Enhancing Autonomous Driving Capabilities Using the CARLA Simulator

Andrea Vincenzo Ricciardi (2009), **Giovanni Rolando** (2018), **Luigi Emanuele Sica** (0522501530), **Andrea Zinno** (2064)

[1]Dept. of Information Eng., Electrical Eng. and Applied Mathematics - University of Salerno, Italy

Abstract In the context of this university project, we developed and improved an autonomous driving agent using the CARLA simulator. Everything began with a basic agent with limited navigation capabilities, which simply followed the predetermined path and braked in the presence of pedestrians, traffic lights, and vehicles, while it collided with obstacles and failed in situations where different decisions were required. To address these limitations, we exclusively used the information provided by CARLA to perceive the environment and from this, we implemented numerous driving logics to handle the challenges present in the various provided scenarios. Our main objective was to enhance the basic agent by equipping it with advanced driving capabilities to manage complex situations such as dealing with bicycles, overtaking parked vehicles, handling intersections, reacting to road cones, and other logics not present in the basic agent. The result of this produced a more robust and versatile agent, capable of handling a wide range of road scenarios with safe driving behaviors within the given project domain.

For correspondence:

a.ricciardi38@studenti.unisa.it (AVR); a.zinno6@studenti.unisa.it (AZ); g.rolando1@studenti.unisa.it (GR); l.sica29@studenti.unisa.it (LS)

1 | Introduction

Autonomous driving represents one of the most ambitious and complex challenges in the field of artificial intelligence and robotics. Our university project fits into this context with the goal of developing and improving an autonomous driving agent using the CARLA simulator, an advanced environment for simulating realistic driving scenarios.

The starting point of our work was a basic agent with very limited navigation capabilities. This agent was able to follow a predetermined path and brake in the presence of pedestrians, traffic lights, and vehicles, but failed to manage unexpected obstacles and situations requiring complex decisions. To better understand the challenges our agent would face, we initially analyzed and identified the main driving scenarios to manage, which include:

- **Lane changing.** The ability of the agent to safely switch lanes when necessary.
- **Negotiations at traffic intersections.** Handling the complex dynamics of intersections, including yielding to other vehicles and deciding when to proceed.
- **Handling traffic lights and traffic signs.** Correctly interpreting and responding to various traffic signals and signs to ensure compliance with road rules.
- **Coping with pedestrians, cyclists, and other elements.** Safely navigating environments with vulnerable road users and other unpredictable elements.

To address the complex situations of autonomous driving, we exclusively utilized the information provided by CARLA to perceive the surrounding environment. Based on these data, we developed and integrated a series of advanced driving strategies, systematically tackling the challenges and progressively improving the agent's performance.

The project's objective was to create a versatile agent capable of adapting to different driving contexts and ensuring a high level of safety and compliance with traffic regulations. Through an iterative approach, we refined the agent's capabilities, moving closer to human-like driving behaviors.
In this document, we will proceed to analyze in detail the logic adopted for the various driving scenarios, illustrating how we designed the agent's behavior in these contexts. Subsequently, we will provide an overview of our system's architecture, explaining the different phases of the agent's decision-making process. Finally, we will describe the practical management of the various scenarios encountered during development and conclude with a comparison between the initial basic agent and the final developed agent, highlighting the improvements and results achieved.

1.1 | Methodology

In this section, we describe in detail the methodology adopted to enhance the autonomous driving capabilities of our agent using the CARLA simulator. We tackled a series of complex driving scenarios, implementing advanced techniques to ensure the agent could navigate safely and efficiently. Our methodology is divided into three main parts: identification of driving scenarios, implementation strategies, and utilization of information provided by CARLA. In the following chapters, these parts will be explored in greater detail, providing a deep understanding of the techniques and considerations underlying our autonomous driving agent.

Identification of Driving Scenarios The first section of our study focuses on the identification of driving scenarios. Through an in-depth analysis of the simulations, we define the critical driving contexts that our agent must be able to recognize and manage. This identification process is crucial to prepare the agent to navigate in a dynamic and unpredictable environment.

Implementation Strategies Next, we outline the implementation strategies we adopted to address the various driving scenarios. This section describes our methodical approach to translating the identified challenges into concrete driving logics, enabling the agent to act effectively and promptly.

Utilization of Controllers and Information Provided by CARLA We examine how we used the information provided by CARLA to guide the agent's decisions. We discuss the use of PID and Stanley controllers and how they were managed.

2 | Driving Scenarios Identified

During the initial phase of the project, we identified the main driving scenarios that the autonomous agent must handle. These scenarios represent common and critical situations that an AV encounters on the road. An accurate description of the various scenarios is presented below.

Pedestrian Management We highlighted, during the scenario review, the presence of pedestrians, particularly in curves and behind objects that might not be immediately visible to the agent. For such scenarios, we need to manage a gradual reduction in speed when pedestrians are detected, until a critical threshold is reached, at which point an emergency stop must be executed.

Traffic Lights and Stop Signs Our base model does not initially include traffic lights, but it is designed to accommodate their presence. The system is expected to evaluate the traffic light's color: if green, the vehicle proceeds without interruptions; if red, it stops. Similarly, when approaching a stop sign, the vehicle slows down and comes to a complete stop.

Static Objects There are also static objects such as cones or warning signs present. If they occupy our lane, we plan to introduce maneuvers like overtaking or lateral offset, if possible, to avoid completely invading the opposite lane.

Junctions Management Junctions, preceded by a stop sign, require the management of stop signs and different types of intersections. These are handled based on the position and movement of surrounding vehicles, merging into the flow when it is safe to do so.

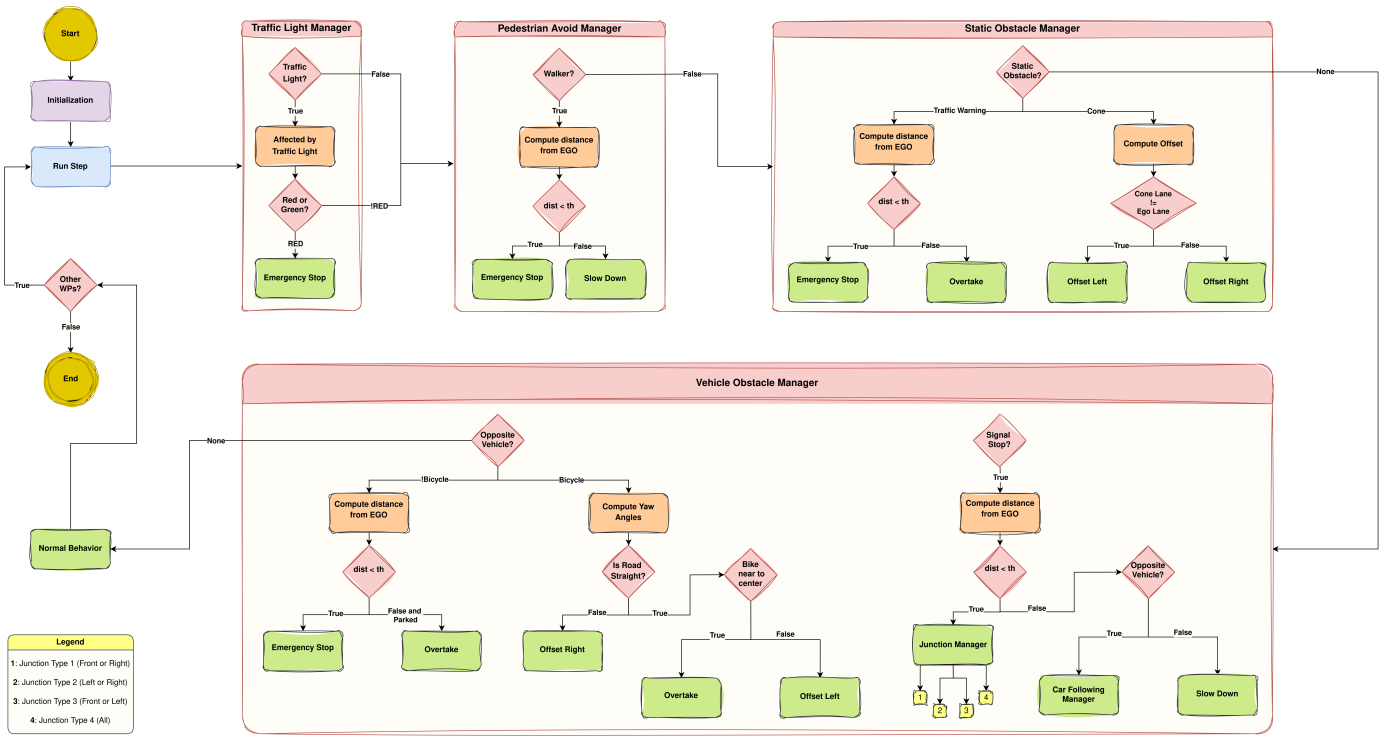

Fig. 1. Architecture of our agent's logic and identified driving scenarios.

Vehicles and Bicycles There are also vehicles parked along the roadside and bicycles either traveling alongside our agent or crossing the road suddenly. This scenario requires managing overtakes or lateral offsets and adaptive cruise control.

In conclusion, the analysis of the tested routes allowed us to address and manage driving scenarios such as overtaking, intersection management, lateral offsets, and dynamic overtakes based on the distance from other vehicles. All necessary information was acquired using the methods provided by CARLA, which allowed us to control the actors and determine their positions within the simulation context.

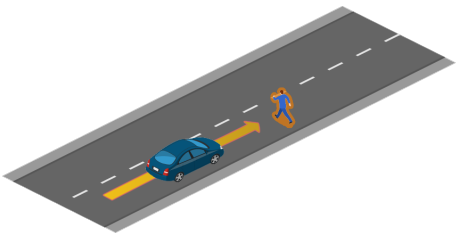

Fig. 2. The ego-vehicle encounters a pedestrian on the road and must perform an emergency brake or an avoidance maneuver.

3 | Implemented Strategies

In this section, we describe the implementation strategies used to address the various identified driving scenarios. Based on the architecture in Fig. 1, we developed specific **managers** for each scenario, responsible for handling their respective functions. Below, we detail the implementation of each manager.

3.1 | Pedestrian Manager

The **Pedestrian Manager** is responsible for the safe management of pedestrians. Based on Fig. 2 and Fig. 3, we observed two possible cases: when pedestrians suddenly cross the road and when they are near a curve.

Fig. 3. While performing a maneuver, the ego-vehicle encounters an obstacle in the road, either a pedestrian or a bicycle, and must perform an emergency brake or an avoidance maneuver.

Therefore, we defined a detection radius of 12 meters for pedestrians. The system constantly monitors the distance between the vehicle and the pedestrian, so that, once detected at this distance, the vehicle has time to slow down. If the pedestrian enters the road (into the vehicle's area of interest) and the distance is below a minimum threshold, emergency braking is activated. To ensure proper functioning even in curves, we increased the vehicle's frontal vision angle. If the distance is greater than the threshold, the vehicle slows down slightly and proceeds to the next waypoint, still ensuring a high level of safety and respect for pedestrians. To ensure proper functioning even on curves, we increased the vehicle's frontal field of view angle. If the distance is greater than the threshold, the vehicle slightly slows down and proceeds to the next waypoint, still ensuring a high level of safety and respect for pedestrians.

3.2 | Stop Sign Manager

The **Stop Sign Manager** handles the vehicle's proper response to stop signs along the route, as shown in Fig. 4. The logic is similar to that used for pedestrians: we detect the stop sign at a distance of 20 meters. Once detected, the system constantly monitors the distance from the sign, slowing down the vehicle as it approaches. If the detected distance is less than a predefined threshold of 2 meters, indicating the immediate proximity of the stop sign, the vehicle performs an emergency stop to comply with traffic regulations. Otherwise, if the distance is greater than the threshold, the vehicle gradually slows down, continuing to proceed towards the next waypoint.

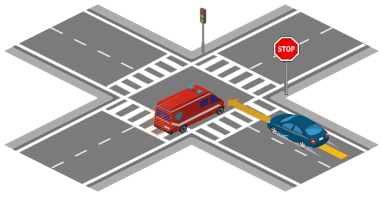

Fig. 4. This image shows the scenario where, in the presence of a stop sign, the vehicle slows down as it approaches until it comes to a complete stop.

Additionally, we perform a check in case there are vehicles ahead of us. If there are stationary vehicles, we verify whether they are stopped due to a stop sign, a traffic light, or another issue. This check was necessary to prevent our vehicle from attempting to overtake by seeing the stopped vehicle ahead. We manage this situation by also checking the reason why the vehicles ahead are stopped, thus avoiding risky overtaking maneuvers. The same approach is applied when we encounter an intersection; in this case, we do not attempt overtaking.

3.3 | Traffic Light Manager

The **Traffic Light Manager** is responsible for the vehicle's interaction with traffic lights along the route. Similarly, we detect all traffic lights at a distance of 50 meters to manage possible urban situations with heavy traffic around traffic lights. When the vehicle approaches a traffic light, the system analyzes its state. If the traffic light is red, the vehicle performs an emergency brake to stop safely and comply with traffic signals. If the traffic light is green, the vehicle continues to proceed towards the next waypoint without interruption. This approach ensures that the vehicle obeys traffic rules and maintains safe and predictable behavior in the presence of traffic lights.

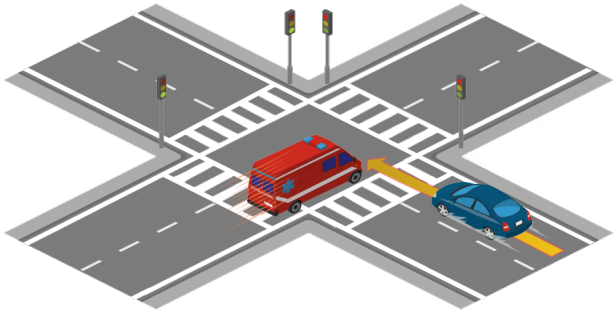

Fig. 5. Ego-Vehicle loses control due to bad conditions.

3.4 | Static Obstacle Manager

The **Static Obstacle Manager** handles obstacles as shown in Fig. 6 and Fig. 7. These obstacles include traffic warning signs and cones. Obstacles are treated differently depending on whether they completely block our lane, partially occupy it, or partially occupy the opposite lane.

Overtake If the obstacle blocks the entire lane, the agent will perform an overtaking operation by dynamically creating waypoints to follow. To implement the overtaking maneuver, we devised a comprehensive logic that considers various factors such as the geometry of the road, the speed of the ego vehicle, and the presence of obstacles or other vehicles. Here's an overview of the key elements of our approach:

- **Overtake Distance Calculation**. We first calculate the distance that the ego vehicle needs to travel to overtake the obstacle in front of it. We employ Pythagorean theorem to approximate the diagonal distance required to change lanes during overtaking. Here, we model the lane width and the length of the ego vehicle as the two sides of a right triangle, with the hypotenuse representing the distance to be traveled while changing lanes. Mathematically, the total overtaking distance d_{overtake} is computed as follows:

$$d_{\text{overtake}} = d_{\text{obstacle}} + d_{\text{same}} + 2 \times \text{hypotenuse} + d_{\text{other}} \quad \textbf{1}$$

  where d_{obstacle} represents the distance from the obstacle to be overtaken and d_{same} denotes the distance traveled on the same lane before initiating the lane change. In addition, we employed a dynamic approach to calculate the distance to be traversed on the overtaking lane, considering the distances between parked vehicles encountered beyond the first one (refer to the code for clarification), denoted as d_{other}. This dynamic calculation adapts the overtaking distance based on the actual spacing between parked vehicles, ensuring a flexible and responsive overtaking maneuver that navigates efficiently through the available space on the road.

Fig. 6. The ego-vehicle encounters an obstacle blocking the lane and must perform a lane change into traffic to avoid it. The obstacle may be a construction site, an accident or a parked vehicle.

- **Overtake Time Estimation**. With the overtaking distance known, we then estimate the time it will take for the ego vehicle to complete the overtaking maneuver. This is calculated using the uniformly accelerated rectilinear motion equation, which takes into account the initial speed of the ego vehicle, its acceleration, and the distance to be covered. The equation is as follows:

$$\text{Overtake Time} = \frac{-v_0 + \sqrt{v_0^2 + 2 \cdot a \cdot s}}{a} \quad \textbf{2}$$

  where v_0 is the initial speed of the ego vehicle, a is its acceleration, and s is the overtaking distance.
- **Collision Checking**. We perform collision checks to ensure the safety of the overtaking maneuver. In our implementation, we make the assumption that vehicles in the opposite lane are moving at the speed limit of the road. This assumption allows us to predict their positions based on their constant speed. We then calculate the distance to these vehicles within a certain range ahead of the ego vehicle's path. If the ego vehicle intersects with any of these vehicles during the overtaking maneuver, we abort the overtaking attempt to prevent accidents.

Assuming no collisions are detected and the road conditions permit, we generate a path for the overtaking maneuver. This path includes waypoints for smoothly changing lanes, overtaking the obstacle, and returning to the original lane.

Lateral Offset In the case of lane narrowing, such as the presence of cones that do not occupy the entire lane, the agent will manage them with a lateral offset, positive (+0.2) or negative (-2), depending on whether they are on our lane or the opposite lane. This allows maintaining traffic flow and ensuring the vehicle's safety. Subsequently, once the cones are passed, we return to the original offset of 0.

Fig. 7. The ego-vehicle encounters an oncoming vehicles invading its lane on a bend due to an obstacle. It must brake or maneuver to the side of the road to navigate past the oncoming traffic.

3.5 | Junction Manager

The **Junction Manager** plays a crucial role in ensuring the safe and efficient navigation of the autonomous vehicle through intersections. Upon detecting an intersection nearby, the system's first operation is to categorize the intersection into one of four main types:

- **Type 1**: The junction allows straight or left turns.
- **Type 2**: The junction allows straight or right turns.
- **Type 3**: The junction allows left turns or right turns.
- **Type 4**: The junction allows movement in all directions.

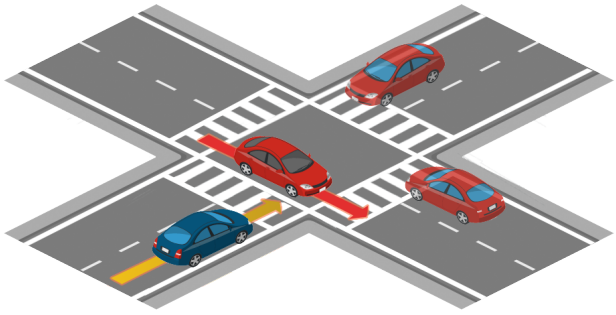

Fig. 8. The ego-vehicle needs to negotiate with other vehicles to cross. This scenario occurs at both signalized and non-signalized junctions.

Upon detecting an intersection nearby, the system identifies the junction point and its type. Leveraging CARLA's feature allowing only one vehicle to enter the intersection at a time, the Junction Manager ensures safe traversal. It considers various scenarios based on the junction type and vehicle's intended direction. The logic for identifying the junctions includes searching for flashing lights of vehicles at exit waypoints. This approach capitalizes on the standard behavior of vehicles to activate turn signals when intending to change direction, providing a reliable indicator of their planned trajectory. By monitoring the presence and status of these turn signals, the Junction Manager gains valuable insight into the intentions of other vehicles within the intersection, facilitating more informed decision-making and reducing the risk of collisions. For each scenario, the Junction Manager employs a systematic approach tailored to the specific characteristics of the intersection:

- **Type 1 (Front Left)**: Vehicles must navigate situations where they can proceed straight or turn left. By analyzing the presence and behavior of vehicles in front and to the left, the Junction Manager determines the appropriate course of action, prioritizing safety while minimizing delays.
- **Type 2 (Front Right)**: In intersections allowing vehicles to either turn right or proceed straight, the system assesses the movement of vehicles ahead and to the right, ensuring safe passage by yielding when necessary and proceeding when the path is clear.
- **Type 3 (Left Right)**: Handling intersections accommodating left and right turns, the Junction Manager evaluates the behavior of vehicles in the left and right lanes, adjusting its approach to avoid potential conflicts and ensure smooth traversal.
- **Type 4 (All Directions)**: Complex intersections with movements in all directions require comprehensive analysis of all paths. By considering the behavior of vehicles in front, to the left, and to the right, the system navigates intersections safely, prioritizing caution in the presence of potential conflicts.

3.6 | Vehicle Manager

The **Vehicle Manager** handles both automobiles and bicycles. When our vehicle is behind another vehicle moving with the flow of traffic, the agent will follow the vehicle while maintaining a safe distance (adaptive cruise control/car following). However, we have developed a system that comprehends if the vehicle in front of us is parked, near a stop sign, or stopped for a traffic light, allowing us to understand the reason for its immobility and adapt our response accordingly. If the vehicle ahead is parked, the agent will execute an overtaking maneuver. Overtaking is not performed on curves, which are considered dangerous.

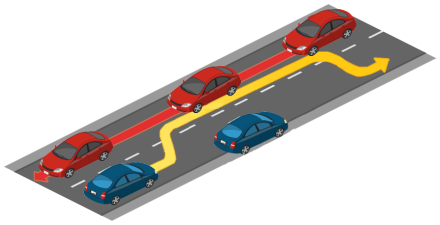

Fig. 9. A car is parked on the side of the road, partially blocking the lane. The ego-vehicle must overtake it.

Regarding bicycles, the decision depends on their position on the road and the distance, similar to how it occurs with pedestrians, slowing down when detected. If bicycles are on the right, the agent overtakes with a lateral offset of 2 meters, evaluating the possibility of offset based on the curve. If bicycles are in the center of the lane, overtaking occurs only if not on a curve. Curve recognition is achieved by analyzing the spatial and angular coordinates of upcoming waypoints: significant variations indicate the presence of a curve, determining caution in overtaking maneuvers.

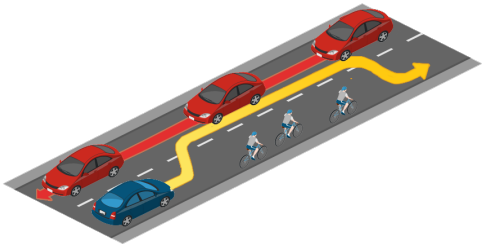

Fig. 10. The ego-vehicle encounters a slow moving hazard blocking part of the lane. The ego-vehicle must brake or maneuver to avoid it next to a lane of traffic moving in the opposite direction.

4 | CARLA Controllers

During our work with the **CARLA** simulator, we employed a series of methods to acquire and interpret the necessary information for the agent's navigation and environment perception. We used the `filter` method to identify specific types of actors within the simulation, such as emergency vehicles, pedestrians, and other relevant elements. To understand distances and predict potential collisions, we relied on bounding boxes, which allowed us to determine the physical boundaries of objects and their relative positions to the agent. This was crucial for developing obstacle avoidance systems and ensuring safe driving. Additionally, we leveraged waypoints and junctions to guide the agent through the simulation's roads, ensuring it followed logical paths and adhered to traffic rules. Waypoints were particularly useful for defining precise trajectories and managing the agent's behavior near intersections. Lastly, we examined the yaw angles of objects, indicating their orientation in space, and used `get_location()` and `get_transform()` to obtain detailed information about the position and orientation of actors. This data was essential for enabling the agent to fully understand its context and react accordingly. These tools and methods provided us with a comprehensive picture of the simulation environment and equipped the agent with the necessary information to make informed decisions and navigate effectively within CARLA's virtual world.

We also paid attention to PID and Stanley controllers, adjusting the values to find a compromise for optimal driving. After several attempts, we reached a configuration that proved to be effective in a variety of scenarios, although not perfect in all cases. Since parameter optimization requires numerous trials, we stopped the process when we achieved generally satisfactory results.

PID and Stanley Controller Parameters

```
"longitudinal_control_dict" :{"K_P": 0.888, "K_I": 0.0768, "K_D": 0.05,
↪  "dt": 0.03}
"lateral_control_dict" : {"K_V": 4.0, "K_S": 1.0, "dt": 0.03}
```

Listing 1. Longitudinal and lateral configuration for our scenarios.

5 | Results

In this section, we will examine and compare the results obtained from the baseline agent and the final agent in two distinct scenarios. The analysis of the results highlights a significant improvement in the performance of the final agent compared to the baseline agent. This progress is manifested in a greater ability to adapt to different situations and in overall safer driving. The comparison table Table 1 illustrates the differences between the baseline agent and our final agent in terms of driving scores, infraction scores, and detected infractions. It clearly demonstrates the superior performance of our approach across both Scenario 1 and Scenario 4. In scenarios where the baseline agent fails, such as colliding with bicycles or obstacles, our agent demonstrates improved performance by effectively avoiding collisions and navigating through obstacles.

Fig. 11. In this scenario, as depicted in the image, the baseline agent collides with a bicycle. Conversely, our agent successfully executes an offset maneuver to avoid the collision.

Fig. 12. The image shows a situation where the baseline agent collides at a checkpoint. In contrast, our agent stops before the checkpoint and subsequently creates an alternative route for overtaking.

Fig. 13. In this scenario, cones obstruct the opposite lane. The baseline agent fails to facilitate the passage of oncoming vehicles, resulting in collisions. Conversely, our agent effectively manages this situation.

Fig. 14. The image depicts a scenario where the baseline agent struggles with traffic warning obstacles. In contrast, our agent navigates around obstacles and, when feasible, plans overtaking maneuvers.

Fig. 15. In this scenario, pedestrians are present in a curve. The baseline agent fails to manage the pedestrians, while our agent anticipates the curve and applies brakes in advance

However, it's important to acknowledge that our approach is not flawless. One notable limitation is its strong correlation and dependence on the speed of the ego-vehicle. While our agent excels in certain scenarios, such as avoiding collisions with pedestrians and vehicles or adhering to traffic signals, there are instances where it may struggle, particularly when faced with high-speed maneuvers or complex traffic situations.

Scenario	Agent	Average Driving Score	Average Infraction Score	Detected Infractions
1	Ours	85.63	0.856	No collisions with pedestrians or vehicles; No infractions for red lights or stop signs; Some episodes of exceeding the minimum speed limit.
1	Baseline	5.52	0.455	Collisions with urban furniture objects; Exceeding the time limit for completing the route.
4	Ours	98.81	0.988	Exceeding the minimum speed limit once.
4	Baseline	1.94	0.019	Collisions with pedestrians and vehicles; Violations of traffic signals; Route timeout.

Table 1. Comparison between the Base Agent and the Final Agent in scenarios 1 and 4.

6 | Future Developments

One potential future development of the project involves integrating advanced sensors to make the simulation environment even more realistic. The use of sensors such as LiDAR, radar, and high-resolution cameras could significantly enhance the autonomous agent's ability to perceive and understand the surrounding environment. These sensors would enable more precise detection of objects and road conditions, facilitating safer and more efficient driving decisions.

Autoware In this context, integrating Autoware, an open-source platform for autonomous driving, could represent another significant development. Autoware offers a wide range of functionalities for perception, path planning, and vehicle control, making it a valuable resource for our project. By using Autoware, we could leverage advanced algorithms and existing modules, speeding up the development process and improving the performance of our autonomous agent. Adopting a standardized platform like Autoware would also facilitate interoperability with other systems and sensors, promoting greater collaboration and data exchange across different projects and institutions.

Connected Vehicles and Smart Cities Furthermore, within the realm of connected cars, it is anticipated that every vehicle will be equipped with advanced communication and perception technology, akin to what is offered by the CARLA Simulator. Connected cars will be able to share real-time information about vehicles, pedestrians, and obstacles present in the scene, enhancing the situational awareness of each vehicle. This collective perception capability will help create a safer and more coordinated road environment, reducing the risk of accidents and improving traffic flow. These developments will not only increase the realism of simulations but will also have a significant impact on research and development of technologies for autonomous driving, bringing the reality of connected and autonomous vehicles closer to practical implementation on a large scale. The combination of advanced sensors, vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2I) communication, along with the use of platforms like Autoware, allows us to explore new frontiers in the safety and efficiency of autonomous driving, contributing to a future where autonomous vehicles will be a fundamental component of urban mobility.