we have focused on two significant types of vertical signage: stop signs and traffic lights. Let's explore the importance of vertical signage management and delve into the functions we have implemented:

Traffic lights are essential elements of urban road infrastructure, controlling the flow of vehicles and pedestrians at intersections. Efficiently interpreting and responding to traffic lights is vital for an autonomous vehicle to navigate safely and follow traffic regulations. In our project, we have implemented functions to detect and interpret the state of traffic lights. Our system determines whether the traffic light is red, yellow, or green, enabling the vehicle to make appropriate decisions, such as stopping or proceeding.

Stop signs are fundamental traffic control devices used at intersections to manage the right-of-way and ensure safety. Recognizing and appropriately responding to stop signs is crucial for an autonomous vehicle to comply with traffic rules. In our implementation, we have developed a function to detect stop signs and triggers the appropriate response, such as deceleration and full stop when required.

Accurate and reliable vertical signage management is crucial for the safe and efficient operation of autonomous vehicles. By properly understanding and responding to stop signs and traffic lights, autonomous vehicles can effectively interact with other road users, ensure compliance with traffic regulations, and enhance overall road safety.

Traffic Light Manager

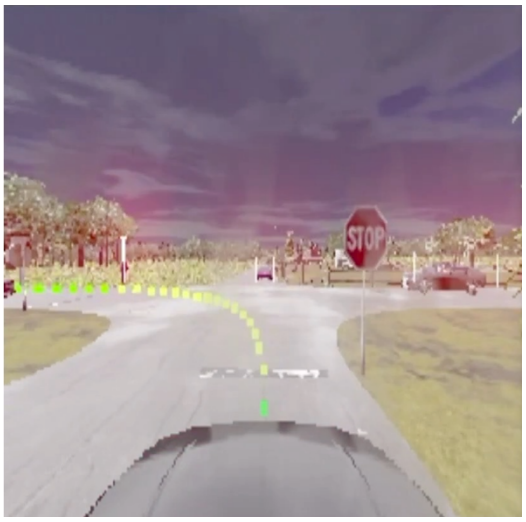
The ***_affected_by_traffic_light*** function is responsible for determining if the autonomous vehicle is affected by a traffic light. Its main purpose is to detect and handle traffic lights by checking their state and the proximity to the vehicle. The function then checks if the vehicle has encountered a traffic light in the previous simulation step. If so, it checks the state of that traffic light. If the state is neither red nor yellow, indicating that the vehicle could pass It, then the reference to the last traffic light is cleared. Otherwise, it returns that there is a red or yellow traffic light affecting the vehicle. If in the previous simulation step no traffic light was encountered, it retrieves the location and waypoint of the ego vehicle. It iterates through the traffic lights in the provided or obtained list and checks if each traffic light is relevant based on its proximity to the vehicle. It also verifies that the traffic light is on the same road as the ego vehicle. For each relevant traffic light, it checks the orientation to determine if it is facing the vehicle. If it's not facing the vehicle, the function continues to the next traffic light. If the traffic light is facing the vehicle and its state is either red or yellow, the function checks if the vehicle is within a certain distance from the traffic light. If it is within distance and within a certain angle of approach, it sets the last traffic light encountered to the current traffic light and returns that there is a red or yellow traffic light affecting the vehicle. If no relevant traffic light is found or none of the found traffic lights are red or yellow and near our vehicle, the function returns that there is no traffic light affecting the vehicle. An example is provided at the following link.

Figure 22. Stop at Traffic Light

Stop Sign

The main requirement related to stop signs stated the need to come to a stop for a duration equal to or greater than three simulation seconds. The function ***stop_sign_manager*** is responsible for managing the behaviours related to stop signs.

Figure 3 3. Stop behavior



Initially, information about the actors present in the simulation scene is obtained, filtering out those representing stop signs. Then, it is evaluated which of these signs affect the autonomous vehicle's driving, determining if the vehicle is impacted by a stop sign and the distance to it, through the function ***_affected_by_stop_sign***.

First, it checks if the option to ignore traffic signals is enabled. In such a case, a result indicating that the vehicle is not affected by any stop sign is returned. Next, it checks if a specific list of stop signs has been provided. If not, all the stop signs present in the simulation environment are obtained, also considering the maximum detection distance for stop signs.
Then it is checked whether the vehicle was already "engaged" in a stop, and if so, the elapsed time is verified (in this case, 3 (simulation) seconds as required). If enough time has passed, the last stop sign detection time is reset. Otherwise, a result is returned indicating that the vehicle is still affected by the previous stop sign. If no stop sign was detected in the previous step or enough time has passed, the search

for a new stop sign begins, which should have a different identifier from the one the vehicle stopped at this moment. The position and waypoint of the autonomous vehicle are obtained. Then, a loop is performed over all the stop signs to evaluate if the autonomous vehicle is affected by a specific stop sign.
The distance between the stop sign and the autonomous vehicle is calculated, and several criteria are applied to determine if the stop sign is relevant to the autonomous vehicle. These criteria include the maximum detection distance, the orientation of the stop sign relative to the autonomous vehicle and whether the stop sign is on the same road as the autonomous vehicle, in addition to having a different identifier than the previous one. If a relevant stop sign is detected, the distance between the autonomous vehicle and the stop sign is checked. If the distance is less than 2.5 meters, the detection time of the stop sign and the stop sign's ID are recorded. These pieces of information are necessary for the subsequent step to verify that the vehicle should handle that specific stop.
Finally, a result is returned indicating if the vehicle is affected by a stop sign, the stop sign's ID (if present), and the distance between the vehicle and the stop sign. An example is available at this link.

Detection of potential collisions

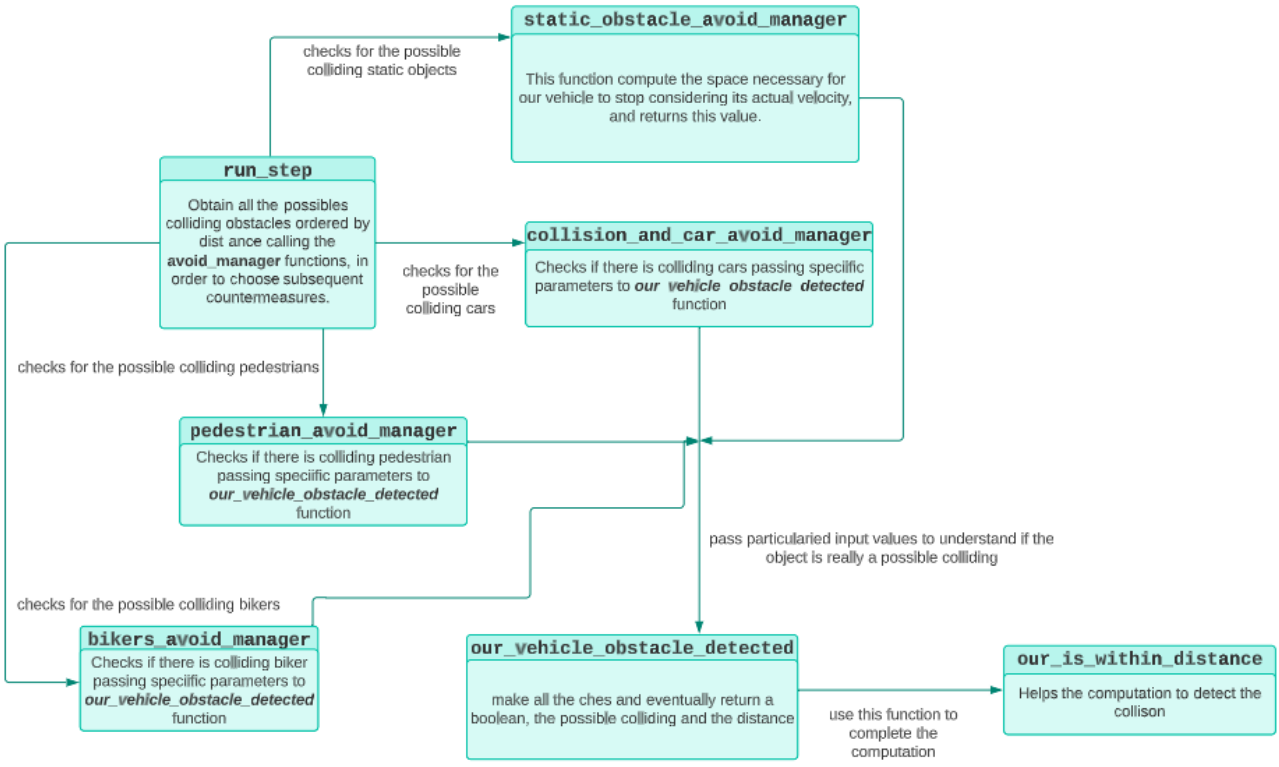
In the context of developing software for vehicles and advanced driving systems, road safety is an absolute priority. In this chapter of the report, we will explore the necessity and importance of integrating a dedicated module for collision detection into our software, in order to enable vehicles to operate safely and reliably reducing the risk of accidents and ensuring the protection of all road users.

This module is necessary not only for vehicle safety, allowing the vehicle to adjust in its behavior in relation to possible collisions, but can also contribute to optimizing traffic efficiency and flow. For example, the system can be used to detect the presence of slow-moving vehicles or objects on the road, enabling the vehicle to adopt an early driving strategy to avoid slowdowns or congestion.

It should be capable of identifying a wide range of objects, including vehicles, pedestrians, cyclists, and stationary objects such as road signs or barriers. It also should also be able to detect objects in various environmental conditions, such as low-light situations or adverse weather conditions.

In a real context this module relies on the use of advanced sensors, such as cameras, lidar and radar, to gather data on the presence and position of surrounding objects. This data is then processed using specifically developed algorithms to identify objects of interest and predict potential collisions, more and more artificial intelligence techniques are being used for this.

In this project for the development of this module, we relied on the information provided directly by the simulator, without going through the previously mentioned sensors or subsequent artificial intelligence algorithms. This allows us, assuming a perfect functioning of the simulator, to have practically zero error in the detection of potential collisions. This assumption is not totally correct because we need to correct and understand much information retrieved from the simulator. So, in the following we will explain all the modules involved for this purpose, their implementations as well as the relationships that exist among them, as it is possible to see in the following schema.



Vehicle Obstacle Detect

The '***_our_vehicle_obstacle_detected***' function is a fundamental function to ensure the correctness of this module. It is called by all the "*avoid_manager*" functions contained in the ***behavior_agent*** used by the ***run_step*** to manage the detection of all the types of actors that could be potential colliding obstacle, like pedestrians, cars, bikers, and static obstacles.

This function implements everything that is needed to understand if an object could collide with our vehicle in order to communicate this to the modules responsible for implementing possible countermeasures. So, it takes in input a list of objects of different types and ordered by distance, a maximum distance from the object within which to consider it a potential colliding object, two values of up and low angles and a lane offset. The purpose of these last three parameters will be explained in the following. This function returns a Boolean indicating if there where the possibility of a collision, with which object and the distance of this object from our vehicle. If no colliding object is found it returns False, None and -1.

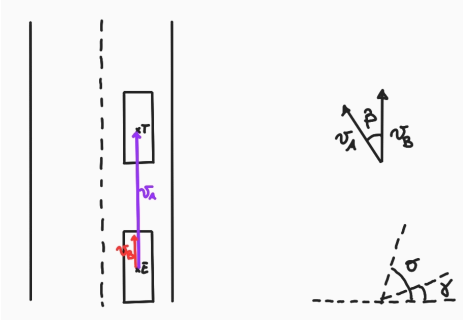
As it will be clear in the following all the input parameters are specified depending on which obstacle we are interested in detecting.

For the implementation of this function, we started from the implementation contained in the baseline in "**vehicle_obstacle_detected**" and we apply some modifications accordingly to all the issues identified. It is important to specify that our version of the function only manage possible collision in all situations except that for the junctions for which another management has been developed, as explained in the report.

First of all are retrieved all the information about our vehicle like its position, its forward vectors, the coordinates of the vertices of the bounding box and also the id of the lane in which they are. The same information is also retrieved for all the other objects ordered by distance and contained in the list. Then are computed information about the offset and the id of the lane on which our vehicle would move. The offset is a parameter depending on the fact that our vehicle has or not the intention of perform a change of the lane to the right or to the left. These two parameters are important in order to assess the possibility of collision, even with a predictive approach towards future actions, while avoiding unnecessary false alarms. If our vehicle and the potential colliding object are not yet in the same lane, the possibility of them being on the same lane in the subsequent waypoints is checked, both in the case of lane change and lane follow. If even this condition, with necessary consideration of offsets, is not satisfied, then we move forward in the controls.

Then is used the function "***our_is_within_distance***" at which are passed *target_rear_transform*, *ego_front_transform*, *target_rear_extent*, *ego_rear_extent*, max_distance and the low and upper angles. The last three parameters are the same received as input in the current described function, *target_rear_transform* and *ego_front_transform* are the location and the rotation of the possible colliding object and our vehicle, respectively, while *target_rear_extent* and *ego_rear_extent* are equal to the square root of sum of the squared half extents along x and y of the bounding boxes of our vehicle and of the potential colliding object. The output of the function "***our_is_within_distance***" contributes to the output returned. In particular it ensures that if has been found an object satisfying all the conditions, it would be considered as a possible colliding object only if the distance evaluated between our vehicle and the possible obstacle is less than the maximum distance passed as parameter and that the angle between our forward vector and the vector connecting our vehicle and the colliding is inside the up and low angle range. In the following it is reported a schema representing these mentioned vectors.

Upper and low angle function



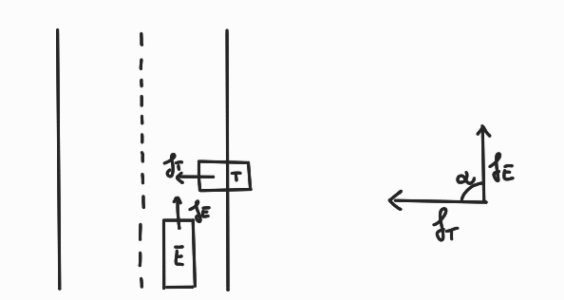
E represents our vehicle, T represents the target object, $\boldsymbol{v_B}$ is the forward vector of our vehicle and $\boldsymbol{v_A}$ is the vector connecting our vehicle with the possible colliding object.

*The angle described between these two **angles** is named beta in the figure and it is checked that beta is within gamma and sigma that represents the lower and the upper angle.*

Obviously, lots of other minor conditions are inserted to discretise between cases.

It is important to specify that a particular implementation, considering the angle described between the two vectors "*ego_forward_vector*" and "*target_forward vector*", whose significance will be clear in the following schema, has been made in order to manage every type of objects, except for the static ones, that could cross the road.

Crossing possible object



*E represents our vehicle, T represent**s** the crossing possible obstacle, $\boldsymbol{f_E}$ is the ego_forward_vector and $\boldsymbol{f_T}$ is the target_forward_vector.*

It is checked that the angle alpha between the two vectors is within 80 and 110 degrees.

If this separate management had not been done, which obviously was not present in the Baseline, since the previously stated conditions did not cover this scenario, there would have been a risk of not considering these objects as potential obstacles, and therefore the command to brake would not have been given to our vehicle.

At the following link it is possible to see an example of this implementation in action this link.

It is important to underline that this implementation consider both the case of lane follow and the case of lane change.

It is also important to clarify that all the following explained functions calls the above descripted one with different input parameters particularized depending on the type of object they refers to. In particular a management common to everyone is the value passed as lane offset that is -1 if the option is to make a change to the left lane, is equal to 1 if the option is to make a change to the right otherwise is equal to 0 if a lane change is not required.

Pedestrians avoid manager

This function allows to particularize the checks of the possible colliding objects on pedestrians, it obtains the list of pedestrians in a certain range passed as parameter (80 meters default) and customize the calls to "***our_vehicle_obstacle_detected***" passing a different up and low angle to check for pedestrian, lane offset and the distance at which looking for.

In the following link it is possible to see the detection of a pedestrian also when it invades our lane suddenly. this link.

Bikers avoid manager

The possible colliding objects checked by this function are the bikers, it obtains the list of bikers in a range passed as parameter (80 meters default) and customize the calls to "***our_vehicle_obstacle_detected***" passing a different up and low angle to check for bikers, lane offset and the distance at which looking for. Is important that bikers are filtered as vehicle in Carla Simulator, so this filtering has been done in a more specifics way with respect to other objects, selecting only the category we are interested in.

In the following it is possible to see the detection of bikers. this link.

Static obstacle avoid manager

This function allows to particularize the checks on the possible colliding objects on the static objects such as barriers, cones etc. It obtains the list of this objects in a certain range passed as parameter (80 meters default) and customize the calls to "***our_vehicle_obstacle_detected***" passing a different up and low angle to check for objects, lane offset and distance at which looking for. It is important to specify that not all the static objects are considered as possible obstacles, but only that which are higher than 25 centimetres on which our vehicle cannot drive over.

In the following it is possible to see the detection of static object we considered as obstacles, so whose height is greater than 25 centimetres, in particular in this video are more than one closed static objects. this link.

Collision and car avoid manager

The possible colliding objects checked by this function are cars. It obtains the list of cars in a range passed as parameter (80 meters default) and customize the calls to "**our_vehicle_obstacle_detected**" passing a different up and low angle to check for cars, lane offset and distance at which looking for.

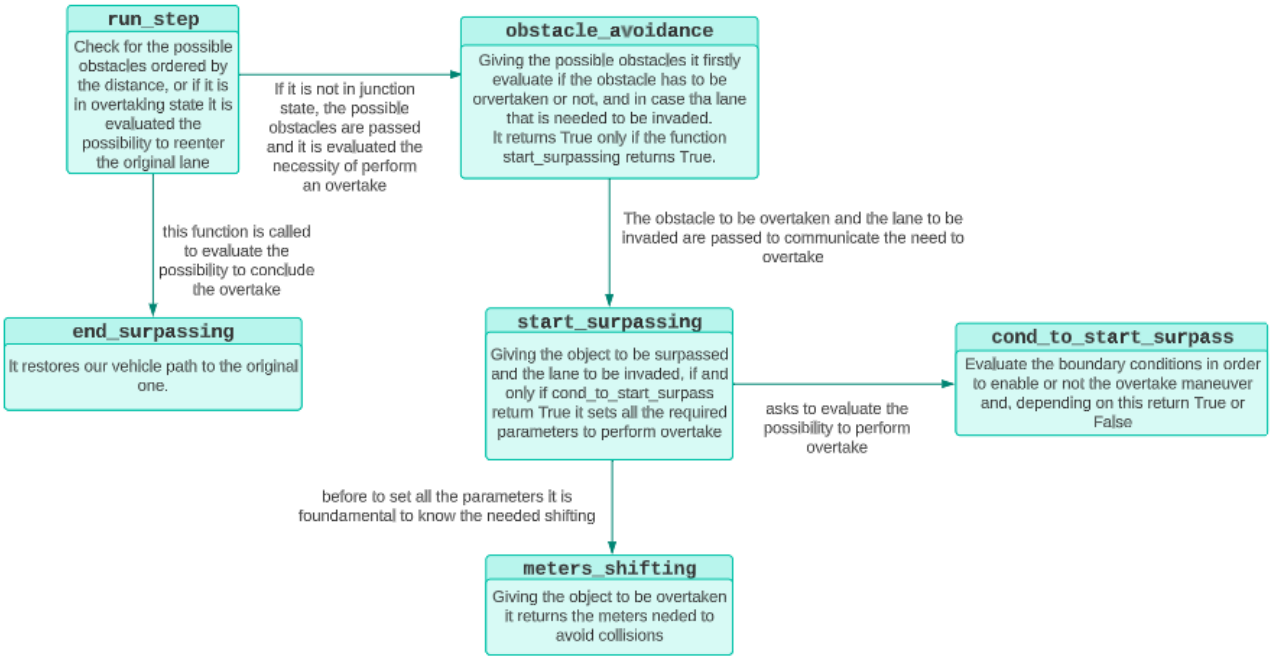
In the following it is possible to see an example of the detection of a stopped vehicle, even if this mechanism of detection obviously works independently of the speed of the target vehicle. this link.

Overtaking

As explained above, the characteristics of the ODD, so of the scenarios in which our autonomous vehicle driving algorithm must work, there are different situations that require to make overtakes. Just as it generally happens in real driving scenarios, the possible obstacles that need to be overtaken can be static obstacles or vehicles that are moving slower than us, such as bikers or other cars stopped on the road for different reasons. The management of overtaking was therefore necessary, not only in order to handle a situation that required a solution for the proper completion of the routes, but also, and above all, to make our algorithm truly robust in real-world scenarios. This is done with the perspective of developing, albeit in an extremely prototypical version, an algorithm that could be installed in a real vehicle.

So, in this section of the document, we will proceed with the explanation, including graphical schematizations, of the main peculiarities of the implementation, which can be divided into three different components. The three components are as follow: the analysis of boundary conditions that determine the feasibility of overtaking, the analysis of the space to deviate from the standard path that the vehicle needs to make in order to avoid the obstacle, and finally, the actual implementation of this deviation.

Here it is possible to see a scheme containing all the module involved for this task, highlighting the interconnections that exist between them.



All the scenarios present in the first five mandatory routes that required overtaking have been divided into two main categories:

- The first category involves one or more obstacles to be overtaken on our lane, that are moving in the same direction as us. Overtaking these obstacles requires temporarily encroaching into the adjacent left lane, where, as a common factor in all these situations, the traffic flow of vehicles is opposite to ours.
- The second category involves any road narrowing, for example due to the presence of cones, which forces vehicles from the adjacent lane travelling in the opposite direction to partially invade our

lane. In such situations, it is necessary for our vehicle to move to the right in order to avoid a collision with the invading vehicle. All these scenarios are characterized by the fact that the lane to the right of the one we are travelling on is not a drivable surface.

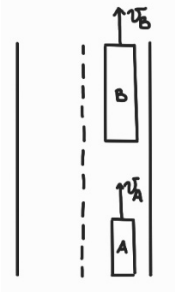
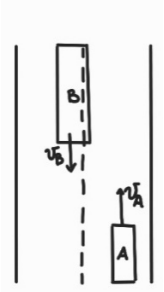
In the following we will refer to these two situations with *Case1* and *Case2* respectively.

Evaluation of the feasibility of overtaking

Before the overtaken manoeuvre can begin, it is first necessary to verify the possibility of completing it. This possibility is obviously analysed based on the actors and situations characterizing the surrounding space. This logic is contained within the function ***cond_to_start_surpass***, which is explained below. In order to estimate the feasibility of overtaking, it is first necessary to make an initial assessment of the obstacles that need to be overtaken, both in terms of quantity and sizes. Therefore, all static objects and vehicles (both bikers and cars) ahead of us are selected and sorted by distance. The goal is to evaluate the potential distance between the identified preceding and succeeding vehicles because if it is below a fixed threshold, it would require multiple overtakes (more than one obstacle). Once the set of vehicles to be overtaken is obtained, an estimation of the total space to be overtaken can be made, by evaluating the distance between our vehicle and the last one in that set. All distance evaluations have been carried out using Shapely library.

Then, the physical motion of both our vehicle and the objects to be overtaken was modelled. Specifically, all calculations were performed considering a relative motion in which the last vehicle to be overtaken was assumed to be stationary, Therefore, our velocity relative to the velocity of the last vehicle was analysed. In this evaluation was assumed that the obstacles to be overtaken, if there were more than one, were all united to each other.

In the following diagram, it can be seen what is meant by stating that the velocities were calculated relative to a reference system united to the vehicles to be overtaken. In the following schemes A refers to our vehicle and B refers to the obstacles to be overtaken, regardless of whether it is one or more than one.

Case1	**Case2**
	
$\boldsymbol{v_{Arelativa} = v_A - v_B}$	$v_{Arelativa} = v_A - v_B$

Considering this relative velocity of our vehicle, the motion of the latter has been modelled as uniformly accelerated motion, so starting from:

$$v_{relative\_final}{}^2 = v_{relative\_initial}{}^2 + 2a(s - s_0)$$

it is possible to obtain the final relative velocity of our vehicle, where the difference $(s - s_0)$ is set equal to the estimated distance to be surpassed, considering the distance between our vehicle and the last vehicle to be surpassed, the extension of this last vehicle and also an additional safety parameter that takes into account the extra space required to complete the manoeuvre. The acceleration considered is the value estimated to be the maximum possible acceleration of a vehicle, appropriately adjusted.

Then starting from:

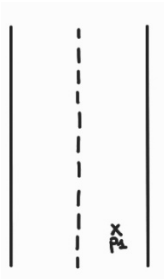
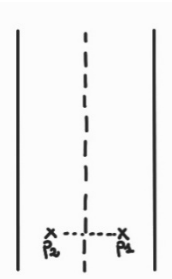
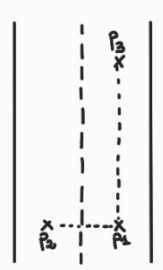
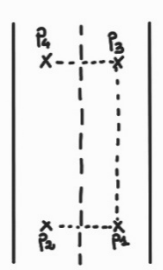
$$v_{relative\_finale} = v_{relative\_iniziale} + at$$

It is possible to obtain the time necessary to complete the overtake.

As specified above, all these calculations have been made considering a reference system united with the obstacles to be overtaken, so it is necessary to compute the real arriving point of our vehicle in the time and with the considered acceleration, with its real and not relative velocity. It has been obtained using the following formula:

$$s = s_0 + \frac{1}{2}(at^2) + v_0 t$$

where s_0 is supposed to be zero in order to compute the distance traveled by our vehicle in the time t that is the time, previous calculated in the relative context, required to complete the overtake, and with v_0 that is the real actual velocity of our vehicle and not the relative one respect to the objects to be overtaken. After obtaining the actual distance that will be covered during the overtaking maneuver, an evaluation of the boundary conditions was carried out. This involved assessing whether or not it was possible to perform overtaking maneuver without the risk of colliding with vehicles traveling in the lane to be invaded. In order to perform these calculations, specific points on the road were highlighted, which will be listed below in a sequence.

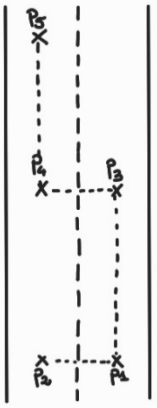
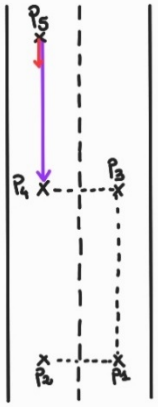
Step1	**Step2**	**Step3**	**Step4**
			
p1 is the waypoint corresponding to the actual position of our vehicle	*p2 is the waypoint related to the point corresponding to current position but in the adjacent lane to be invaded.*	*p3 is the waypoint in the lane of our vehicle, spaced apart from us of a distance equal to the estimated distance traveled by our vehicle traveled within the time required for the overtaking maneuver*	*p4 is the waypoint related to the point corresponding to p3 but in the adjacent lane to be invaded.*

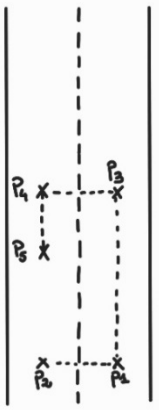
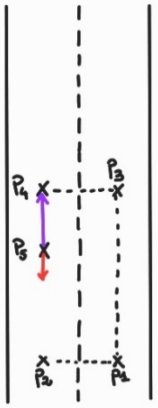
Therefore, with a good level of approximation (in the case of curved roads, for example), the distance between p2 and p4 will be considered equal to the distance between p1 and p3. From this scheme, it can be inferred that, with a good approximation, p2 and p4 have been considered as the starting and ending points respectively of the lane invasion on the adjacent side.

Obviously, the schematizations mentioned above refers to what we are indicating as *Case1*, where the lane to be invaded is the one adjacent on the left. Similar considerations have been made for *Case2* as well.

After that, the motion of the potential oncoming vehicle on the lane our vehicle will invade was modeled. In other words, a vehicle was selected from those present on the lane to be invaded, whose direction of travel, as reiterated several times, is opposite to ours, and it was chosen based on the proximity and being within a ninety-degree field of view, therefore all vehicles behind our have been excluded. Assuming that no suitable potential oncoming vehicle satisfying these conditions was found, the overtaking maneuver could proceed safely. However, if a suitable oncoming vehicle was found, its current position, instant velocity and instant acceleration were extracted for further analysis. For the motion of the potential oncoming vehicle was made the assumption of uniformly accelerated motion, considering instant acceleration, in order to evaluate its displacement. This assumption may not necessarily reflect the actual scenario accurately, but it is certainly a more robust approximation than modeling with uniform linear motion (with constant speed), thus providing a higher chance of success of overtake.

Below are the sequences that highlight the points and vectors considered for the evaluations of the availability of completing the overtake without collisions, starting from the previous represented points.

	Step5	**Step6**
Situation 1		
	*p5 is the waypoint corresponding to the position in which the oncoming vehicle is on the lane to be invaded. In this situation the position of the oncoming vehicle relative to p4 is **earlier**, in reference to its direction of travel.*	*The red arrow indicates the forward vector of the oncoming vehicle. The purple arrow indicates the vector connecting the points p4 and p5. In this situation these two vectors have the **same direction**.*

Situation 2		
	*p5 is the waypoint corresponding to the position in which the oncoming vehicle is on the lane to be invaded. In this situation the position of the oncoming vehicle relative to p4 is **later**, in reference to its direction of travel.*	*The red arrow indicates the forward vector of the oncoming vehicle. The purple arrow indicates the vector connecting the points p4 and p5. In this situation these two vectors have **opposite directions**.*

So, the magnitude of the purple arrow is the distance between the actual position of the oncoming vehicle with respect to the point where has been estimated the completion of the invasion of the other lane.

Using the formulas of uniformly accelerated motion mentioned above, and considering the current position, velocity, and acceleration of the oncoming vehicle, it has been calculated the time required to reach the point p4 and thus the potential collision point with our vehicle.

Then the dot product between the red and the purple vector was calculated, so that where this product was greater than zero, it means that the vectors' directions are in agreement, and in alternative case, it means that they are opposite.

Then it is concluded that the overtake is possible only if the time that our vehicle needs to go from p2 to p4 is lower than that the oncoming vehicle needs to go from p5 and p4, provided that the dot product between the two aforementioned vectors is greater than zero, otherwise it means that the oncoming vehicle is already in the "invasion zone", so between p2 and p4 (situation 2).

All these considerations made for *Case1* remain valid for *Case2* as well, with the exception that the check on the dot product is not necessary. In *Case2*, in fact, it involves partially invading the shoulder area of the road that is not traveled by any vehicles.

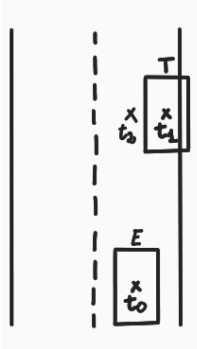
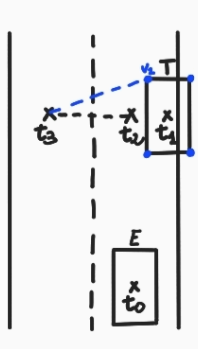
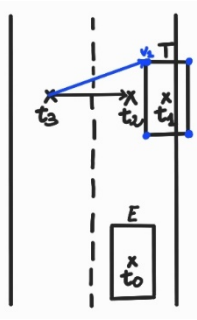
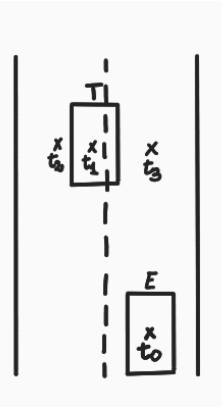
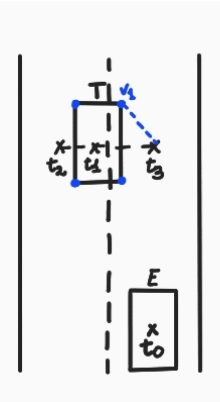
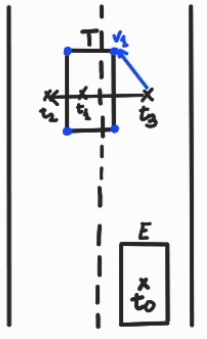
Everything that has been said so far represents the logic for evaluating the possibility of performing the overtaking maneuver. However, it was necessary to introduce an additional check aimed at managing anomalous situations dependent on the simulator itself. Specifically, it often happens within the simulator that vehicles with acceleration and velocity equal to zero, as they have just spawned into the world, quickly reach sustained speeds with physically impossible accelerations. This falsifies the checks described, leading to possible collisions. To handle these situations as well, a control was implemented. If the oncoming vehicle has a velocity below a certain threshold, which is dependent on the maximum speed of the road, it

is assumed to belong to this category. Therefore, it is assumed to be accelerating and an overestimation of its velocity and an underestimation of the time required to travel the distance between p4 and p5 are made. In this way, we have made our approach robust even in situations that may be possible in the simulator but not in reality.

Assessment of the space to move to avoid the obstacle

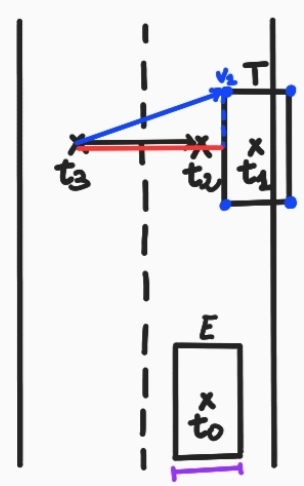
After identifying the obstacle and the need to overtake in the situation referred to as *Case1*, or to shift towards our right due to a narrowing in the situation referred to as *Case2*, it was necessary to determine how much this obstacle would encroach upon our lane. This was done to understand the extent of the right or left shift needed for our waypoints, thereby avoiding completely encroaching upon the adjacent lane.

Also in this case, schemes will be provided representing in sequence the points taken into account for the calculations.

	Step1	**Step2**	**Step3**
Scheme of *Case1*			
Scheme of *Case2*			
Description of both cases	*E is our vehicle and t0 the centre of it. T is the possible obstacle (bike, vehicle or static object) which forces us to deviate from our path, and t1 is its centre. t2 is the waypoint that corresponds to the projection of the centre t1 in the centre lane in which the vehicle has the majority part od itself. t3 is the waypint in corrispondence of t2 but in the adjacent lane.*	*It was computed the distance between t2 and t3 that with a good degree of approximation is equivalent to the dimension of a lane (considering that the road has lanes of equal width). Furthermore, among all the vertices of obstacle T, the one closest to point t3 has been selected, and has been labeled as v1 in the diagrams.*	*Then are determined two vectors, the first, with blue arrow, connecting t3 and v1 and the second, with black arrow, connecting t3 and t2.*

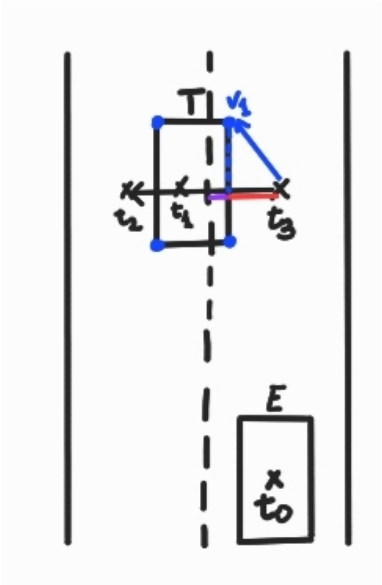
Step4

Scheme of *Case1*



So, it was computed the projection of the blue vector (t3-v1) on the black one (t3-t2), and the red line is obtained, then the quantity of which the obstacle T invade our lane is obtained making the difference, in absolute value, between the extension of our vehicle and this projection (the red line). This calculation is made considering that if the obstacle is perfectly centered in the lane our vehicle has to move from the center of the lane at least of its extension (indicated with the purple line), but greater is the projection indicated with red light and smaller is the portion of the obstacle that invades our lane, so a smaller shifting is required.

Scheme of *Case2*



So it was computed the projection of the blue vector (t3-v1) on the black one (t3-t2), and the red line is obtained, than the quantity of which the obstacle T invade our lane is obtained making the subtraction, in absolute value, between the half distance between t3 and t2, and the projection represented with the red line, so the obtained result is exactly the purple line.

Once the space occupied by the obstacle to be overtaken encroached upon our lane, the distance our vehicle had to deviate was evaluated by adding a safety constant to the invasion space calculated as explained above. This safety constant is dependent on the type of the object being overtaken. Furthermore, it should be noted that all representations and diagrams were made considering a single obstacle, but they were extended, without altering the logic, to the case of multiple objects.

Actuation of overtaking maneuver

The effective manage of the overtaking is implemented trough three main functions: ***obstacle_avoidance***, ***start_surpassing*** and ***end_surpassing***. These functions, along with some lower-level modifications, play a crucial role in enabling safe and efficient overtaking manoeuvres. The function **start_surpassing** is the one that initiates the overtaking manoeuvre, using positive feedback from other supporting functions. The function **end_surpassing** is the one that make the completion of the overtaking manoeuvre. The function **obstacle_avoidance** is the one responsible for detecting and handling obstacles that may impede the vehicle's path and so that considers the various conditions to determine when a change in the vehicle's trajectory is required to avoid a potential obstacle.

So, in this section of the report, each of these functions will be described in detail, outlining their specific roles, inputs and decision-making processes, in order to complete the actuation of overtaking manoeuvre. Additionally, it will be highlighted how we have modified at a lower level the planning to ensure smooth and reliable overtaking manoeuvre.

Obstacle_avoidance

This function is located within the behaviour planner and is responsible for detecting obstacles that need to be overtaken, requiring a movement towards the right or left, depending on the specific case. The possible obstacle to be overtaken are bikes, vehicles, and static objects. The first step is to analyse the vehicles found near us, considering only the closest one, provided that is distance from us is less than 15 meters. This choice is specifically made because the handling of possible collisions with any type of object requires a separate implementation. Therefore, braking or stopping in the presence of obstacles will be undertaken independently. However, the logic for overtaking should only start if the vehicle to be overtaken is indeed in a condition to be passed and is at a reasonably close distance from us.

Otherwise, if an object that satisfies this condition exists, an analysis of different scenarios is carried out to assess whether or not it is necessary to actuate a change in our vehicle's path, so information about vehicle's position and about the position of its corners are retrieved. Therefore, at this point, three different scenarios branch out:

- The object in question is completely on the same lane of our vehicle: in this case a lateral left displacement is required, the specific logic for the displacement varies depending on the type of object being overtaken. The specific analysis and implementation characteristic for each specific type of object are outlined in the following.

Case	Analysis
The obstacle is a bike	Considerations are made regarding the distance, the speed of the bicycle as well as its relative orientation to ours. Control over orientation is necessary to avoid initiating unnecessary overtakes, especially if it is a bicycle crossing the road on which our vehicle is moving, Control over speed is performed to determine if it is below a certain threshold. If the speed of the bikes is not above this threshold, overtaking would not be necessary.
The obstacle is a vehicle	Considerations are made regarding the distance and the speed of the vehicle. Control over speed is necessary to avoid overtaking vehicles whose speed is not below a certain threshold, Additionally, to prevent unnecessary overtaking manoeuvres and align with real-world scenarios, an additional check is performed concerning vehicle's light state. This includes checking if the lights indicate emergency situations, such as hazard lights or if the vehicle is stopped. If any of these conditions are met the vehicle is considered to be overtaken.
The obstacle is a static object	Considerations are made regarding the distance of the static object. In particular, all the static objects whose height is considered not so many high so that the vehicle can pass over them, are ignored.

The distance and the speed considered to determine if an object need to be overtaken are obviously dependent on the specific type of object. In the checks explained above has positive feedback, it is then used the function **cond_to_start_surpass**, above explained. Depending on the result of this function, and on the fact that our vehicle is or not already in an overtaken manoeuvre, is given the Boolean output by this function to start or not the overtake.

- The object in question is only *partially on the same lane* of our vehicle: in this situation it is necessary to determine if it is required a lateral right displacement or a left one, so there are two different possibilities reported in the following schema:

Case	Analysis
The obstacle is at the right of our vehicle	In this case it is necessary to evaluate the possibility to perform a lateral left shifting.
The obstacle is at the left of our vehicle	This is the situation we refer to *Case2* in this chapter, for which it is required a lateral right shifting of our vehicle. To understand this situation, it is evaluated if the dot product between the forward vector of our vehicle and those of that invading our lane are opposite. In this case, when the distance of our vehicle respect to the opposite is less than a certain threshold is evaluated the possibility to perform a lateral right shifting.

In particular, for both of the previous cases it is evaluate the positive or negative response of the function *cond_to_start_surpassing*, before start the shifting.

- **No one of the previous** situation: so, there is no need to change the path of our vehicle.

So, in summary this function returns a Boolean value indicating whether a lateral displacement has been initiated. If none of the conditions are met, it returns False, indicating that a lateral displacement is not currently necessary.

Start_surpassing

First of all, through the function *cond_to_surpass*, it is analyzed the possibility or not to do the overtake manoeuvre. If the response is positive, are saved a piece of information like the actual lane of our vehicle, then are set to the local planner some parameters like the operation to be done, and so a shifting to right or to left, the meters to shift required to complete the overtake without collisions, and also the speed required for our vehicle. In particular there are some differences if the shifting has to be made to right or to left. If our vehicle is in *Case1* it is set a speed that is the double of the speed limit of the road, because of the fact that during overtake manoeuvres like this, it is important that the duration of the invasion of the other lane is as short as possible. On the contrary, if our vehicle is in *Case2*, it is important that our vehicle slows down, so that no hasty manoeuvres are made that could result in our car going off the road. Furthermore, a parameter is set to indicate that our vehicle is currently overtaking, in order to prevent it from attempting another overtaking manoeuvre while engaged in such action.

Depending on the type of object to be overtaken it is also set a parameter that consider a security distance, which will determine the timing for returning to the original lane after overtaking.

End_surpassing

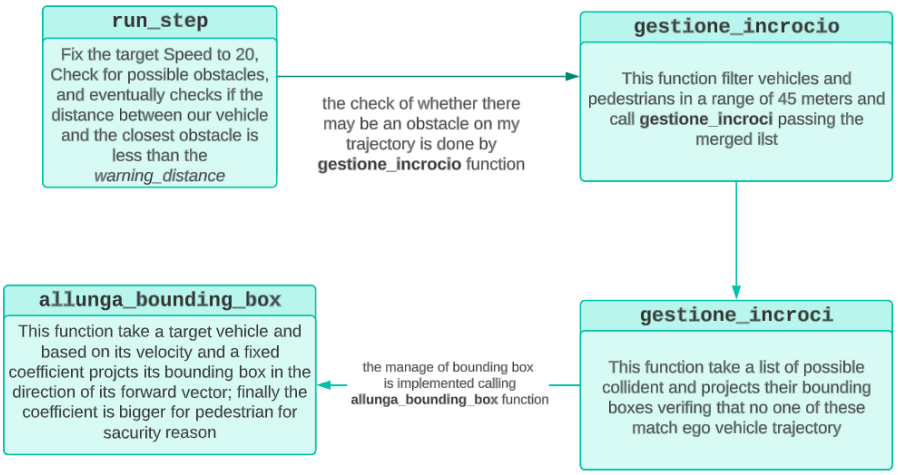
In this function the possibility of returning to the original lane after completing the overtaking manoeuvre is evaluated. Specifically, the absence of obstacles that would prevent the re-entry into the lane is considered. In the absence of such obstacles the lane change is performed only if the safety parameter, appropriately calibrated for each type of obstacle, has been exceeded. In order to restore the situation to its previous state before the overtaking manoeuvre, the characteristic parameters of our vehicle are reported to their previous values, thereby exiting the overtaking phase and, after a fixed number of time steps, allowing the potential enablement of other manoeuvre of this kind.

It is important to specify for both of these two functions that the safety parameter has been made dependent on the specific obstacle, as different types of obstacles have different dimensions, particularly different widths, and therefore require re-entry into the lane, just as it happens in reality, to be done at different angle of inclination.

In the following it is possible to see at least an example of overtaking, and in particular the situations to which we referred as *Case1*, for each type of possible obstacle to be overtaken and also of multiple obstacles. It is visible that the invasion of the adjacent lane is minimal and that the overtake starts only when all the boundary conditions allow it. this link, this link, this link.

There is also an example of that we referred to as *Case2*, so the necessity to perform a shifting due to a road restriction. this link.

Junction State



The management of intersections in the context of autonomous driving is a crucial aspect in ensuring the safety and efficiency of autonomous vehicles. Within the CARLA simulator, a strategy has been implemented to handle intersections and prevent collisions between vehicles. The following description provides a detailed analysis of the code that manages this situation. In the provided code, the management of intersections is specifically addressed when the autonomous vehicle and the incoming waypoint are located at an intersection. In this case, a reduced speed of 20 km/h is set using the ***set_speed*** method of the local planner. This approach slows down the vehicle to ensure greater safety during intersection crossing maneuvers. Subsequently, the ***gestione_incrocio*** function is called to evaluate if there are vehicles that could potentially cause a collision in the intersection. This function takes a list of actors and a maximum distance within which to consider waypoints for future projection of our vehicle. This list includes both vehicles and pedestrians to handle their presence within or near an intersection. Within the ***gestione_incrocio*** function, the following steps are performed: First, information about the autonomous vehicle, such as its transformation and current waypoint, is obtained using the functions provided by the CARLA simulator; Next, a polygon representing the predicted path of the autonomous vehicle is generated. This is achieved by using the bounding box of the vehicle and the subsequent waypoints. The polygon is created by projecting the bounding box of the vehicle onto the next three waypoints. Additionally, the coordinates of the polygon vertices are calculated to perform subsequent collision checks.

For each actor in the provided list, the following steps are executed:

a. The distance between the autonomous vehicle and the considered vehicle is checked. If the distance exceeds a predefined threshold (4 times the maximum distance parameter), the vehicle is excluded from evaluation.
b. The bounding box of the considered vehicle is extended along the X-axis based on its speed. This extension takes into account the vehicle's dimensions and velocity to represent an anticipated occupancy area.
c. A polygon representing the extended bounding box of the considered vehicle is generated.
d. The intersection between the polygon representing the path of our vehicle and the modified (we discuss that below) polygon representing the potential colliding vehicle is checked. If an intersection occurs, a potential collision is detected.
e. If the considered vehicle is inside the real bounding box of our vehicle, the angle between the forward vectors of the two vehicles is calculated. If the angle is less than **57 degrees** and the considered vehicle is ahead of our vehicle, a potential collision risk is detected.

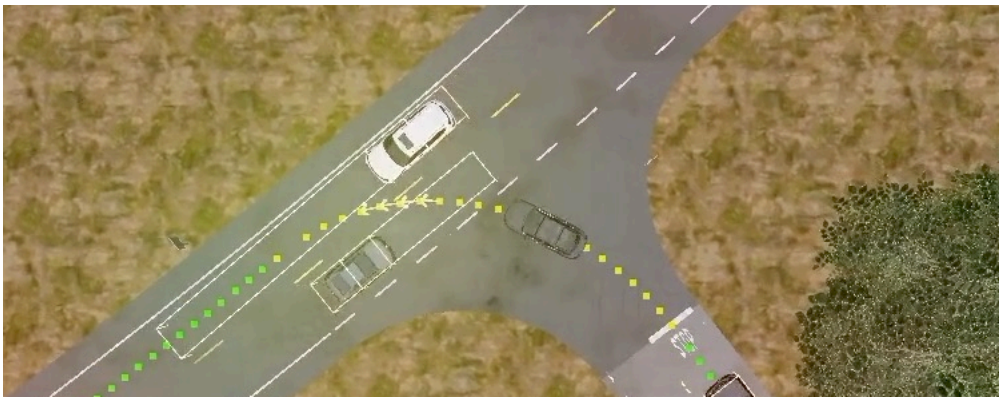
f. In the event of a detected collision, a controlled braking procedure is executed for the autonomous vehicle to avoid the collision. If the considered vehicle is sufficiently close, emergency braking is applied.
g. If no collision is detected, vehicle management is performed based on the distance between our vehicle and the considered vehicle. This management may include the "***car following***" procedure to maintain a safe distance from the vehicle ahead.

At the end of the function, the result indicating the presence or absence of obstacles in the intersection, the vehicle that could cause a collision (if present), and the distance between the autonomous vehicle and the considered vehicle are returned. An example of junction can be seen at this link

Allunga_bounding_box

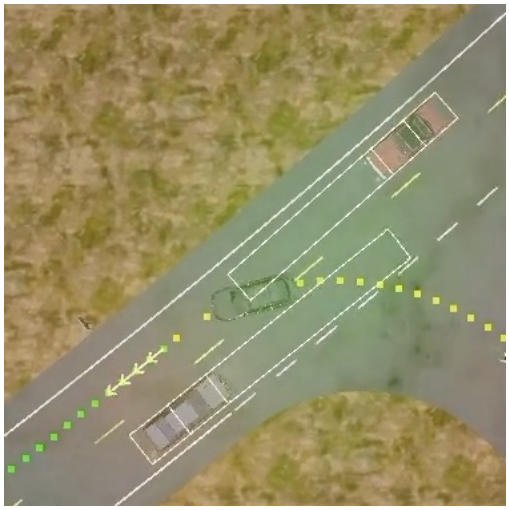
The ***allunga_bounding_box*** function is used to extend the bounding box of an actor along the X-axis, using the forward vector of the actor itself. This extension takes into account the vehicle's speed to represent the anticipated occupancy area based on its dynamics. By analyzing the bounding boxes of actor and the polygons representing the path of the autonomous vehicle, potential collisions are detected, and necessary measures are taken to prevent them. This approach contributes to ensuring safety and efficiency in autonomous driving at intersections.

Figure 45. Above view of junction



In the image below, we can see an example of the "extended bounding box" that we consider, along with the three barely visible waypoints. The calculation of the bounding box and the analysis of collisions are performed at each run step. This allows for immediate response if a vehicle in the intersection decides to yield and slow down, causing the elongation of the bounding box to decrease. Another check is performed when the current bounding box of the vehicle intersects with the extended bounding box, examining the angles between the forward vectors and the position of the other considered vehicle, in order to determine the vehicle's position and appropriate behavior. Specifically, if the current bounding box of the vehicle collides with the extended bounding box, if the next waypoint is closer to us, and if the angle between the two forward vectors is less than ***57 degrees***, it indicates that the vehicle is in a subsequent position within the intersection compared to the other vehicle and has already occupied the road (as shown in right figure). Therefore, there is no need to stop, only to continue on its path. The example below perfectly illustrates this situation. As we can see, our vehicle is "subsequent" within the intersection compared to the other vehicle, and with an angle below ***57 degrees***, we have ensured that it has already entered the lane and is aligning with it.

Figure 54. Above view of junction



In contrast, this other scenario highlights the need to check the distance from the next waypoint, specifically: the bounding box returned by Carla is not always accurate. In this particular case, despite our

Figure 66. Frontal view in junction

distance from the following vehicle, an intersection is detected between our bounding box and the elongated bounding box of the vehicle in front of us and the vehicle follow the previously descripted behavior, so it continues to accelerate causing a collusion. To handle this situation, we have decided to use the distance from the waypoint as an additional discretization factor. As we can see, the following vehicle is immersed within the waypoint, resulting in a shorter distance compared to ours. This allows us to easily determine when a vehicle is in front of us and manage the collision accordingly. An example is available at this link.

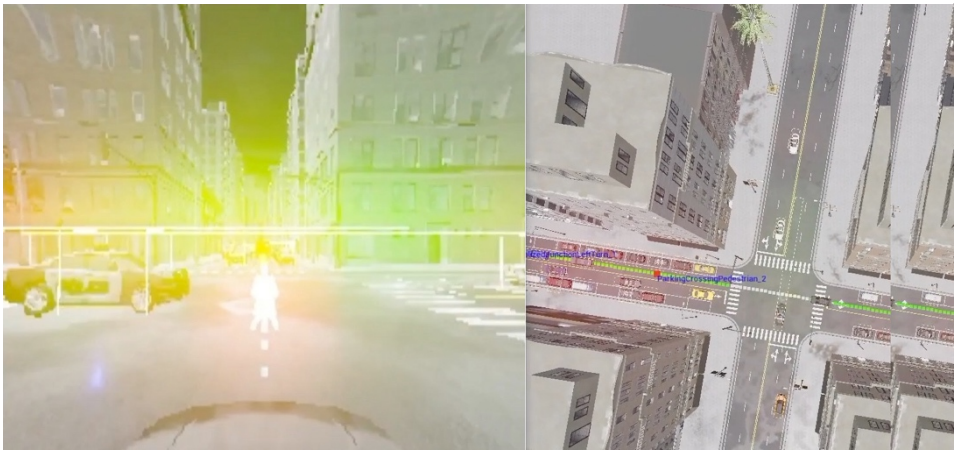
The following screenshot illustrates the management of pedestrians in the intersections. As mentioned earlier, pedestrians and vehicles are considered in a similar manner, with the only difference being the parameter multiplied by the pedestrian's speed to elongate the bounding box, even though they have a lower velocity. This allows us to anticipate the pedestrian's movement and stop in time. From this perspective, we can also see the elongated bounding boxes of other vehicles. Below the screenshot of the walker situation and the link at the relative video.

Figure 77. Manage Walker in Junction

Another implementation choice made was to slightly shift the generated bounding box forward. This solution essentially allows us to "restart" more quickly when a vehicle passes in front of us in the intersection, increasing the vehicle's dynamics in crossing the intersection and passing more swiftly.

Features extracted from extra routes

Figure 88. Elongated bounding box behaviour



From the extra routes, we evaluated the system's behavior in a specific scenario where our vehicle has a green traffic light, but another vehicle runs a red light. In the screenshot below, you can observe how the extension of the bounding box allows us to "anticipate" the path of the violating vehicle, enabling us to stop promptly and avoid collisions. This approach, which involves extending the bounding box, enables the system to detect vehicles that run red lights and take the necessary measures to prevent accidents. The extension of the bounding box expands the surveillance area and provides more accurate information about the intentions of other vehicles, allowing our vehicle to react proactively and safely. An example is available at this link.

Car Following Manager

The `***car_following_manager***` function is responsible for managing car-following behaviours when there is another vehicle in front of the autonomous car. Its purpose is to ensure safe and smooth driving by adjusting the speed of the autonomous car based on the behaviour of the leading vehicle. First, the function calculates the speed difference between the autonomous car and the leading vehicle. It then determines the Time-to-Collision (TTC) by dividing the distance between the vehicles by the speed difference. The maximum acceptable TTC is defined by the safety time parameter. If the TTC is within the safety time range, indicating that the autonomous car is too close to the leading vehicle, the function reduces the target speed gradually to match that of the leading vehicle. The new target speed is set, and the local planner is instructed to adjust the speed accordingly. In the case where the TTC is within a certain range above the safety time, the function tries to maintain a speed similar to the leading vehicle, ensuring a safe distance is maintained. The target speed is adjusted accordingly, and the local planner executes the necessary steps. Finally, if the autonomous car is neither too close to the leading vehicle nor within the specified safety time range, it behaves normally, maintaining a target speed equal to the speed limit. The local planner continues to run step-by-step without any major adjustments. By dynamically adjusting the speed based on the behavior of the leading vehicle, the `***car_following_manager***` function promotes safe and efficient car-following behavior, allowing the autonomous car to adapt to different traffic situations on the road.

In the following link it is possible to see an example car following management in action. Here it is possible to see that the speed of our vehicle is dependent on the speed of the vehicle that is in front of us in order to respect always a minimum distance from it. this link.

Test

In order to assess the performance and effectiveness of our autonomous vehicle driving algorithm, extensive testing was conducted using both the Stanley controller and the PID controller as lateral controllers. This section presents the results of these tests, which were performed on the set of the mandatory routes and one additionally for the extra ones. In particular, we employed five different machines for the experiments relative to the mandatory routes, including four machines utilized by members of our team and one machine provided by the university. For each machine involved in the testing, detailed hardware specifications were recorded to ensure consistent evaluation and analysis. These specifications include OS, the processor type, graphics card model, RAM capacity and the type of storage used (SSD or HDD). The variations in hardware among the different machines provide insights into the potential impact of computational resources on the performance of the control algorithms. On every machine we executed two runs of the 5 mandatory routes of the project (so, totally 10 routes) and evaluated the results of every single route and the global scores.

TEST - PID

This section includes the tests made using PID controller as lateral controller. In the following link it is also possible to see the video of the five mandatory routes with this lateral controller this link. Is also available the raw one, so without any speed up. The results given by the five different machines are here.

Test Machine 1

- Ubuntu 18.04.6 LTS
- Intel Core i7 – 6700 CPU 3.4 GHz x 4
- NVIDIA GeForce GTX 960
- RAM 16 GB
- HDD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	95.579952	96.858214	84.629097	100.0	100.0	**MEAN:** 96.098872
	ROUTE COMPLETION	100	100	100	100	100	100.0
	INFRACTION PENALTY	0.9558	0.968582	0.846291	1.0	1.0	0.960989
RUN 2	DRIVING SCORE	92.744604	96.864053	94.312795	100.0	100.0	**STD_DEV:** 4.811
	ROUTE COMPLETION	100	100	100	100	100	0.0
	INFRACTION PENALTY	0.927446	0.968641	0.943128	1.0	1.0	0.048

Test Machine 2

- Ubuntu 20.04.5 LTS
- AMD Ryzen 5 3600 3.6 Ghz x 6
- AMD RX 580
- RAM 16 GB
- HDD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	94.363739	95.870752	94.876472	100.0	100.0	**MEAN:** 95.1592 100.0 0.951595 **STD_DEV:** 6.272 0.0 0.063
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.943637	0.958708	0.948765	1.0	1.0	
RUN 2	DRIVING SCORE	0.941482	93.110393	79.225633	100.0	100.0	
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.941482	0.931104	0.792256	1.0	1.0	

Test Machine 3

- Ubuntu 20.04.6 LTS
- Intel Core i7 – 1165 G7 CPU 2.8 GHz x 4
- NVIDIA MX 450
- RAM 16 GB
- SSD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	93.700464	97.229951	75.504137	100.0	100.0	**MEAN:** 93.28691 95.22 0.980295 **STD_DEV:** 10.272 10.688 0.027
	ROUTE COMPLETION	100	100	76.1	100	100	
	INFRACTION PENALTY	0.937005	0.9723	0.99217	1.0	1.0	

Test Machine 4

- Ubuntu 20.04.6 LTS
- AMD Ryzen 7 5800HS 3.2 GHz x 8
- NVIDIA GeForce RTX 3060
- RAM 16 GB
- SSD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	94.148209	94.950873	92.27506	100.0	100.0	**MEAN:** 95.884364 100.0 0.958844 **STD_DEV:** 3.708 0.0 0.037
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.941482	0.949509	0.922751	1.0	1.0	
RUN 2	DRIVING SCORE	93.688848	92.97286	90.807786	100.0	100.0	
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.936888	0.929729	0.908078	1.0	1.0	

Test Machine 5

University servers.

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	93.311736	97.237059	59.904755	100.0	100.0	**MEAN:** 90.090718 100.0 0.90090718 **STD_DEV:** 17.095 0.0 0.171
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.933117	0.972371	0.599048	1.0	1.0	

TEST - STANLEY

This section includes the tests made using Stanley controller as lateral controller. In the following link it is also possible to see the video of the five mandatory routes with this lateral controller this link. Is also available the raw one, so without any speed up The results given by the five different machines are here.

Test Machine 1

- Ubuntu 18.04.6 LTS
- Intel Core i7 – 6700 CPU 3.4 GHz x 4
- NVIDIA GeForce GTX 960
- RAM 16 GB
- HDD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	92.825517	97.2504	95.034899	100.0	100.0	**MEAN:** 95.112685 100.0 0.951127 **STD_DEV:** 6.74 0.0 0.067
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.928255	0.972504	0.950349	1.0	1.0	
RUN 2	DRIVING SCORE	92.825517	95.570952	77.799483	99.820081	100.0	
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.928255	0.95571	0.777995	0.998201	1.0	

Test Machine 2

- Ubuntu 20.04.5 LTS
- AMD Ryzen 5 3600 3.6 Ghz x 6
- AMD RX 580
- RAM 16 GB
- HDD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	95.603417	93.437956	96.349	100.0	100.0	**MEAN:** 97.1111193 100.0 0.971112 **STD_DEV:** 2.796 0.0 0.028
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.956034	0.93438	0.96349	1.0	1.0	
RUN 2	DRIVING SCORE	95.603417	92.831336	97.286801	100.0	100.0	
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.956034	0.928313	0.972868	1.0	1.0	

Test Machine 3

- Ubuntu 20.04.6 LTS
- Intel Core i7 – 1165 G7 CPU 2.8 GHz x 4
- NVIDIA MX 450
- RAM 16 GB
- SSD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	94.883185	93.992207	96.214	100.0	100.0	**MEAN:** 97.017878 100.0 0.970179 **STD_DEV:** 2.835 0.0 0.028
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.948832	0.939922	0.96214	1.0	1.0	

Test Machine 4

- Ubuntu 20.04.6 LTS
- AMD Ryzen 7 5800HS 3.2 GHz x 8
- NVIDIA GeForce RTX 3060
- RAM 16 GB
- SSD

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	95.579952	98.029	88.529279	100.0	100.0	**MEAN:** 95.533311 100.0 0.955333 **STD_DEV:** 6.34 0.0 0.063
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.9558	0.98029	0.885293	1.0	1.0	
RUN 2	DRIVING SCORE	95.140516	97.448868	80.605497	100.0	100.0	
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.951405	0.974489	0.806055	1.0	1.0	

Test Machine 5

University servers.

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	98.446084	98.407	66.898639	100.0	100.0	**MEAN:** 92.750345 100.0 0.927503 **STD_DEV:** 14.473 0.0 0.145
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.984461	0.98407	0.668986	1.0	1.0	

About both test, that with Stanley lateral control and that with PID lateral control, it is important to specify that the machine 3,like the University servers, and unlike the others, performs only one run because of limited computational power and because of the excessive time required.

Results

In this chapter of the report, it will be reported the final results of both our system, that having PID as lateral controlled and that having Stanley, evaluated on the mandatory routes but also on the extra ones. It is important to underline that the effort has been done, for time limited resources, for the optimization of the mandatory routes, while the extra one have been only used for evaluating the generalization capability of our system.

PID FINAL RESULTS

The files json with these PID results are visualizable here and the implementation can be found here.

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	94.363739	93.202137	89.938496	100.0	100.0	**MEAN:** 95.500874 100.0 0.955009
	ROUTE COMPLETION	100	100	100	100	100	
	INFRACTION PENALTY	0.943637	0.932021	0.899385	1.0	1.0	**STD_DEV:** 4.416 0.0 0.044

TEST N°	SCORES	ROUTE 8	ROUTE 9	ROUTE 10	ROUTE 11	ROUTE 12	GLOBAL
RUN 1	DRIVING SCORE	0.152272	14.3748	1.00893	29.48075	8.728875	**MEAN:** 10.749125 74.106 0.170624
	ROUTE COMPLETION	100	66.55	3.98	100	100	
	INFRACTION PENALTY	0.001523	0.216	0.2535	0.294808	0.087289	**STD_DEV:** 11.992 41.792 0.122

STANLEY FINAL RESULTS

The files with these Stanley results are visualizable here and the implementation can be found here.

TEST N°	SCORES	ROUTE 0	ROUTE 1	ROUTE 2	ROUTE 3	ROUTE 4	GLOBAL
RUN 1	DRIVING SCORE	93.103201	96.157304	96.349	100.0	100.0	MEAN: 97.121901
	ROUTE COMPLETION	100	100	100	100	100	100.0
	INFRACTION PENALTY	0.931032	0.961573	0.96349	1.0	1.0	0.971219
							STD_DEV: 2.926
							0.0
							0.029

TEST N°	SCORES	ROUTE 8	ROUTE 9	ROUTE 10	ROUTE 11	ROUTE 12	GLOBAL
RUN 1	DRIVING SCORE	1.253622	12.96	3.887552	29.970796	1.826344	MEAN: 9.979641
	ROUTE COMPLETION	100	100	93.66	100	100	98.732
	INFRACTION PENALTY	0.012536	0.1296	0.041506	0.299708	0.018263	0.100323
							STD_DEV: 12.127
							2.835
							0.121

Starting from the inspection of the reported tables it seems clear that there is a marked difference between obtained results. In particular the performance on the extra routes undergoes a visible decline. It is important to specify that, even if the percentage obtained is very low, the functionalities implemented and described above work also in the extra routes. Obviously, the scenarios of extra routes have different pitfalls and so require different values of some fixed parameters or the managing of not considered situations. This means that, even if the number of collisions, or timeout scenario are so high, the logic for junctions, overtaking, stop, and the detection of all types of obstacles continue to work, but require an appropriate tuning of parameter in order to reach a

correctly conclusion. So, the generalization capability of our systems is not universal, but limited to the ODD defined by the mandatory routes. However, we are conscious of the fact that having more time would have also allowed us to achieve greater generalization, which currently remains a commitment for future developments.

The analysis of the entire system has been made in a way that was valid for both implementations, that with Stanley and that with PID as lateral control. In particular we choose to develop both implementations leaving the logic for all the principles maneuvers unchanged. Based on the tests conducted and the above final results for both implementations, a series of considerations will now be made to identify critical issues and strengths of each one, and to subsequently choose the one that we believe to be the best.

There is an important structural difference between the two controllers, specifically, the Stanley controller performs control by considering both the heading error and the crosstrack error, unlike the PID controller which focuses only on steering. This difference means that the Stanley controller has greater control in situations such as curves, allowing the vehicle to navigate them correctly while maintaining a good speed. This likely positively influenced the results obtained in terms of *min_speed*, which appears to be the determining factor in the difference between the results of the two proposed implementations.

Due to the structure of the Stanley controller, when the speed of our vehicle is very high (as in situations like overtaking), the steering inputs become less pronounced, with the same crosstrack error. This aspect, which is generally an advantage for vehicle stability, becomes a disadvantage when it comes to quickly completing the maneuver of overtaking, especially when it involves crossing into a lane with oncoming traffic (as in our routes). In such cases, it may be necessary to make more abrupt movements in order to avoid collisions. This aspect would lead us to prefer the PID controller, which does not have such limitations.

To conclude, since the purpose of our system is to closely resemble real-world systems, the maneuver stability provided by a lateral controller like Stanley offers greater assurance. For this reason, in agreement with the best performance observed during testing, we agree that Stanley is the more suitable implementation among the two. In fact, properly developed, allows excellent maneuvering speed and the absence of collisions, also in the previously mentioned critical situations.

Conclusions

Throughout our project on autonomous vehicle driving, we have been able to tackle complex challenges and develop innovative solutions. A key factor that facilitated our progress and enabled us to effectively address these challenges was the utilization of the CARLA simulator. CARLA provided us with a powerful and realistic virtual environment that allowed us to test and refine our algorithms and systems in a safe and controlled manner. One of the significant advantages of using CARLA was the ability to leverage its built-in simplifications and abstractions of the real-world driving environment. The simulator offered a comprehensive set of simplified models and representations of various elements, including road infrastructure, traffic dynamics, and sensor simulations. This allowed us to focus on specific aspects of our project, such as control algorithms and decision-making strategies, without being overwhelmed by the complexity and variability of real-world scenarios. The simulator allowed us to have no difficulty in taking all the information we needed for our autonomous driving algorithm to work without problems. All this would not be applicable ad-hoc in the real world where, to give a trivial example, we could not easily trace in real-time the position, speed and acceleration of other vehicles present in the world or we could not easily have the precise location and status of every object in the world. Ideally, if we have the possibility of constantly receiving this information from the real world, our algorithm would be applicable and would work giving the results already found on CARLA. Surely despite the facilitation given by the simulator, the implementation of various policies such as overtaking or intersection management was not easy as both turn out to be very complex tasks that can include a very large number of cases that can hardly reside perfectly in an algorithm. Precisely from this discourse, the problem of the level of generalization of our algorithm arises; as we were able to experiment especially with the extra routes, which we didn't fully rely on when implementing the code but were mostly used as a "test set" of our algorithm, what has been implemented is very much in overfitting with the mandatory routes that have been assigned to us. In this argument, there is also a discourse on the randomness that has been found in the system: on the various machines on which the tests have been carried out the problems have been various and different; the randomness resides a lot in the actors of the world, which on different servers could be present or not in some situations and, for example in the crossings, the change in the quantity of these actors could give rise to new problems that had not been noticed. Surely our focus was to create a code that could take into account the possible cases we could run into and that was as generalizable as possible. Furthermore, it is necessary to take into account the fact that the CARLA scenarios are in any case not perfectly consistent with reality; at intersections and when overtaking, the behaviour adopted by the other players in the world is in any case that of following their own path, without considering other situations that are taking place, therefore referring to the two cases mentioned above, we expect that in real scenarios during an overtaking the vehicles in the opposite lane start to decelerate, which did not happen in the simulator, while as regards intersections we expect that if our vehicle has already entered the intersection, the other vehicles present will wait for it. Other problems encountered relate to the CARLA data which were sometimes obsolete and could cause problems for our algorithm, an example is the case of vehicle spawning in which it happens that if we are stopped waiting to overtake, the vehicles that spawn in opposite lane have very high speeds and accelerations which could prevent us from carrying out our calculations correctly; problems that, obviously, would not be encountered if the vehicles were already moving and did not spawn near us. A separate discussion must be made for the MinSpeed problems that we obtained in the results; this infraction is due to the fact that our vehicle goes slower than the pace of the cars around us in the simulator. This problem has been analyzed and an attempt has been made to resolve it: we have not noticed unnecessary slowdowns from our vehicle and during all the routes we always try to maintain a speed close to the road limit without exceeding it; moreover, in general, in the real world we have no impediments relating to the

minimum speed on the road which is only present on the motorway, so we expect our algorithm to behave correctly from this point of view. We preferred not to increase the speeds too much to fix this infraction as it would have made driving too careless and could have caused other more serious issues that could have led to collisions. Another very particular case study encountered during the tests is that of pedestrians and bikers with quite anomalous behaviour; these situations were taken into consideration by our algorithm as it happened that while they did not invade our street, they fell into the street or behaved dangerously by suddenly throwing themselves into the street; despite the quite anomalous behaviour, our algorithm handles such situations to avoid infringements. We therefore believe that we have achieved an excellent result on the mandatory routes that had been assigned to us. A separate discussion for the extra ones, in which there are problems not seen in the mandatory ones and which have not been dealt with currently, but which could be included in possible future developments.

Future Developments

As described in the previous chapter, there are still some critical issues or situations that have not been addressed. These have primarily been identified in the extra routes, which, as agreed upon, are not evaluative and have not been the focus of our implementation efforts. However, what we have observed is that in scenarios present in both types of routes (mandatory and extra), the behavior appears to be correct, except for some cases that can be easily managed with fine-tuning of the parameters used. Therefore, we outline below the specific areas and their corresponding future developments:

- **Intersection Management**: The expansion of this area focuses on the calculation related to the new bounding box created to project other vehicles in the intersection. Similar to other management aspects, one of the main modifications we intend to develop is to use more precise calculations, leveraging kinematics, to determine the extension of the bounding box for other vehicles to ensure collision detection. Currently, through fine-tuning, we lengthen the bounding box proportionally to the vehicle's speed. Another improvement is related to correctly calculating the distance between our vehicle and the vehicle with which we would potentially collide. Currently, when a potential future collision is detected, we immediately enter the emergency stop state, abruptly halting our vehicle. It would be beneficial to calculate the actual distance to allow our vehicle to advance further into the intersection, enabling quicker intersection management. Another development pertains to "predicting" intersections. Currently, to enter the logic of intersections, we only consider the current and next waypoints. However, it would be necessary to have a mechanism to look further ahead and adjust the vehicle's behavior based on the distance to the intersection. For example, we could slow down if we realize that the intersection is a certain distance away, ensuring safer navigation. Nonetheless, there are still cases where we fail to complete the routes accurately. These cases, however, appear to be fairly random (having occurred a very limited number of times) and therefore not entirely attributable to our development efforts. We want to highlight two specific cases that still occur occasionally. The first case occurs in route 0. Specifically, there are instances where the preceding vehicle becomes stuck within the intersection, nearly colliding with another vehicle already occupying it. This causes an exception and leads to a significant penalty in the score calculation. In general, the behavior of other drivers at intersections aligns closely with reality. Since entering the intersection is approached cautiously for maximum safety, other drivers have ample time to slow down and allow us to proceed safely. However, there is one specific intersection where the traffic rules are not fully respected. This situation occurs in route 4 at the penultimate intersection. Specifically, while in all other analyzed intersections, when we are occupying a lane within the intersection, other vehicles detect our presence and correctly yield, allowing us to pass safely by stopping their vehicles. However, in this particular intersection, that behavior is not contemplated. The vehicles continue their course even if we occupy the intersection, which, in some cases, results in a rear-end collision with our vehicle. Although this collision would be legally attributed to the vehicle that hits us, from a simulation perspective, it still counts as an incident, leading to a significant loss of points for that specific route.
- **Vertical signage management**: One of the key developments in this field is related to the analysis of multiple types of road signs. Currently, our system primarily focuses on traffic lights and stop signs, but there is a need to consider other signs such as speed limits, road work warnings and accident notifications. An initial area of focus is the integration of speed limit signs into our algorithm. Currently, this information is obtained directly from the simulator, which may not reflect real-world conditions where speed limit data is not readily accessible or constantly updated in real-time. While advanced navigation systems (like Google Maps) can provide some information about the road and

corresponding speed limits, relying solely on this data is insufficient as road work or temporary changes can impact speed limits. Therefore, incorporating speed limit sign detection and recognition is crucial for enhanced autonomous driving capabilities. Similarly, the management of road work signs is also important to address. By detecting and recognizing road work signs, our system can proactively adjust its behavior and prepare for potential obstacles or changes in the road ahead. A similar approach can be applied to accident notification signs, allowing our autonomous vehicle to anticipate potential hazards and adapt its driving strategy accordingly. Overall, the current management of the considered signs, such as traffic lights and stop signs, is robust in various scenarios. However, an interesting idea worth exploring is the modification of our traffic light management to better understand the timing and duration of the yellow signal. By accurately assessing whether it is still safe to proceed during a yellow signal, our vehicle can enhance its behavior and make more socially acceptable driving decisions. During the analysis of the extra routes, it was found that relying solely on stop signs observation is not sufficient; it is also necessary to consider the horizontal road markings, which are not represented as “actors” in the simulation. In a specific route, the presence of horizontal stop markings was identified, but the absence of vertical signs led to inappropriate behaviour of the autonomous vehicle, as it did not come to a stop and crossed the intersection at an excessive speed. However, the horizontal road markings are currently not represented as road actors in the system, which means that resolving this issue would require further research to understand how to detect such markings. These developments, however, are not necessary for the current context. It might be necessary to examine the OpenDrive specifications, which could provide insights on how to handle horizontal road markings. This is an additional feature that is currently not present in the current version of the system but could be considered for future developments and improvements. However, in the final score calculation, the leaderboard does not detect this type of issue, as the "RunningStopTest" is zero, indicating a lack of handling of this problem, likely due to the absence of the "STOP" actor.

- **Overtaking**: The logic behind overtaking has been extensively discussed in the corresponding chapter, and it remains a robust and substantial development. However, there are future advancements to be made in implementing overtaking within the same lane. The current development was focused on resolving the basic routes, which only considered two-lane roads, while extra routes present scenarios with four-lane roads. Implementing this new feature of perform overtaking encroaching a lane whose direction is the same as our, would not require significant effort as the existing functionality can be reused, with appropriate discretization for both scenarios. Specifically, the modification involves a simple change in the "where" we look when assessing overtaking opportunities. Currently, we look ahead, but in this case, we would need to look behind. This modification is minimally invasive, but due to time constraints, the need for extensive testing (which exceeds the development time required), and the fact that these situations only occur in the extra routes, it was decided not to include this function in this specific release. In the extra routes, there have also been some cases where overtaking fails to initiate. Due to logistical reasons, further investigation into the underlying causes has not been conducted.
  The current limitations in the implementation of overtaking cannot be solely attributed to our development efforts. During implementation, we encountered several "anomalous" situations within the simulator that caused significant challenges—situations that would not occur in the real world. The primary concern is related to vehicle spawning. One of the main issues identified in the simulator relates to vehicle spawning, particularly when vehicles are spawned with a speed of zero and then instantly have an unrealistic acceleration, reaching too much fast the road's maximum speed limit. Since our algorithm relies on vehicle kinematics and considers the speeds of vehicles in

the other lane to calculate the feasibility of overtaking, having a vehicle with a speed of zero initiates overtaking correctly, resulting in a severe collision. Through additionally checks, we have currently managed to address these edge cases. However, future developments will focus on analyzing this specific challenge. One approach under consideration is to develop an algorithm capable of detecting these "false positives" and discarding the retrieved data to ensure that we are indeed considering a vehicle with the correct speed, rather than a newly spawned vehicle. Additionally, within the simulator, similar to the intersection scenario mentioned earlier, traffic rules are not always adhered to during overtaking. Article 148 of the traffic code states that vehicles should facilitate overtaking maneuvers without accelerating, but in the simulator, vehicles in the other lane continue at the same speed, ignoring the ongoing overtaking maneuver. A similar observation can be made regarding the overtaking of cyclists. Even though they may be keeping to the right side, there is still room for a safer overtaking maneuver.