






## Article

# Enhancing Autonomous Driving in Urban Scenarios: A Hybrid Approach with Reinforcement Learning and Classical Control

Rodrigo Gutiérrez-Moreno , Rafael Barea , Elena López-Guillén , Felipe Arango , Fabio Sánchez-García and Luis M. Bergasa 

R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, F. Sánchez-García and L.M. Bergasa are with the Electronics Department, University of Alcalá (UAH), Spain. {rodrigo.gutierrez, rafael.barea, elena.lopezg, juanfelipe.arango, fabio.sanchezg, luism.bergasa}@uah.es

\* Correspondence: rodrigo.gutierrez@uah.es

**Abstract:** The use of Deep Learning algorithms in the domain of Decision Making for Autonomous Vehicles has garnered significant attention in the literature in the last years, showcasing considerable potential. Nevertheless, most of the solutions proposed by the scientific community encounter difficulties in real-world applications. This paper aims to provide a realistic implementation of a hybrid Decision Making module in an Autonomous Driving stack, integrating the learning capabilities from the experience of Deep Reinforcement Learning algorithms and the reliability of classical methodologies. This work encompasses the implementation of concatenated scenarios in simulated environments, and the integration of Autonomous Driving modules. Specifically, the authors address the Decision Making problem by employing a Partially Observable Markov Decision Process formulation and offer a solution through the use of Deep Reinforcement Learning algorithms. Furthermore, an additional control module to execute the decisions in a safe and comfortable way through a hybrid architecture is presented. The proposed architecture is validated in the CARLA simulator by navigating through multiple concatenated scenarios, outperforming the CARLA Autopilot in terms of completion time, while ensuring both safety and comfort.

**Keywords:** Autonomous Driving; Deep Reinforcement Learning; Decision-Making; Vehicle Control; CARLA Simulator.

**Citation:** Gutiérrez-Moreno, R.; Barea, R.; López-Guillén, E.; Arango, F.; Sánchez-García, F.; Bergasa, L.M. Enhancing Autonomous Driving in Urban Scenarios: A Hybrid Approach with Reinforcement Learning and Classical Control. *Journal Not Specified* **2024**, *1*, 0. <https://doi.org/>

Received:

Revised:

Accepted:

Published:

**Copyright:** © 2024 by the authors. Submitted to *Journal Not Specified* for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The rapid progression of Autonomous Driving (AD) has made significant inroads into both industry and academia. It has evolved into one of the most intensively researched fields, experiencing exponential growth in recent years [1]. Despite this, most vehicles currently include only Advanced Driver Assistance Systems (ADAS) as a pathway to achieve full AD. However, the ongoing advancements in AD, coupled with an analysis conducted by McKinsey [2], indicate that ADAS and AD combined could potentially contribute a substantial \$300 billion to \$400 billion to the passenger car market by the year 2035. This insight underscores the profound significance of AD in shaping the future of the automotive industry. Without a doubt, these improvements in Autonomous Vehicles (AVs) promise to reduce the number of accidents on the road. Over a million people die in traffic-related accidents each year. By removing human factors from vehicle control, AD could significantly enhance traffic safety [3].

In the field of AD, Decision Making (DM) is undoubtedly one of the most critical aspects influencing the vehicle's behaviour. The ability to determine the appropriate action based on the surrounding environment, in a manner that is both safe and efficient, is a fundamental concern in this domain [4]. It is universally recognized that every AD system incorporates a DM module, aimed at minimizing human errors in driving tasks. This research is centred on these crucial aspects, to develop a framework for autonomous DM in realistic scenarios within the AD context.

Deep Reinforcement Learning (DRL), a subset of Machine Learning (ML), emerges as a promising candidate to help in the decision task [5]. As delineated above, AD challenges within DM offer a clear avenue for exploration. DRL appears to be an ideal tool for the task, given its close alignment with the Partially Observable Markov Decision Process (POMDP), its adaptability to diverse domains, and its applications in DM engineering. At its core, DRL combines Reinforcement Learning (RL) with Deep Neural Networks (DNNs). By integrating the principles of Bellman's equation with the advanced capabilities of Neural Networks (NNs), DRL can analyse environments and infer the optimal decision for a given set of inputs, surpassing the capabilities inherent to traditional RL. This process occurs through repeated trial and error experiments as humans do, during which an agent incrementally acquires a desired behaviour by continuously interacting with its environment.

However, the application of DRL techniques does not always satisfy the requirements of AD. As mentioned before, the main problem to be solved by AVs is related to traffic safety, where DRL approaches may have some difficulties due to their primary focus on optimizing a task without considering comfort and safety. These techniques are divided into model-based and model-free. Model-based methods incorporate vehicle dynamics but frequently encounter concerns, such as high computational demands and the complexity of creating accurate environmental models, which compromise reliability in unpredictable scenarios like those found in AVs [6]. On the other hand, model-free methods seek to find the best solution to a specific problem, without taking into account the vehicle's dynamics. By a trial and error process, these methods learn a certain behaviour, which is not usually the safest nor the most comfortable, but that identifies high-level behaviours. This is why we propose a hybrid approach in this work that mixes the best of the DRL and classical control approaches. While a model-free DRL algorithm takes care of choosing the optimal high-level actions, some other classic modules are in charge of getting safe and comfortable movements defined for the actions, including the vehicle's dynamics.

This work develops a hybrid hierarchical DM architecture for urban scenarios. The adoption of hierarchical systems is widely acknowledged in the academic community [7]; however, unlike hierarchical DRL [8] that presents practical implementation concerns, our proposal uses a classical control system for generating vehicle movement signals, which significantly improves driving smoothness and safety. This allows to develop a realistic approach for driving in various complex urban scenarios, transcending the typical focus on singular use cases found in existing literature based on RL. The key contributions of the proposal are summarized as follows:

- **Contribution 1:** This paper proposes a hybrid methodology that integrates multiple components: preprocessing of map information, high-level DM facilitated by the DRL module, and low-level control signals managed by a classic controller. Our approach not only solves individual complex urban scenarios but also handles concatenated scenarios. This contribution is an extension of the work previously published in the conference IV 2023 [9].
- **Contribution 2:** In this work, a novel low-level controller is developed. This includes a Linear-quadratic regulator (LQR) controller for trajectory tracking and a Model Predictive Control (MPC) controller for manoeuvre execution. The online integration of these two controllers results in a hybrid low-level control module that allows the execution of high-level actions in a comfortable and safe manner.
- **Contribution 3:** This study also presents a RL framework developed within the Car Learning to Act (CARLA) simulator [10] to evaluate complete vehicle navigation with dynamics. Unique to this framework is the incorporation of evaluation metrics that extend beyond mere success rates to include the smoothness and comfort of the agent's trajectory.

## 2. Related Works

The DM module in AD essentially acts as the operational ‘brain’ of the vehicle. This module plays a pivotal role in ensuring both safe navigation and operational efficiency. Many studies in the literature concentrate on specific DM tasks [9,11]; however, our research encompasses the entire AD stack. In this section, we review some representative AD architectures in the literature, particularly in the context of DM and DRL.

### 2.1. Transformer-based Reinforcement Learning

The use of transformer-based algorithms has been gaining relevance in recent years in the context of AD [12]. With various works following this approach [13,14]. One of the most relevant and representative frameworks in this line employs a Scene-Rep Transformer to enhance RL DM capabilities [15]. This method allows the system to function independently of scenario-specific implementations, covering a range of scenarios with a singular approach. The selected state includes the past trajectories of the vehicles within the scenario and the future potential centrelines for each vehicle. The actions proposed in this work are the longitudinal velocity of the ego vehicle and a lane change signal, executed by the SUMO simulator. Experiments are carried out in both hyper-realistic and interactive driving environments. This work addresses three distinct urban scenarios independently but fails to offer a framework capable of navigating through these scenarios in a concatenated manner.

### 2.2. Attention-based Deep Reinforcement Learning

Another trend is attention-based architectures [16–18]. Specifically, the authors of [18] propose an attention-based driving policy combined with an LSTM network for the estimation of the value function. The aim is to manage unprotected intersections, employing DRL. This proposal places a greater emphasis on realistic implementation within an AD architecture. They utilize past trajectories and discrete route information as the state input and use a target velocity for the action space. A PID controller is responsible for generating the throttle and brake signals from this target velocity. In these scenarios, trajectory tracking is managed by an external controller, which implies that steer-related actions are not generated by the DRL agent. Lane changing is not considered, thus decisions are limited to adjusting the longitudinal velocity. The study demonstrates good results in terms of success rate primarily focusing on various intersection scenarios. However, velocity profiles, indicate the presence of abrupt control commands, leading to significant fluctuations in the target velocities.

### 2.3. Combining Machine Learning and Rule-based Algorithms

Some other works propose a hybrid strategy that combines the advantages of rule-based and learning-based methods for DM and control, aiming to mitigate their limitations [19,20]. The authors in [21] present a practical implementation, with emphasis on trajectory generation. This work opts for a simple but effective implementation. The authors propose a low-dimensional state space, with low-level controllers in charge of generating the driving commands. The graphical representation of this work shows smooth control signals, which are easily followed by a model with dynamics. However, the work lacks diversity of scenarios, focusing mainly on a single environment in simulation.

### 2.4. Tactical Behaviour Planning

Other works focus on solving various scenarios using a tactical behaviour planning [22,23]. The authors of [24] propose high-level DM for AD. They formulate the problem as a continuous POMDP, combining the advantages of two State of the Art (SOTA) solvers Monte Carlo Value Iteration and Successive Approximations of the Reachable Space under Optimal Policies. The authors also focus on high-level actions concerning linear velocity, specifically addressing throttle and brake commands, to which the vehicle’s dynamic model responds. The operational behaviour of the proposed framework demonstrates how

the vehicle follows the control commands. However, in the presented results, there is a notable lack of comfort metrics, with only velocity being considered. This approach shows effective behaviour across various scenarios, but the utilization of these algorithms presents challenges such as high computational costs and scalability issues.

**Table 1.** Comparison between the Deep Reinforcement Learning Architectures.

Architecture	Transformer-based	Attention + LSTM	Decision-Control	Tactical Behaviour	Ours
Ref	[15]	[18]	[21]	[24]	-
State Dimensionality	High	High	Low	Low	Low
Preprocessing	Transformer	Attention	-	-	Map
Action	Low-level	Low-level	High-level	High-level	High-level
Control Signal	Sharp	Sharp	Smooth	Smooth	Smooth
Multiple Scenario	✓	✓	×	✓	✓
Concatenated Scenario	×	×	×	✓	✓
Computational Cost	High	High	High	Low	Low
Scalability	✓	✓	×	×	✓
Real Implementation	×	✓	×	×	✓

## 2.5. Discussion

The comparative analysis presented in Table 1 delineates the distinctive attributes and efficiencies of the previously introduced DRL approaches, including Transformer-based, Attention-based, Decision-Control, Tactical Behaviour, next to our proposed methodology. This discussion highlights the distinct features and evaluates the overall effectiveness and feasibility of these architectures in the AD context regarding the following parameters.

- State Dimensionality and Preprocessing:** The architectures employing Transformer and Attention mechanisms are characterized by handling high state dimensionalities, utilizing sophisticated preprocessing techniques to manage complex input data. Conversely, the Decision-Control, Tactical Behaviour, and our approach, with an emphasis on low state dimensionalities, leverage simpler preprocessing methods such as map data, aiming for computational efficiency and reduced complexity in data handling.
- Action and Control Signal:** Transformer-based and Attention-based provide low-level control commands, often resulting in sharper vehicle control. In contrast, our methodology, along with Decision-Control and Tactical Behaviour, opt for high-level actions that yield smoother control signals, promoting more naturalistic and comfortable driving behaviours.
- Scenario Handling:** The capacity to adapt to multiple, including concatenated, scenarios shows the versatility of DRL models in AD. Our approach, similar to Transformer-based and Attention-based models, supports a broad spectrum of driving situations, crucial for developing adaptable and dynamic AD systems.
- Computational Cost and Scalability:** In terms of computational cost, both our proposal and Tactical Behaviour have lower costs and higher efficiency. Moreover, our model, together with the Transformer-based and the Attention-based, show scalability, being easily transferable from one environment to another.
- Real Implementation:** Among the reviewed architectures, only our approach and the Attention-based model have been validated in real-world settings, showcasing their reliability and applicability beyond simulated environments.

In conclusion, key to our methodology is the generation of smooth control signals, which are evaluated against comfort metrics to ensure a comfortable and safe driving experience. Unlike many existing systems that struggle with abrupt or jerky movements, our system prioritizes the smoothness of manoeuvres, making it more aligned with human

driving behaviours. Our system stands out by handling concatenated driving scenarios, showing its flexibility and ability to adapt to different driving situations. Another advantage of our approach is its operation within a low-dimensional state space, which, coupled with its low computational cost, makes it an efficient and scalable solution for AD.

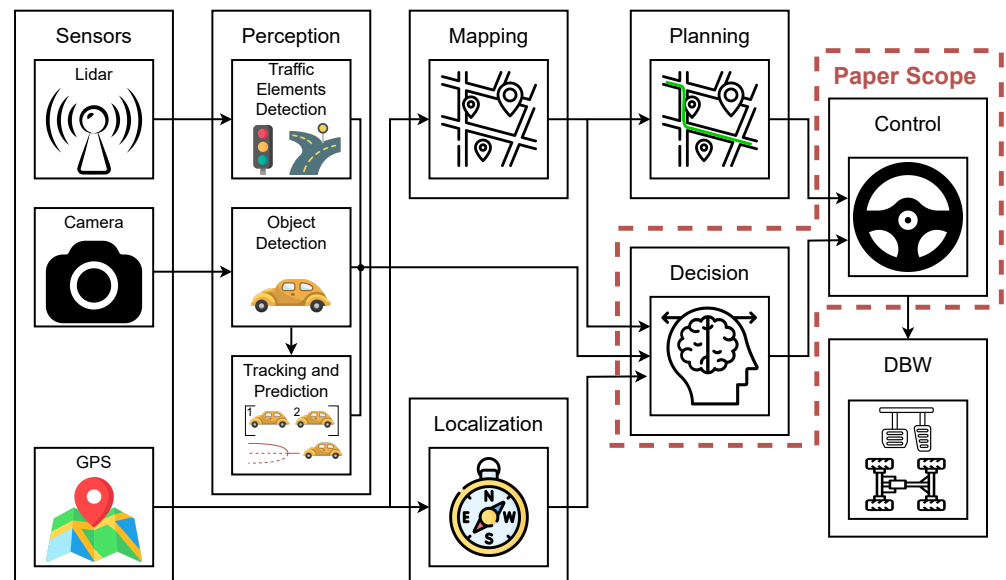
### 3. Background

This section delves into the foundational concepts and methodologies underpinning DM in AD. First, we provide an overview of the AD landscape. Moving forward, we introduce the mathematical framework used for our DM implementation. Finally, we present the algorithm used in our proposal.

#### 3.1. Autonomous Driving

Current AD SOTA identifies two principal software architecture categories: End-to-End and modular. End-to-End architectures are seen as "black-box" models where a single Neural Network (NN) manages the entire driving task directly from raw sensor data. This could potentially eliminate errors as intermediate representations are optimized. However, these models suffer from a lack of interpretability and present challenges in their implementation within real systems [25,26]. Conversely, modular architectures, often referred to as "glass models", decompose the driving task into separate modules, independently programmed or trained. Such architectures enhance interpretability, facilitate knowledge transfer, and allow parallel development, making them a standard in industrial research. However, they carry the risk of error propagation through the intermediate stages, potentially leading to suboptimal performance. The integration of these AD stacks is frequently mentioned in literature as a common approach for real-world applications [27,28].

The hybrid DM architecture proposed in this work is integrated into the AD stack of the Robesafe group [29]. Figure 1 illustrates the modular approach adopted by the group which consists of the following modules: perception, localization, mapping, planning, decision and control; and identifies the main modules developed in this paper (outlined with red dashed lines).



**Figure 1.** Autonomous Driving Stack modular pipeline.

This work is focused on the decision module, which is a critical component in the architecture of AVs. It serves as the brain of the vehicle, processing information from various sensors and subsystems to make informed decisions that ensure safe and efficient navigation. This module's primary responsibility is to interpret the environment, predict the actions of other road users, and determine the best course of action in real time, making



it indispensable for the successful operation of self-driving vehicles. There are several approaches to develop the decision module in AD [30]:

- **Rule-Based Systems:** Use a predefined set of rules to guide the vehicle's decisions.
- **Finite State Machines:** Model the DM process as a series of states and transitions.
- **Behaviour Trees:** This approach structures the DM process in a tree-like hierarchy, allowing for modular and scalable systems.
- **Machine Learning and Deep Learning:** These techniques, where DL is a subset of ML, enable vehicles to learn from data and make decisions based on features extracted from data.
- **Reinforcement Learning:** is a subset of ML, characterized by its focus on training an agent through experiments to interact effectively with its environment.

Among the various approaches, RL stands out as a promising method for tackling the challenges of DM in uncertain and complex environments, making it an attractive option for the future of AD technology.

### 3.2. Partially Observable Markov Decision Processes

Partially Observable Markov Decision Processes (POMDPs) are a mathematical framework for the RL implementation. At its core, a POMDP extends the Markov Decision Process (MDP) framework by incorporating elements of uncertainty in the observation of the system's state [31]. A POMDP can be formally defined by the tuple  $(S, A, T, R, \Omega, O)$ , where:

- $S$  is a finite set of states, representing the possible configurations of the environment.
- $A$  is a finite set of actions available to the decision-maker or agent.
- $T : S \times A \times S \rightarrow [0, 1]$  is the state transition probability function and  $T(s, a, s')$  represents the probability of transitioning to state  $s'$  from state  $s$  after taking action  $a$ .
- $R : S \times A \rightarrow \mathbb{R}$  is the reward function, associating a numerical reward (or cost) with each action taken in a given state.
- $\Omega$  is a finite set of observations that the agent can perceive.
- $O : S \times A \times \Omega \rightarrow [0, 1]$  is the observation function,  $O(a, s', o)$  defines the probability of observing  $o$  after taking action  $a$  and ending up in state  $s'$ .

Due to partial observability, the agent cannot directly access the true state of the environment. Instead, it maintains a belief state, a probability distribution over the set of possible states, representing its degree of certainty about the environment's actual state. The agent updates this belief state based on its past actions and observations.

### 3.3. Deep Reinforcement Learning

In the context of MDPs the state is fully observable and the Markov property implies that the future state is dependent only on the current state and action. However, in POMDPs the state  $s$  of the system is not fully observable, so the agent receives observations  $o$  that provide partial information about the state. This implies that while the underlying process is Markovian (the next state depends only on the current state and the action taken), the agent does not have full visibility of the state. This is the case of our DM, which takes into account the current belief state to make decisions. Due to this, POMDPs require a different approach compared to standard MDP for determining this belief state and learning value functions or policies. In DRL, the agent effectively determines the belief state and learns both the value function and the policy without the need for explicit feature engineering, leveraging the capabilities of DNNs. In this work, we implement an approach where DNNs are used to learn the value function, policy, and internal state of the agent modelled as a POMDPs in charge of the DM of our AD stack.

With numerous DRL algorithms available, based in our experience, we have selected the Trust Region Policy Optimization (TRPO) [32] algorithm to be integrated within our proposal. TRPO optimizes policy parameters while ensuring monotonic improvement in policy performance. The key idea in TRPO is to take the largest possible step in the

policy space without causing a significant deviation in the behaviour of the policy. This is achieved by optimizing a surrogate objective function subject to a trust region constraint. To ensure that the new policy is not too far from the old policy, a constraint based on the Kullback-Leibler divergence is used:

$$\text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s), \pi_{\theta}(\cdot|s)] \leq \delta \quad (1)$$

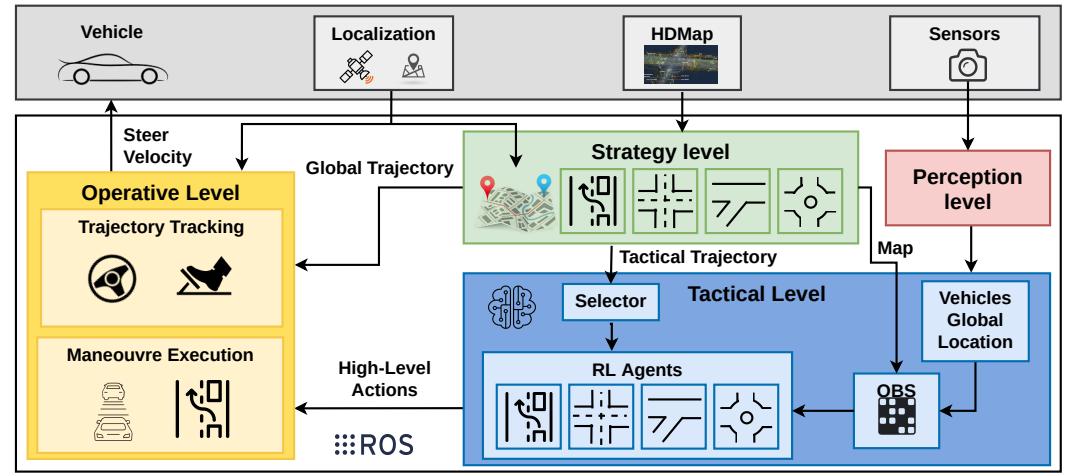
where  $\delta$  is a small positive number that defines the size of the trust region. The loss function in TRPO is defined as follows:

$$L(\theta) = \mathbb{E}_t \left[ \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \quad (2)$$

Here,  $\pi_{\theta}(a_t|s_t)$  represents the probability under the new policy parameters  $\theta$  of taking action  $a_t$  given state  $s_t$ ,  $\pi_{\theta_{\text{old}}}(a_t|s_t)$  is the probability under the old policy parameters, and  $\hat{A}_t$  is an estimator of the advantage function at time  $t$ .

#### 4. Methodology

This section provides an overview of the proposed hybrid DM architecture, focusing on its general structure and delving into each module of the AD stack, highlighting how these modules interact. As elaborated in the previous section, the current SOTA showcases various research directions, including the application of DL-based methods, the utilization of traditional algorithms, and the formulation of hybrid approaches. Our architecture has been inspired by and developed in line with these hybrid proposals. In this way, this work tackles the integration of different modules in an AD stack, encompassing high-level DM and local manoeuvre control within a hybrid architecture. Our system is uniquely designed to interact with both simulated and real-world environments. It has three primary inputs: vehicle location, HD map information, and sensor data.



**Figure 2.** The proposed hybrid architecture. The strategy level defines a tactical trajectory with the map information and the ego vehicle location. The tactical level executes high-level actions in correlation with the perception ground truth. The operative level combines the trajectory and the actions, calculating the driving commands.

The architecture of our system is structured into four levels, as illustrated in Figure 2. The perception level is responsible for processing sensor data. The decision module of an AD stack is typically divided into three tasks: global, local, and behavioural planning. The global planning is executed by the **strategy level**, where a tactical trajectory is defined based on the HD map information. This level lays the foundation for navigation and routing. Behavioural planning is carried out by the **tactical level**, corresponding to our high-level DM module. Here, high-level decisions are made, guiding the vehicle's behaviour across various driving scenarios. Local planning and low-level control are undertaken by the

Manoeuvre Execution module and the Trajectory Tracking module, respectively. These modules constitute the **operative level**, translating high-level decisions into actionable control commands and managing the vehicle's movements and interactions with its environment. This work contributes mainly to the tactical (DM module) and operative (local planning and control module) levels, with a small contribution at the strategy level (global planning module).

Overall, this approach allows for a clear division of functionalities within the AD stack, facilitating effective and efficient AD operations. In the following sections, we will describe the levels of the proposed architecture.

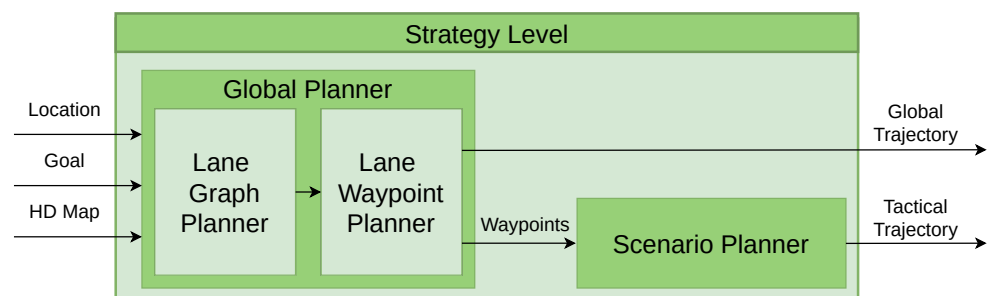
#### 4.1. Strategy Level

Within the strategy level, two modules are found: the global planner and the scenario planner. The global planner is tasked with defining the global trajectory, which generates the route the vehicle has to follow. In contrast, the scenario planner generates the tactical trajectory, containing the locations of different driving scenarios on the map. The architecture of the strategy level is presented in Figure 3.

The strategy level is based on prior research presented in [33], where the global planner module was developed. For this work, we have expanded upon their work by developing the scenario planner.

The global planner receives the ego-vehicle's position and goal position and calculates the optimal route between them as a list of waypoints centred in the lanes using the Dijkstra algorithm [33]. A waypoint is a structured object representing a 3D point with location (x, y, z), rotation (pitch, yaw, roll), and additional topological information from the HD map, including road and lane ID, lane width, lane markings, and the speed limit of the road. This module comprises two planners. The first planner, known as the Lane Graph Planner (LGP), establishes a topological (road-lane) route. Following this, the Lane Waypoint Planner (LWP) takes charge of generating waypoints. The map information encapsulated within the waypoints is used by the scenario planner to create the tactical trajectory. This tactical trajectory delineates the start and end points of various driving scenarios along the route. It serves as a crucial input for a high-level selector which, by considering this trajectory and the vehicle's current location, determines whether the vehicle is within a specific scenario (e.g. roundabout, merge, etc.).

In summary, this level uses three inputs: the current location of the vehicle, comprehensive map data, and a destination point. From these inputs, the system delineates two critical trajectories. The first, known as the global trajectory, is followed by the low-level control. The second trajectory, referred to as the tactical trajectory, is composed of scenario-specific waypoints, strategically positioned within the map.



**Figure 3.** Strategy level. The global planner calculates the global trajectory while the scenario planner generates the tactical trajectory.

#### 4.2. Tactical Level

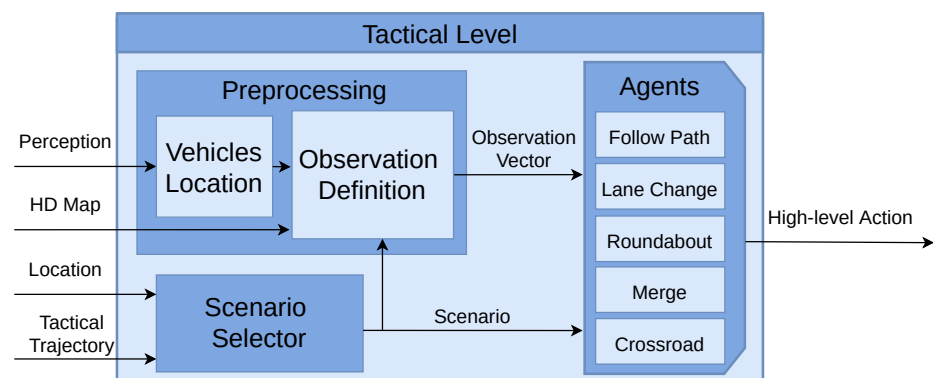
The tactical level is responsible for processing and evaluating information received from other levels to make decisions. The inputs for this level include the locations of different scenarios, which are provided in the tactical trajectory, HD map data, the vehicle's actual



location, and the location and velocities of adversarial vehicles. The output of the tactical level is a high-level action, which essentially dictates the vehicle's immediate behaviour. These decisions are stopping, continuing to drive, and executing lane changes. which are then executed by the operative level. An overview of the tactical level is illustrated in Figure 4.

The tactical level encompasses three modules: Scenario Selector, Perception Data Preprocessing, and Behaviour Modules, commonly referred to as Agents.

- **Scenario Selector:** This module is in charge of selecting the agent to be executed. In the tactical trajectory, the locations where each use case starts and ends are stored. The "Follow Path" agent is activated by default. When the vehicle reaches one of these locations, the selector activates the corresponding agent. Once the end of the use case is reached, the "Follow Path" agent is activated again.
- **Preprocessing:** Another task is the preprocessing of perception data, which involves transforming global locations and velocities of surrounding vehicles into an observation vector. This process starts by obtaining the global location of each vehicle, which is then mapped to a specific waypoint on the HD map. Subsequently, each waypoint is associated with a particular lane and road. This information, coupled with the current scenario as determined by the Scenario Selector, forms the basis for generating distinct observation vectors. Importantly, these vectors are scenario-specific, varying according to the different driving situations encountered.
- **Agents:** In the proposed architecture, five distinct behaviours (use cases) can be executed. By default, the 'Follow Path' behaviour is selected, where the operative level follows the global trajectory while maintaining a safe distance from the leading vehicle, and no active decisions are made. Upon the Scenario Selector choosing a specific scenario, one of the following agents is activated (Lane Change, Roundabout, Merge, Crossroad). These agents then take actions (drive, stop, turn left, turn right) based on the corresponding observation vector.



**Figure 4.** Tactical level. Perception data is processed in conjunction with the HD map to formulate the observation vector. The system selects a specific scenario based on the tactical trajectory and the current location. A decision module is selected, which is responsible of executing decisions.

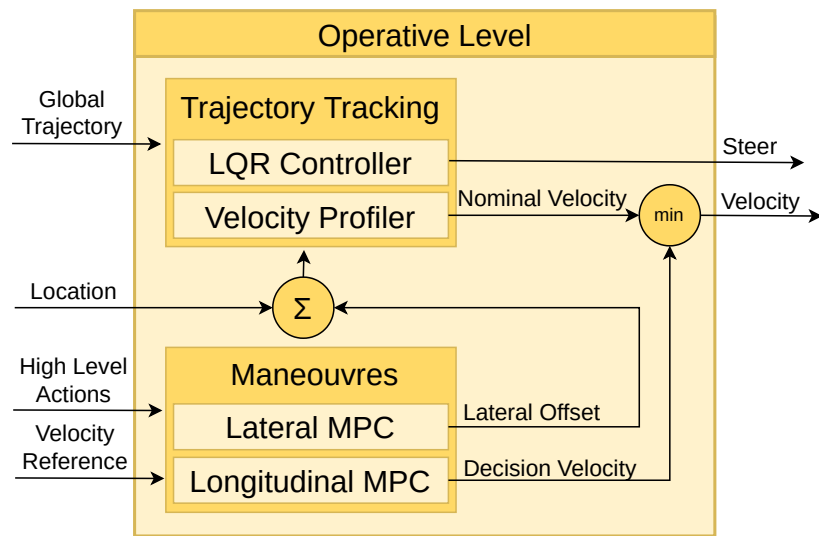
#### 4.3. Operative Level

The operative level is responsible for two primary tasks: following the global trajectory generated by the strategy level (Trajectory Tracking module), and executing high-level actions determined by the tactical level (Manoeuvre Execution module). To accomplish the first task, an LQR controller with delay compensation is employed. This allows easy adjustment of parameters through cost functions and real-time execution at medium driving velocities. More information can be found in [34]. The second task is executed using MPC controllers to incorporate lateral and longitudinal constraints, ensuring smooth movements during manoeuvres. This section introduces the architecture of the operative level, as

illustrated in Figure 5. The diagram highlights the interaction among the two controllers within this level. A deeper explanation was published by the authors in [35].

In "Follow Path" behaviour, the vehicle follows the nominal commands in a smooth trajectory set by the Trajectory Tracking module. When an action is selected by the Tactical Level, the Manoeuvre Execution module modifies the nominal control signals in a smooth way. This ensures that the desired action is executed smoothly and comfortably.

- **Trajectory Tracking:** Utilizing the provided waypoints, a smooth trajectory is computed using the LQR controller. At each simulation time step, the lateral and orientation errors are calculated to generate a steering command aimed at minimizing these errors. Besides, a velocity command is derived based on the curvature radius of the trajectory. These steering and velocity commands constitute the nominal commands, enabling the vehicle to follow the predefined path accurately.
- **Manoeuvres:** In the operative level, manoeuvres modify the nominal commands based on the tactical level's requirements, covering three primary tasks. Firstly, when a preceding vehicle is detected, the MPC controller adjusts the ego vehicle's velocity to adapt it to the ahead vehicle velocity keeping a safe distance. Secondly, if a stop action is demanded by an agent, the decision velocity decreases smoothly, being the resulting command the minimum between this constrained velocity and the nominal velocity. Lastly, in the case of a lane change request (left or right), the MPC controller generates a lateral offset to modify the vehicle's location, and the LQR controller generates a smooth steering signal to facilitate the lane change.



**Figure 5.** Operative level. The global trajectory is meticulously followed using a Linear Quadratic Regulator controller. Additionally, manoeuvres are executed with the aid of a Model Predictive Control system.

## 5. Experiments

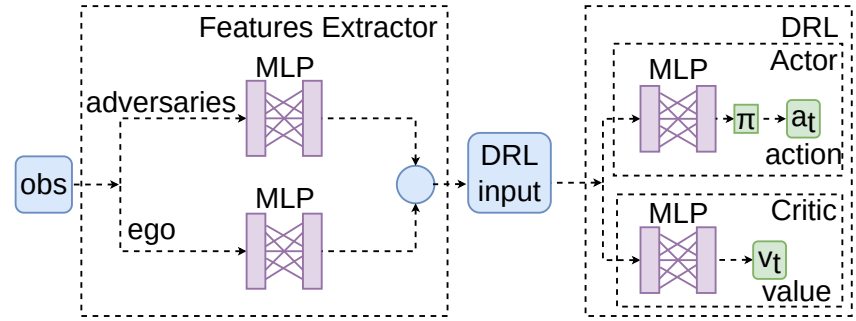
Urban driving environments encompass a variety of scenarios. In this work, we identify and explore four key scenarios common in many cities: crossroads, merges, roundabouts, and lane changes. We formulate these scenarios using POMDPs, treating each scenario independently. This method allows for a segmented understanding of the DM process, breaking it down into distinct tasks. The vehicle under control, referred to as the "ego vehicle," is defined as an agent. This agent gathers data from the environment in the form of observations and executes actions based on a defined policy. This policy updates through the training process, which utilizes the reward function.

The TRPO algorithm in this study is implemented using the SB3 library [36]. An agent is trained for each use case over one million time-steps, ensuring convergence. The NN

architecture is divided into two main components: a features extractor module and the DRL algorithm.

- **Features Extractor Module.** In line with insights from our previous research, this work incorporates a feature extraction module, which has proven to enhance the convergence of training [37], comprising a dense Multi-Layer Perceptron (MLP) to process observations from the environment. Information about both adversarial and ego vehicles is separately processed through the feature extractor. The outputs are then concatenated into a single vector, serving as the input for the DRL algorithms.
- **Deep Reinforcement Learning Algorithms.** An actor-critic framework is adopted. Within this framework, one MLP functions as the actor, determining the actions to take, while a separate MLP serves as the critic, evaluating the action's value.

All MLP present two hidden layers, with each layer comprising 128 neurons, and utilize the *tanh* activation function. The input layer's dimension is based on the number of elements in the observation vector. The dimension of the output layer corresponds to the number of possible actions. A representation of the framework is depicted in Figure 6.



**Figure 6.** A representation of the algorithm configuration. Features from both adversaries and the ego vehicle are extracted separately. The concatenated extracted information is then input into the actor-critic structure of the DRL algorithm.

### 5.1. POMDP formulation

The POMDP formulation for the four use cases is characterized by low-dimensional representations. However, there are differences in the number of vehicles and types of actions depending on the scenario.

#### 5.1.1. State

The state of a vehicle is defined by its distance to a relevant point (shown as  $d_i$  in Figure 7), its longitudinal velocity, and its driving intention:  $s_i = (d_i, v_i, i_i)$ , these values are normalized between  $[0, 1]$  and the driving intention of the adversarial vehicles are described by  $i \in [0, 2]$ . In the lane change scenario, a vehicle may have three intentions: change left ( $i = 1$ ), keep driving in its lane ( $i = 0$ ), or change right ( $i = 2$ ). In the crossroad scenario, these intentions are related to the trajectory to be followed: turn left ( $i = 1$ ), continue straight ( $i = 0$ ), or turn right ( $i = 2$ ). Regarding the merge scenario, the vehicles may yield ( $i = 0$ ) or take the way ( $i = 1$ ). Finally, in the roundabout scenarios, the intentions are to leave before ( $i = 0$ ) or after ( $i = 1$ ) the ego vehicle exit. We define the state of the environment as the collection of the individual states of the adversarial vehicles:  $s = (s_1, s_2, \dots, s_n)$ , being  $n$  the number of adversaries within the scenario.

#### 5.1.2. Observation

The observation matrix is defined by the nearest vehicles to the ego vehicle as shown in Figure 7. The observation space is defined by two key components: the relative normalized distances to surrounding vehicles, for the lane change scenario, and to the intersection point, for the rest of the scenarios, and the normalized velocities of these vehicles. This

approach ensures a comprehensive and scaled representation of the vehicle dynamics and spatial relationships in the system.

### 5.1.3. Action

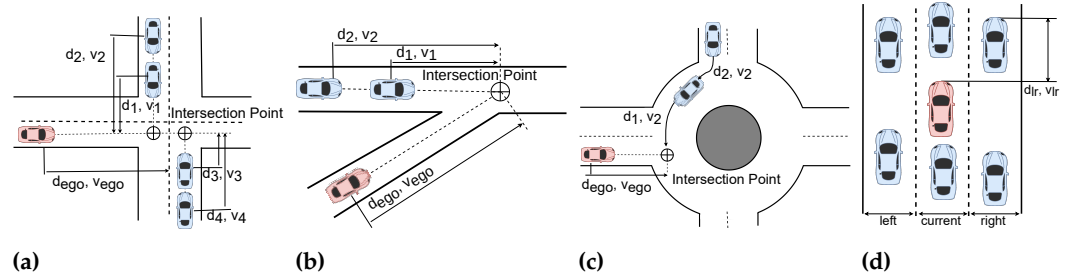
The agent only controls the high-level decisions while the velocity is automatically controlled by the operative level. We propose a discrete set of actions: change to the left lane, continue in the current lane, and change to the right lane for the lane change scenario. Alternatively, for the rest of the scenarios, the possible actions are "drive" and "stop".

### 5.1.4. Reward

We aim to drive at a desired velocity without colliding with adversarial vehicles. A positive reward is given when the vehicle reaches the end of the road and a negative reward is given when it collides. We propose a small cumulative reward based on its longitudinal velocity, to ensure that the vehicle tries to drive as fast as possible. The values of each component of the reward function have been chosen after testing different approaches. The reward function is defined as follows.

- Reward based on the velocity:  $k_v * v_{ego}$ ;
- Reward for reaching the end of the road:  $+1$ ;
- Penalty for collisions:  $-2$ ;

where  $k_v = 1 \times 10^{-3}$ . Under this setup, the episode reward ranges between  $[-2, 1.1]$ , due to the small values of the constants for velocity and right lane adherence. This range allows us to measure the vehicle's performance based on the average reward per episode.



**Figure 7.** Definition of the observation matrix for each scenario. The ego vehicle (red) and the adversaries (blue) are described for each use case. (a) Intersection. (b) Merge. (c) Roundabout. (d) Lane Change.

### 5.2. Evaluation Metrics

In order to comprehensively evaluate the performance of an RL agent, a thorough analysis is conducted focusing on safety, comfort, and efficiency metrics. Our evaluation is divided in two parts: a quantitative analysis and a qualitative analysis. Firstly, we examine various numerical metrics, each of which offers insights into different aspects of an agent performance:

- **Success Rate (%)**: This metric indicates the frequency of succeed episodes performed by the agent during simulation, providing a direct measure of safety.
- **Average of 95th Percentile of Jerk (per episode, in  $m/s^3$ )**: Jerk is the rate of acceleration changes. This metric reflects the smoothness of the driving, relating to passenger comfort.
- **Average of Maximum Jerk (per episode, in  $m/s^3$ )**: This metric measures the highest jerk experienced.
- **Average of 95th Percentile of Acceleration (per episode, in  $m/s^2$ )**: This metric provides insight into how aggressively the vehicle accelerates, impacting both comfort and efficiency.
- **Average Time of Episode Completion (in seconds)**: It measures the duration taken to complete an episode, indicating the efficiency of the agent.

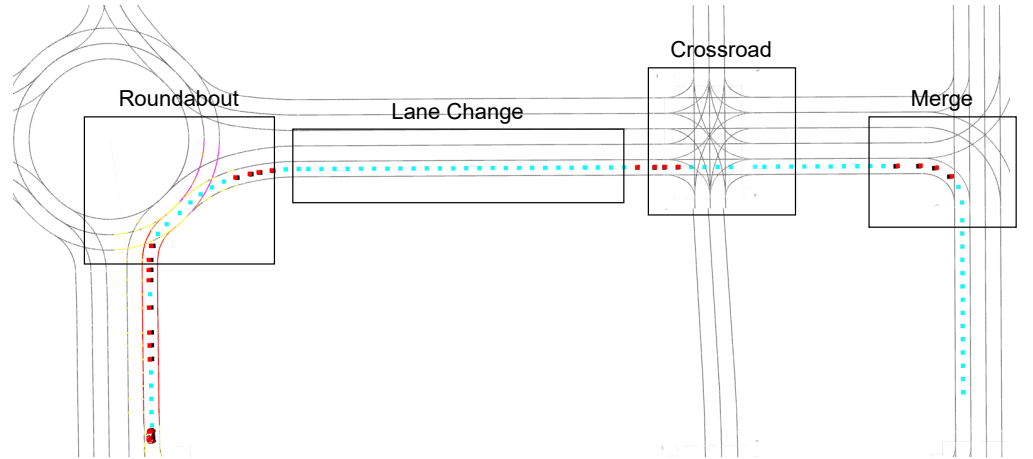
- **Average Speed (in m/s):** This metric assesses the agent's ability to maintain a consistent and efficient speed throughout the episode.

In the following sections, to evaluate our proposal, we compare it against the CARLA Autopilot, which is commanded using a PID controller. This operates under the management of the Traffic Manager (TM) module, being adversarial vehicles aware of its pose and actions. In this way, the TM generates the trajectories for all vehicles (including the ego vehicle) avoiding collisions.

### 5.3. Concatenated use cases scenario

In our setup, the perception module directly gets ground truth information from the simulator, bypassing the need for real-world sensor data processing. The scenario is set in the Town03 map, selected for its diverse use cases and high level of realism. We develop a scenario wherein the ego vehicle navigates through diverse urban environments under varying driving conditions. To achieve this, we ensure a diversity of routes and behaviours, as well as sufficient complexity in terms of traffic density and vehicle dynamics. We outline two steps to define a scenario. Initially, an ideal spot is identified, where a sufficient number of vehicles can navigate. Subsequently, we define the vehicles' routes and behaviours. This is accomplished using the TM integrated in the PythonAPI, which enables us to randomly designate the vehicles' trajectories. These vehicles can reach maximum speeds of 5 to 15 m/s (18 to 54 km/h) and their intentions matches the ones introduced in Section 5.1. However, realistic simulators often encounter issues with computing times. To mitigate this problem, we employ synchronous simulation, where each simulation step is completed before the next begins. This ensures all tasks for a given step, like calculations and decisions, are fully processed within a fixed frame rate. Additionally, we ensure efficient traffic flow generation by spawning and destroying vehicles at critical points, thereby avoiding the presence of vehicles outside the scope of the scenario.

The ego vehicle begins its route and approaches a roundabout, where it must safely enter. Upon exiting, it continues onto a two-lane road where vehicles in the right lane are moving slowly, needing an overtaking manoeuvre. Subsequently, it encounters a high traffic crossroad where it must identify a gap to cross. The final challenge involves merging right to complete the route. The complete scenario is depicted in Figure 8.



**Figure 8.** Concatenated Scenario: The vehicle initially navigates through a roundabout, subsequently approaching a two-lane road populated with slow-moving vehicles. This is followed by a crossroad and, ultimately, a merge intersection. The blue dots represent the path points to be followed, while the red points signify the tactical trajectory's use case indicators.



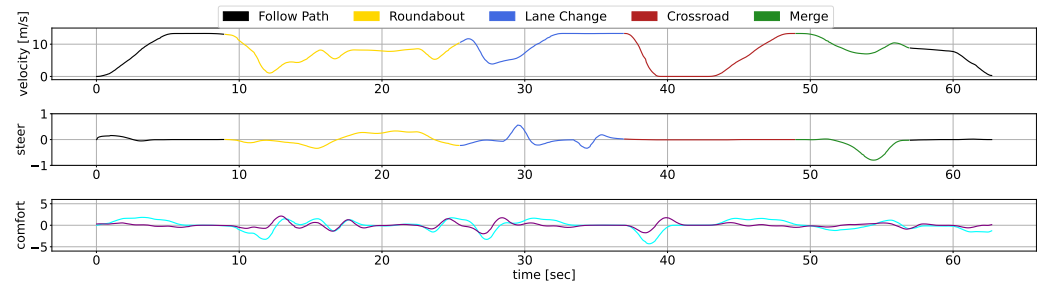
To measure the effectiveness and efficiency of our approach, we conducted a comparative analysis against the CARLA Autopilot. The findings of this comparison are detailed in Table 2. While the Autopilot system boasts a flawless success rate of 100%, this outcome is significantly influenced by its privileged access to the simulation's internal information. This insider advantage allows the Autopilot to navigate without the uncertainties that typically tackle autonomous systems. Conversely, our proposed system achieved impressive success rates of 95.76%. These figures underline our system's robustness and its adeptness at handling dynamic driving scenarios with limited information. A deeper dive into the jerk dynamics reveals our system's superior smoothness, with the 95th percentile of jerk per episode markedly lower than the Autopilot. Specifically, our system recorded jerk metrics of 4.37, compared to the Autopilot's 8.63 m/s<sup>3</sup>. This reduced jerk indicates a smoother driving, enhancing passenger comfort and safety. Furthermore, our system outperformed the Autopilot in terms of manoeuvre completion time and average speed, evidencing its efficiency. Our system completed the manoeuvres in significantly less time (76.85 seconds) than the Autopilot (140.23 seconds). Moreover, our system maintained higher average speeds, showcasing its ability to navigate the environment not only more quickly but also more smoothly.

**Table 2.** Performance metrics for the concatenated scenarios. Success rate percentage, jerk dynamics, acceleration, time to complete manoeuvres, and average speed are evaluated to assess the efficiency, smoothness, and safety of the proposed autonomous driving stack.

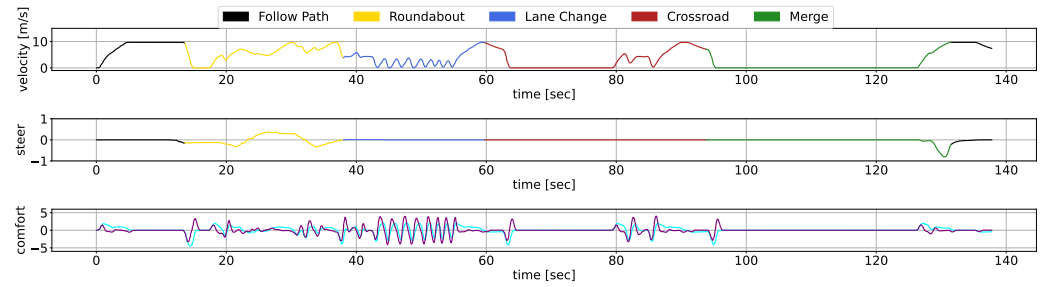
Metric	Agent	Concatenated Scenario
Success Rate [%] ↑	Ours Autopilot	95.76 <b>100</b>
95th Percentile of Jerk (per episode, in m/s <sup>3</sup> ) ↓	Ours Autopilot	<b>4.37</b> 8.63
Maximum Jerk (per episode, in m/s <sup>3</sup> ) ↓	Ours Autopilot	<b>6.20</b> 11.94
95th Percentile of Acceleration (per episode, in m/s <sup>2</sup> ) ↓	Ours Autopilot	3.33 <b>2.98</b>
Time (sec) ↓	Ours Autopilot	<b>76.85</b> 140.23
Speed (in m/s) ↑	Ours Autopilot	<b>7.60</b> 2.75

We showcase the temporal evolution of vehicle signals throughout the concatenated scenario, examining velocity, acceleration, jerk, and steering. The diagrams use different colours to represent the separate use cases, with Figure 9 showcasing our AD stack, and Figure 10 depicting the CARLA Autopilot's performance.

The black lines indicate periods outside specific scenarios, where both methodologies begin similarly. Upon nearing the roundabout, our approach efficiently merges, unlike the Autopilot, which stops, unable to navigate through small gaps. Subsequently, both methodologies trail a leading vehicle; however, our approach ensures smoother transitions during the ACC manoeuvre. Notably, when the preceding vehicle significantly reduces speed, our AD stack adeptly overtakes, whereas the autopilot remains behind, leading to jerk signal spikes due to alternating acceleration and braking. Similar patterns are observed in the crossroad and merge scenarios, where our system exhibits rapid, smooth actions in contrast to the Autopilot's erratic behaviour. Overall, the graphical analysis underscores our methodology's superior velocity and smoothness, attributable to the integration of DRL modules for scenario complexity management and classical low-level control for ensuring safety and comfortability.



**Figure 9.** Our AD stack temporal response within the scenario. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.



**Figure 10.** CARLA Autopilot temporal response within the scenario. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.

## 6. Conclusions and Future Works

We have developed an innovative hybrid DM architecture for AD systems, integrating classical control techniques with DRL. This hybrid approach combines the reliability of traditional control methods with the adaptive capabilities of DRL to changing scenarios. The proposed architecture is currently integrated within an AD stack, but its design is sufficiently flexible to be adapted to other architectures. The proposed architecture has surpassed traditional RL approaches by effectively deploying DRL algorithms in a hyperrealistic simulation, successfully solving various concatenated urban scenarios. Despite these advances, our proposal has some limitations, particularly in adapting to new situations and scenarios. This challenge is related to both the algorithms employed and the scenario generation process.

The focus of this work proposal lies in a classical implementation of DRL, as the aim is to integrate it into a complete AD architecture. However, there are several lines of research within RL that can improve the presented architecture and solve the limitations found. This includes Transfer Learning, where previously trained models can be used to enhance our approach; end-to-end methods, which allow the system to adapt more easily to new scenarios by extracting the state vector directly from sensor data; and Inverse Reinforcement Learning, which can learn a policy based on expert knowledge, resulting in more naturalistic behavior. Regarding simulation, countless possibilities exist, where we identify lines of research, such as scaling our proposal by incorporating more diverse traffic situations or creating a framework for the Automatic Generation of Scenarios where the adversarial vehicles present a human-like behaviour.

**Funding:** This work has been supported by the Spanish PID2021-126623OB-I00 project, funded by MICIN/AEI and FEDER, TED2021-130131A-I00, PDC2022-133470-I00 projects from MICIN/AEI and the European Union NextGenerationEU/PRTR, PLEC2023-010343 project (INARTRANS 4.0) from MCIN/AEI/10.13039/501100011033, and ELLIS Unit Madrid funded by Autonomous Community of Madrid.

## References

1. Rosenzweig, J.; Bartl, M. A review and analysis of literature on autonomous driving. *E-Journal Making-of Innovation* **2015**, pp. 1–57.
2. Deichmann, J. *Autonomous Driving's Future: Convenient and Connected*; McKinsey, 2023.
3. Wansley, M. The End of Accidents. *UC Davis L. Rev.* **2021**, *55*, 269.
4. Hubmann, C.; Becker, M.; Althoff, D.; Lenz, D.; Stiller, C. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In Proceedings of the 2017 IEEE intelligent vehicles symposium (IV). IEEE, 2017, pp. 1671–1678.
5. Wang, X.; Wang, S.; Liang, X.; Zhao, D.; Huang, J.; Xu, X.; Dai, B.; Miao, Q. Deep reinforcement learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems* **2022**.
6. Moerland, T.M.; Broekens, J.; Jonker, C.M. Model-based Reinforcement Learning: A Survey. *CoRR* **2020**, *abs/2006.16712*, [2006.16712].
7. Dinneweth, J.; Boubezoul, A.; Mandiau, R.; Espié, S. Multi-agent reinforcement learning for autonomous vehicles: A survey. *Autonomous Intelligent Systems* **2022**, *2*, 27.
8. Sun, Q.; Zhang, L.; Yu, H.; Zhang, W.; Mei, Y.; Xiong, H. Hierarchical reinforcement learning for dynamic autonomous vehicle navigation at intelligent intersections. In Proceedings of the Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 4852–4861.
9. Gutiérrez-Moreno, R.; Barea, R.; López-Guillén, E.; Arango, F.; Abdeslam, N.; Bergasa, L.M. Hybrid Decision Making for Autonomous Driving in Complex Urban Scenarios. In Proceedings of the 2023 IEEE Intelligent Vehicles Symposium (IV), 2023.
10. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In Proceedings of the Proceedings of the 1st Annual Conference on Robot Learning; Levine, S.; Vanhoucke, V.; Goldberg, K., Eds. PMLR, 13–15 Nov 2017, Vol. 78, *Proceedings of Machine Learning Research*, pp. 1–16.
11. Tram, T.; Batkovic, I.; Ali, M.; Sjöberg, J. Learning When to Drive in Intersections by Combining Reinforcement Learning and Model Predictive Control. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2019, pp. 3263–3268. <https://doi.org/10.1109/ITSC.2019.8916922>.
12. Agarwal, P.; Rahman, A.A.; St-Charles, P.L.; Prince, S.J.; Kahou, S.E. Transformers in reinforcement learning: a survey. *arXiv preprint arXiv:2307.05979* **2023**.
13. Huang, Z.; Liu, H.; Wu, J.; Huang, W.; Lv, C. Learning interaction-aware motion prediction model for decision-making in autonomous driving. In Proceedings of the 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2023, pp. 4820–4826.
14. Fu, J.; Shen, Y.; Jian, Z.; Chen, S.; Xin, J.; Zheng, N. InteractionNet: Joint Planning and Prediction for Autonomous Driving with Transformers. In Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2023, pp. 9332–9339.
15. Liu, H.; Huang, Z.; Mo, X.; Lv, C. Augmenting Reinforcement Learning with Transformer-based Scene Representation Learning for Decision-making of Autonomous Driving, 2023, [arXiv:cs.LG/2208.12263].
16. Chen, Y.; Dong, C.; Palanisamy, P.; Mudalige, P.; Muelling, K.; Dolan, J.M. Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, 2019, pp. 0–0.
17. Cao, D.; Zhao, J.; Hu, W.; Ding, F.; Huang, Q.; Chen, Z. Attention enabled multi-agent DRL for decentralized volt-VAR control of active distribution system using PV inverters and SVCs. *IEEE transactions on sustainable energy* **2021**, *12*, 1582–1592.
18. Seong, H.; Jung, C.; Lee, S.; Shim, D.H. Learning to Drive at Unsignalized Intersections using Attention-based Deep Reinforcement Learning. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), 2021, pp. 559–566. <https://doi.org/10.1109/ITSC48978.2021.9564720>.
19. Avilés, H.; Negrete, M.; Reyes, A.; Machucho, R.; Rivera, K.; de-la Garza, G.; Petrilli, A. Autonomous Behavior Selection For Self-driving Cars Using Probabilistic Logic Factored Markov Decision Processes. *Applied Artificial Intelligence* **2024**, *38*, 2304942.
20. Lu, J.; Alcan, G.; Kyrki, V. Integrating Expert Guidance for Efficient Learning of Safe Overtaking in Autonomous Driving Using Deep Reinforcement Learning. *arXiv preprint arXiv:2308.09456* **2023**.
21. Aksjonov, A.; Kyrki, V. A Safety-Critical Decision Making and Control Framework Combining Machine Learning and Rule-based Algorithms, 2022, [arXiv:cs.AI/2201.12819].
22. S Kassem, N.; F Saad, S.; I Elshaaer, Y. Behavior Planning for Autonomous Driving: Methodologies, Applications, and Future Orientation **2023**.
23. Gong, X.; Wang, B.; Liang, S. Collision-Free Cooperative Motion Planning and Decision-Making for Connected and Automated Vehicles at Unsignalized Intersections. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2024**.
24. Sefati, M.; Chandiramani, J.; Kreiskoether, K.; Kampker, A.; Baldi, S. Towards tactical behaviour planning under uncertainties for automated vehicles in urban scenarios. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1–7. <https://doi.org/10.1109/ITSC.2017.8317819>.
25. O'Kelly, M.; Sinha, A.; Namkoong, H.; Duchi, J.C.; Tedrake, R. Scalable End-to-End Autonomous Vehicle Testing via Rare-event Simulation. *CoRR* **2018**, *abs/1811.00145*, [1811.00145].

26. Ma, Y.; Wang, Z.; Yang, H.; Yang, L. Artificial intelligence applications in the development of autonomous vehicles: a survey. *IEEE/CAA Journal of Automatica Sinica* **2020**, *7*, 315–329. <https://doi.org/10.1109/JAS.2020.1003021>. 615
27. Kessler, T.; Bernhard, J.; Buechel, M.; Esterle, K.; Hart, P.; Malovetz, D.; Truong Le, M.; Diehl, F.; Brunner, T.; Knoll, A. Bridging the Gap between Open Source Software and Vehicle Hardware for Autonomous Driving. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 1612–1619. <https://doi.org/10.1109/IVS.2019.8813784>. 616
28. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **2020**, *8*, 58443–58469. <https://doi.org/10.1109/ACCESS.2020.2983149>. 617
29. Gómez-Huélamo, C.; Diaz-Diaz, A.; Araluce, J.; Ortiz, M.E.; Gutiérrez, R.; Arango, F.; Llamazares, Á.; Bergasa, L.M. How to build and validate a safe and reliable Autonomous Driving stack? A ROS based software modular architecture baseline. In Proceedings of the 2022 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2022, pp. 1282–1289. 618
30. Liu, Q.; Li, X.; Yuan, S.; Li, Z. Decision-Making Technology for Autonomous Vehicles Learning-Based Methods, Applications and Future Outlook, 2021, [[arXiv:cs.RO/2107.01110](https://arxiv.org/abs/cs/2010.01110)]. 619
31. Shani, G.; Pineau, J.; Kaplow, R. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems* **2013**, *27*, 1–51. 620
32. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust Region Policy Optimization. In Proceedings of the Proceedings of the 32nd International Conference on Machine Learning; Bach, F.; Blei, D., Eds., Lille, France, 07–09 Jul 2015; Vol. 37, *Proceedings of Machine Learning Research*, pp. 1889–1897. 621
33. Diaz-Diaz, A.; Ocaña, M.; Llamazares, A.; Gómez-Huélamo, C.; Revenga, P.; Bergasa, L.M. HD maps: Exploiting OpenDRIVE potential for Path Planning and Map Monitoring. In Proceedings of the 2022 IEEE Intelligent Vehicles Symposium (IV), 2022. 622
34. Gutiérrez, R.; López-Guillén, E.; Bergasa, L.M.; Barea, R.; Pérez, ; Gómez-Huélamo, C.; Arango, F.; del Egidio, J.; López-Fernández, J. A Waypoint Tracking Controller for Autonomous Road Vehicles Using ROS Framework. *Sensors* **2020**, *20*. <https://doi.org/10.3390/s20144062>. 623
35. Abdeslam, N.; Gutiérrez-Moreno, R.; López-Guillén, E.; Barea, R.; Montiel-Marín, S.; Bergasa, L.M. Hybrid MPC and Spline-based Controller for Lane Change Maneuvers in Autonomous Vehicles. In Proceedings of the 2023 IEEE International Conference on Intelligent Transportation Systems (ITSC), 2023, pp. 1–6. 624
36. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* **2021**, *22*, 1–8. 625
37. Gutiérrez-Moreno, R.; Barea, R.; López-Guillén, E.; Araluce, J.; Bergasa, L.M. Reinforcement Learning-Based Autonomous Driving at Intersections in CARLA Simulator. *Sensors* **2022**, *22*. <https://doi.org/10.3390/s22218373>. 626