A Curriculum Approach to Bridge the Reality Gap in Autonomous Driving Decision-Making based on Deep Reinforcement Learning

Rodrigo Gutiérrez-Moreno[1], Rafael Barea[1], Elena López-Guillén[1], Felipe Arango[1] and Luis M. Bergasa[1]

***Abstract*—Decision-Making is a fundamental topic in the domain of Autonomous Driving where significant challenges must be tackled due to the variable behaviours of surrounding agents and the wide array of encountered scenarios. The primary aim of this work is to develop a hybrid Decision-Making architecture able to be validated on a real vehicle that marries the reliability of classical techniques with the adaptability of Deep Reinforcement Learning approaches. To address the crucial transition from simulated training environments to real-world applications, this research employs a Curriculum Learning approach, facilitated by the deployment of Digital Twins and Parallel Intelligence technologies, significantly narrowing the Reality Gap and enhancing the applicability of learned behaviours. The viability of this approach is evidenced through a Parallel Execution, wherein simulated and real-world tests are conducted simultaneously. Specifically, our approach consistently surpasses the performance benchmarks set by existing frameworks in the literature within SMARTS, achieving success rates over 91%. Additionally, it completes various scenarios in CARLA up to 50% faster than the Autopilot, demonstrating improved comfort and safety.**

***Index Terms*—A Curriculum Approach to Bridge the Reality Gap in Autonomous Driving Decision-Making based on Deep Reinforcement Learning.**

I. Introduction

The deployment of Autonomous Driving (AD) systems in urban environments requires the design of an advanced Decision Making (DM) framework capable of processing complex environmental inputs and implementing safe and optimal actions. In recent years, Reinforcement Learning (RL) has emerged as a cutting-edge approach for handling the inherent uncertainties and dynamic challenges found in these settings. However, applying RL training directly to real vehicles can be both expensive and unsafe [1]. Hence, safety considerations must not only focus on algorithmic aspects but also consider the costs of sensors and the potential damage to vehicles during the training process. To mitigate these issues, numerous strategies have shifted towards an initial experimental phase that utilizes high-fidelity simulations. These simulations replicate critical scenarios, identifying risky behaviors in training the DM system before progressing to real-world tests [2]. Within the field of AD, a notable disparity often arises between simulated environments and real conditions, commonly known as the Reality Gap (RG). Various approaches have been suggested to bridge this RG, primarily divided into three categories: 1) Sim2real, which involves transferring knowledge from simulations to the real world, where DM systems are trained in simulation and then refined on real tests. 2) Digital Twins (DT), where virtual representations of the real world help vehicles gain insights about their DT by synchronizing data from both the real and simulated environments in an offline manner. 3) Parallel Intelligence (PI) technology, which combines the advantages of both Sim2real and DT. In this last group the learned knowledge is applied to the real vehicle through DT with real-time interaction between the real and virtual worlds and online feedback through a Parallel Execution (PE) [2].

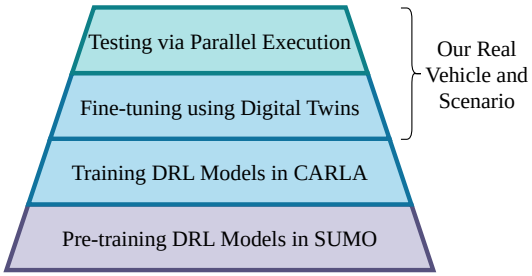


Fig. 1: Methodology for decision-Making design: 1) DRL agent pre-training in SUMO. 2) Training of the DRL model in CARLA. 3) Development and fine-tuning of the Digital Twins. 4) Parallel Execution testing in a Real-World scenario.

In this work, we propose a novel methodology for the practical implementation of a DM module in our AD stack, following a Curriculum Learning (CL) strategy and emphasizing four key steps, as depicted in Figure 1. The first step involves training the Deep Reinforcement Learning (DRL) agent in a lightweight simulator, such as SUMO [3], to develop an initial behavior model. In the second step, the trained model is transferred to a hyperrealistic simulation environment (CARLA [4]), performing a second training stage to refine the model. The third step is building a DT, which includes the dynamics of our ego vehicle and replicates the road layout of our real testing scenario. This DT serves as a virtual testing setup, allowing for the evaluation of our DM

*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.

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

in simulation in a safe way. Finally, in the fourth step, our DM approach is validated in a real-world setting through a Parallel Execution (PE), where synchronized experiments in both the simulated and physical environments are carried out in real-time. Interaction with adversarial vehicles is simulated following an Augmented Reality (AR) strategy, while the framework is evaluated in our real vehicle. This innovative approach narrows the gap between simulated training and real-world application allowing great flexibility in the design of use cases at a low cost and safety.

Therefore, this research presents a significant opportunity to develop a DRL-based system designed specifically for real-world applications. Building upon our prior work [5], we extend our study by introducing four distinct driving scenarios. Additionally, we conduct a deeper analysis of the DRL agents, comparing their performance to other State of the Art (SOTA) approaches. The results are also examined more thoroughly, with the inclusion of comfort metrics. We also increase the number of experiments in the parallel execution stage, providing more extensive validation. Our key contributions in this work are as follows:

- Methodology for DM design that reduces the gap between simulation and real-world application, consisting of a CL strategy for the DRL agent training and testing.
- Implementation of various SOTA DRL algorithms within the domain of AD, and a comparison against other SOTA methodologies within the standard framework Scalable Multi-Agent Reinforcement Learning Training School (SMARTS) [6].
- RL framework developed within the CARLA simulator to evaluate complete vehicle navigation with dynamics.
- Substantial advancement towards real-world RL applicability through the development of a DT based on our actual platform. We present a PE experiment with AR that effectively bridges the gap between simulated models and real implementation.

II. Related Works

As part of our review of DM methods for AD, we observed a range of approaches. While classical methods have been widely used, in this paper we shift our focus to learning-based techniques, which have recently gained significant attention [7]. We can find three different tendencies: 1) Statistical learning-based methods enable ADs to acquire human-like DM skills through extensive training data [8]. 2) Deep Learning (DL)-based methods are prominent for end-to-end approaches, utilizing raw sensor data for low-level control [9]. 3) RL-based methods aim to maximize returns through trial-and-error strategies. Among these approaches, RL stand out as a promising method for tackling the challenges of DM in the uncertain environments found in AD.

In this realm, DRL has emerged as a leading approach able to analyse environments and to infer the optimal decision for a given set of inputs, surpassing the capabilities inherent to RL. For DRL implementation, three fundamental elements requires consideration: the definition of the state space, the action space, and the reward function. Regarding state representation, traditional methods typically revolve around lower-dimensional features, such as: proximity to obstacles, lane positioning, or vehicular velocities [10]. These models exhibit robust behavior in complex situations, demonstrating strong resilience and adaptability. Other approaches include the adoption of higher-dimensional features, such as Bird-Eye-View imagery [11], image augmentation techniques [12], and the use of occupancy grids [13]. About the action space definition, some approaches propose the use of high-level actions, such as ”stop,” ”drive slow,” or ”drive fast” [14], and decisions like ”take way” or ”give way” [15]. Others, focus on lane changing with commands like ”change left,” ”idle,” and ”change right” [16]. Regarding the reward function definition, a positive reward is commonly given for successfully ending an episode, while a collision incurs a negative reward. In [17], a positive reward is linked to the velocity of the vehicle, providing an incentive to move. A different perspective, given by [18], assigns a negative reward based on the simulation time of each episode. All above approaches have been applied to specific scenarios, but these only provide partial results applicable to larger applications.

On the other hand, some works present complete AD implementations based on DRL. One of them employs a Scene-Rep Transformer to enhance RL DM capabilities [19]. The actions proposed in this work are the longitudinal velocity of the ego vehicle and a lane change signal, executed by the SUMO simulator. Other works place a greater emphasis on realistic implementation within an AD architecture. The authors of [20] propose an attention-based driving policy for managing unprotected intersections, employing DRL. A hybrid approach is proposed by the authors of [21], presenting a DM and control framework that capitalizes on the strengths of both rule-based and learning-based techniques, while mitigating their disadvantages. Although these proposals address various scenarios and move closer to realistic applications, they encounter challenges when it comes to extending their applicability beyond simulated environments. Consequently, most research on DRL-based methods has focused on simulated experiments.

However, the aim of this proposal is to work with real vehicles, where safety and cost play a pivotal role. In this way, the transition from the simulated world to the real world is very important. To overcome this RG, as we mentioned before, three different approaches are considered. Regarding Sim2Real techniques [22], CL is a training strategy that trains a machine learning model from easier data to harder data, which imitates the meaningful learning order in human curricula [23]. In [17], an automatic curriculum generation method is proposed, and [24] obtains a better-overtaking performance using a tree-stages CL methodology. Transfer Learning is another technique in which knowledge learned from a task is re-used to boost performance on a related task. [25] proves that transfer learning using simulated accident data leads to better generalization to more diverse scenarios. In [26] a transfer learning for semantic segmentation of off-road driving environments using a pre-trained segmentation network is performed. Another approach is the use of DT, which is a digital representation of a physical entity, which can simulate

the entire life cycle of the operating system and synchronize the mapping with the physical twin [27]. A transfer learning strategy to efficiently train a DRL policy in simulation and deploy it in a real-time vehicle application is shown in [28]. In [29], a DT environment model that can predict the transition dynamics of the physical driving scene is proposed to improve the data efficiency of RL, which often requires a large amount of agent-environment interactions during the training process. Recently, researcher have employed PI approaches to facilitate the transfer of knowledge from simulated environments to the real world. These approaches combine the benefits of Sim2real and DT in modeling complex systems, addressing challenges that individual methods encounter in handling the RG problem. Liu et al. [30] enhance the safety and reliability of intelligent vehicles by integrating virtual vehicles with diverse roles in complex physical scenarios. Wang et al. [31] introduces the foundational idea of parallel testing, utilizing a cyclic updating method to tackle the RG problem.

Despite these advancements, there remains a need for real-world implementations that integrate RL-based DM systems within a complete architecture, bridging the RG. Our approach offers a concrete pathway for developing a fully operational RL-based DM system and proves the aplicability of this methodology by conducting a test on our real vehicle.

III. Background

In this work, we implement the DM module using DRL within a Partially Observable Markov Decision Process (POMDP) framework. This section lays the foundation for our implementation by defining these concepts in detail.

A. POMDP Formulation

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. A POMDP can be formally defined by the tuple (S, A, T, R, Ω, 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.
- $T : S \times A \times S \to [0, 1]$ 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.
- $R : S \times A \to \mathbb{R}$ is the reward function, associating a numerical reward (or cost) with each action taken in a given state.
- Ω is a finite set of observations that the agent can perceive.
- $O : S \times A \times \Omega \to [0, 1]$ 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.

B. Deep Reinforcement Learning

Markov Decision Processes (MDPs) satisfy what is know as Markov property, which states that the future state of the system depends only on the current state and the action taken, not on the sequence of events that preceded it. This can be represented as:

$$P(s_{t+1}|s_t, s_{t-1}, \ldots, s_0, a_t) = P(s_{t+1}|s_t, a_t). \quad (1)$$

The implication of the Markov property is that the state encapsulates all relevant historical information from the past, making the system memory-less with respect to transition dynamics.

A policy π is a strategy that specifies the action a to be taken in each state s. It can be deterministic or stochastic. A deterministic policy has a definite action for each state, while a stochastic policy provides probabilities of choosing each action in each state. The return is a crucial concept used to assess the effectiveness of a policy. It represents the total accumulated reward an agent expects to gain over a trajectory. The return, denoted as G_t, from a time-step t is typically defined as the sum of discounted rewards that an agent receives in the future, which can be expressed as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (2)$$

where γ is the discount factor with $0 \leq \gamma < 1$.

Value functions are fundamental to evaluate the quality of states and actions under a specific policy. The state value function, denoted as $V^\pi(s)$, represents the expected return when starting in state s and following policy π thereafter. It is defined as:

$$V^\pi(s) \triangleq \mathbb{E}[G^\pi | s = s_0] \quad (3)$$

The state-action value function, or Q-function, denoted as $Q^\pi(s, a)$, represents the expected return starting from state s, taking action a, and thereafter following policy π. It is defined as:

$$Q^\pi(s, a) \triangleq \mathbb{E}[G^\pi | s = s_0, a = a_0] \quad (4)$$

The concept of training in RL involves teaching an agent to make optimal decisions through interaction with its environment to maximize cumulative rewards. The Bellman equation plays a crucial role in the training process. It provides a recursive relationship for value functions, which is key in computing them efficiently. The Bellman equation for the state value function is:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s')] \quad (5)$$

and for the state-action value function:

$$Q^\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s', a')] \quad (6)$$

The optimal policy π^* is the one that maximizes the value functions for all states or state-action pairs. The optimal state value function is defined as:

$$V^*(s) \triangleq V^{\pi^*}(s) = \max_{\pi}(s) V^{\pi}(s) = \max_{a} \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V^{\pi}(s') \right] \tag{7}$$

The optimal state-action value function is defined as:

$$Q^*(s,a) \triangleq Q^{\pi^*}(s) = \max_{\pi}(s) Q^{\pi}(s,a) = \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} Q^*(s', a') \right] \tag{8}$$

and the optimal policy is defined as:

$$\pi^*(s) = \arg\max_{a} Q^*(s,a) \tag{9}$$

These equations consider the idea that the value of a state (or state-action pair) is the immediate reward plus the discounted value of the successor state(s). Training in RL typically involves iteratively adjusting the policy based on the Bellman equation to improve the estimated value functions. This iterative process continues until the policy converges to the optimal policy π^*, which maximizes the expected return from any given state or state-action pair.

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. Therefore, the DM process 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.

Traditional RL methods typically rely on tabular representations or simple function approximations, supplemented with manually engineered features, to deal POMDPs. In contrast, DRL, 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 represent the value function, policy, and internal state of the agent. With numerous DRL algorithms available, our focus is narrowed to those which fit our goals and are compared in this work, aiming to provide comprehensive insights into their functionality. A common method of classifying DRL algorithms is into value-based and policy-based categories. We evaluate both approaches: Deep Q-Network (DQN), which is value-based, and Advantage Actor Critic (A2C), Trust Region Policy Optimization (TRPO), and Proximal Policy Optimization (PPO), which are policy-based.

IV. Curriculum Methodology

Urban driving environments encompass a variety of scenarios. We formulate these scenarios using POMDPs, treating each of them independently. This method allows for a segmented understanding of the DM process, breaking it down into distinct tasks. The vehicle under control in each scenario, is defined as an agent, being one agent for each scenario. Building on the concepts introduced in the previous section, an 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, using the reward function. The development of the training and testing of the agents is done in a curricular way, which is organized into four key phases: pre-training in SUMO; integration and training in CARLA; Digital Twins design and fine-tuning; and validation through a Parallel Execution.

A. Pre-training in SUMO

The initial phase uses SUMO for its fast simulation capabilities, allowing for training at a low computational cost. Although SUMO lacks vehicle dynamics, it provides a sufficient environment for the agent to learn basic behaviors and shape an initial policy. Here, the policy weights are randomly initialized, and each agent interacts with its scenario-specific environment to learn appropriate policy. Therefore, as depicted in Figure 2a, the agent's action, $a_{j,s}$, is taken based on the policy $\pi_{j,s}$, which depends on the observation $o_{j,s}$ and the learned weights $\theta_{j,s}$:

$$a_{j,s} = \pi_{j,s}(o_{j,s} \mid \theta_{j,s}) \tag{10}$$

The observation $o_{j,s}$ that is provided to the agent is derived from the action taken $a_{j,s}$ and the transition function $T_{j,s}$:

$$o_{j,s} = f_{j,s}(a_{j,s}, T_{j,s}) \tag{11}$$

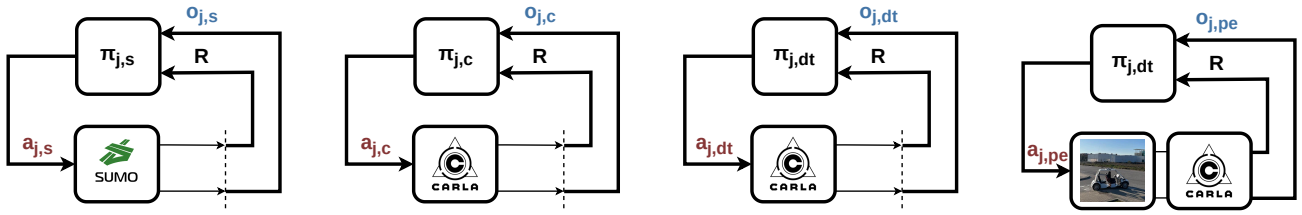
Here, j represents the driving scenario, while s indicates that this simulation phase is conducted in SUMO.

B. Training in CARLA

Once basic behaviors are learned, the agent undergoes a second training phase in CARLA, which provides a realistic simulation of vehicle dynamics. Training from scratch in this hyperrealistic simulator could be time-intensive and has risk of non-convergence, as discussed in our previous work [32]. In this second phase, we perform a second training, starting from the SUMO weights $\theta_{j,s}$ as prior information and subsequently re-training in CARLA to obtain a new policy $\pi_{j,c}$. As illustrated in Figure 2b, the agent's action $a_{j,c}$ is determined based on the policy $\pi_{j,c}$, which depends on the observation $o_{j,c}$ and the weights derived from the prior information from SUMO $\theta_{j,c}, \theta_{j,s}$.

$$a_{j,c} = \pi_{j,c}(o_{j,c} \mid \theta_{j,c}, \theta_{j,s}) \tag{12}$$

Again, the observation $o_{j,c}$ that is provided to the agent is influenced by the action taken $a_{j,c}$ and the transition function $T_{j,c}$:



(a) (b) (c) (d)

Fig. 2: A visual representation of curriculum methodology: (**a**) Training basic bheaviour without vehicles dynamics in SUMO. (**b**) Re-training in CARLA with vehicles dynamics. (**c**) Fine-tuning the DRL models for our real settings using Digital Twins. (**d**) Validating our approach through a Parallel Execution.

$$o_{j,c} = f_{j,c}(a_{j,c}, T_{j,c}) \tag{13}$$

Here, c indicates that the simulation is carried out by CARLA.

C. Fine-tuning using Digital Twin

As a preliminary step to real-world implementation, a DT of the vehicle and environment is created. We first obtain the vehicle's physical parameters and take measurements of our real test scenario to accurately replicate them in CARLA. Once these models are established, we start a new training phase to generate the DRL models for our real experiments. This training process begins with the previously acquired weights from CARLA, denoted as $\theta_{j,c}$, which serve as prior information. Subsequently, we retrain with the DT to obtain a new policy $\pi_{j,dt}$. As illustrated in Figure 2c, the agent's action $a_{j,dt}$ is determined by the policy $\pi_{j,dt}$, which depends on the observation $o_{j,dt}$ and the weights from both CARLA and the DT ($\theta_{j,dt}$ and $\theta_{j,c}$, respectively).

$$a_{j,dt} = \pi_{j,dt}(o_{j,dt} \mid \theta_{j,dt}, \theta_{j,c}) \tag{14}$$

Furthermore, the observation $o_{j,dt}$ provided to the agent is a function of the action taken $a_{j,dt}$ and the transition function $T_{j,dt}$:

$$o_{j,dt} = f_{j,dt}(a_{j,dt}, T_{j,dt}) \tag{15}$$

In these equations, dt denotes that the agent operates within the DT, reflecting the dynamics of our real platform.

D. Parallel Execution

The final phase of our methodology is a PE to validate the applicability of our architecture under a real world test. Here, the agent executes actions according to the previously learned policy $\pi_{j,dt}$, without a new training phase. The actions during this phase are produced as follows:

$$a_{j,pe} = a_{j,dt} = \pi_{j,dt}(o_{j,dt} \mid \theta_{j,dt}, \theta_{j,c}) \tag{16}$$

In this phase, the ego vehicle operates in the real world, interacting with its physical environment and responding to real dynamics. However, adversarial vehicles are represented within a simulated environment in CARLA, allowing for controlled testing scenarios without introducing real-world adversarial agents. This approach combines the real-world dynamics $T_{j,rw}$ of the ego vehicle with the virtual dynamics $T_{j,dt}$ of simulated adversaries, creating a mixed reality environment. The observations provided to the agent depend on both the simulation in CARLA and the dynamics in real world:

$$o_{j,pe} = f_{j,pe}(a_{j,pe}, T_{j,rw}, T_{j,dt}) \tag{17}$$

Here, pe denotes Parallel Execution, while rw refers to the Real-World.

V. Our Architecture

We carry out the curricular methodology using our AD stack, previously detailed in [33]. This approach comprises four distinct levels (Figure 3). While the perception level handles sensor data processing, it's not the focus of this work. This is why we directly retrieve ground truth perception from the simulator. The other three levels (strategy, tactical and operative) are part of our hybrid DM proposal.

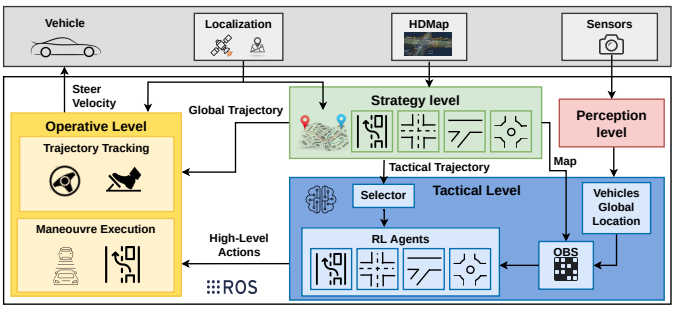


Fig. 3: The proposed hybrid DM architecture: Strategy, Tactical and Operative levels.

The strategy level [34] consists of two parts: a global planner that maps out the overall vehicle route and a scenario planner that identifies specific driving situations on the map.

The tactical level receives inputs like scenario locations, HD map data, adversarial vehicles' position and the ego vehicle's position to make key decisions about immediate actions by using a DRL model for each driving scenario.

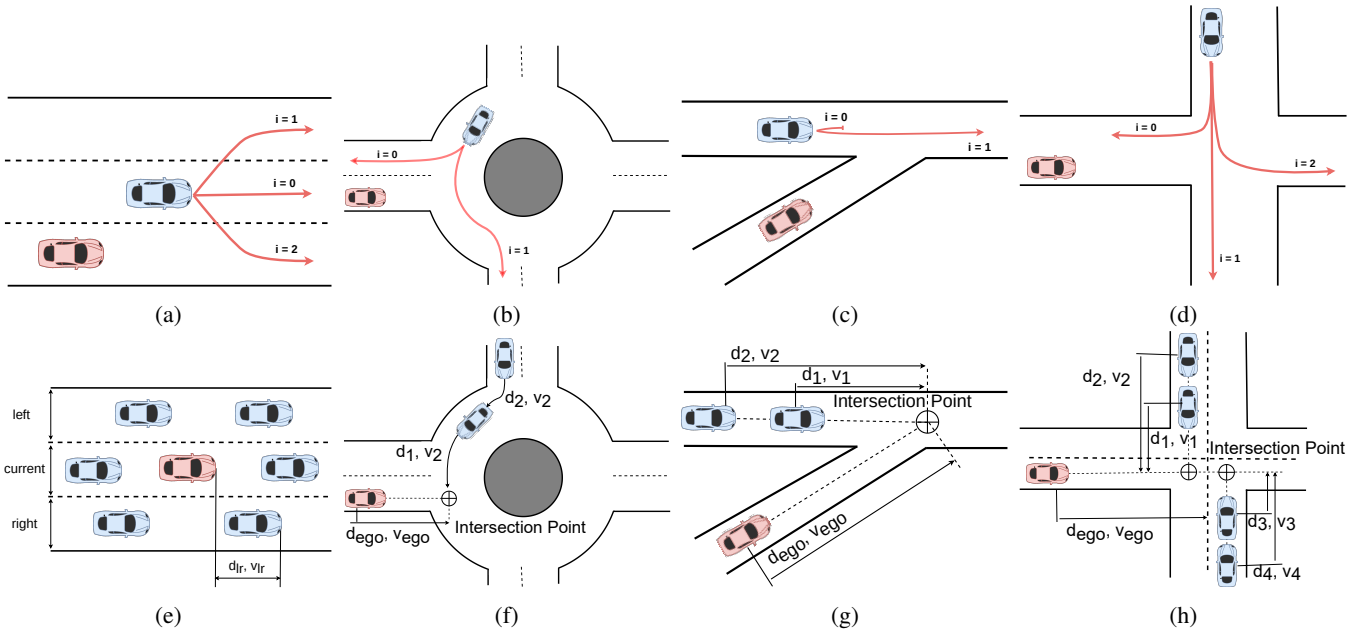


Fig. 4: The state and observations representations of the scenarios. (**a**) Lane Change state. (**b**) Roundabout state. (**c**) Merge state. (**d**) Crossroad state. (**e**) Lane Change observations. (**f**) Roundabout observations. (**g**) Merge observations. (**h**) Crossroad observations.

The above decisions are then performed at the operative level through two control systems: a Linear Quadratic Regulator [35] for trajectory tracking and a Model Predictive Control [36] for manoeuvers. Together, they ensure smooth adherence to the strategic plan (global planning) while enabling real-time responsiveness (local planning).

A. Deep Reinforcement Learning Architecture

This work focuses on the tactical level, involving various DRL agents, that share the same architecture as show in Figure 5, which is divided into two main components:

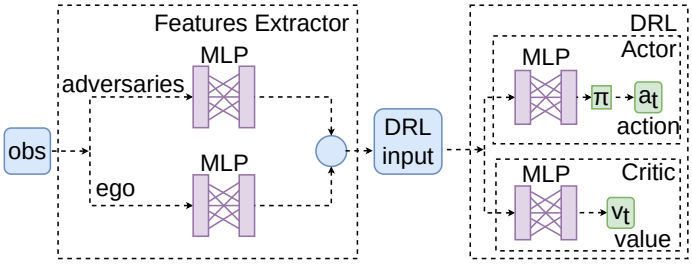


Fig. 5: The policy-based algorithms configuration.

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 processes observations from the environment. Information pertaining to both adversarial and ego vehicles is separately processed through the feature extractor and then concatenated into a single vector, serving as the input for the DRL algorithms.

DRL Algorithms: Our study employs two categories of algorithms: value-based (DQN) and policy-based (A2C, TRPO, and PPO). The value-based algorithm incorporates a single MLP for its operation, in contrast to the policy-based algorithms, which adopt an actor-critic framework. 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 observations. The dimension of the output layer corresponds to the number of possible actions.

B. POMDP Modeling for Urban Scenarios

We identify and explore four key scenarios common in many cities: different types of intersections (crossroads, merges, roundabouts) and lane change. We define each of these scenarios with an uniform POMDP formulation, characterized by the adoption of low-dimensional observation vectors and high-level actions. Specifically we define the scenarios as follows:

1) State Space: The state of a vehicle is defined by its distance to a relevant point d_i, its longitudinal velocity v_i, and its driving intention i_i: $s_i = [d_i, v_i, i_i] \in \mathbb{R} \times \mathbb{R} \times \{0, 1, 2\}$. For the lane change scenario, d_i is the distance to the ego vehicle, and each vehicle has three possible intentions: change left ($i = 1$), keep driving in its lane ($i = 0$), or change right ($i = 2$), as illustrated in Figure 4a. In intersection scenarios, d_i represents the distance to the intersection point, and driving intentions depend on the specific type of intersection. In a roundabout, the intention is predefined by the route to be

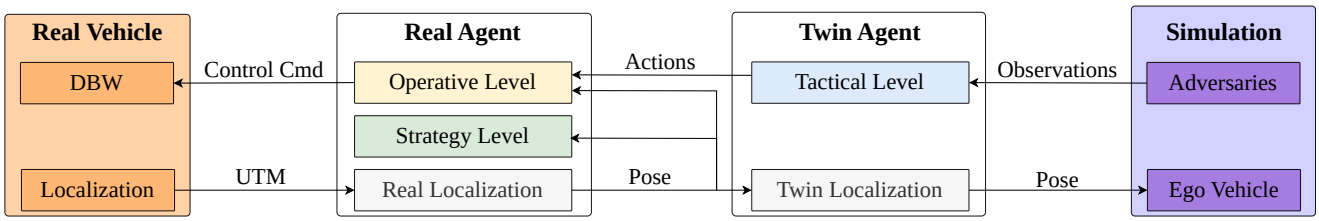


Fig. 6: The Real Agent processes GPS input and generates control signals, the DBW module controls the real vehicle, while the Twin Agent synchronizes the simulation. The DM coordinates actions with simulated observations.

followed, where vehicles exit at the first ($i = 0$) or second exit ($i = 1$), as shown in Figure 4b. In the merge scenario, adversarial vehicles are divided by their level of cooperation: vehicles that consistently yield ($i = 0$), while those that proceed without yielding ($i = 1$). These behaviors are illustrated in Figure 4c. Finally, for crossroads, intentions correspond to the intended route. Vehicles turning right ($i = 0$), those going straight ($i = 1$), and those turning left ($i = 2$). These intentions are illustrated in Figure 4d.

2) Observation Space: In our approach, vehicles are not able to know the intentions of surrounding vehicles. Therefore, the observation of a vehicle is defined by its distance to a relevant point d_i and its longitudinal velocity v_i: $o_i = [d_i, v_i] \in \mathbb{R}\times\mathbb{R}$. Specifically, in the lane change scenario, the observation matrix is defined by the nearest vehicles in the current and adjacent lanes of the ego vehicle. As shown in Figure 4e, we consider the information of six vehicles: three leading vehicles and three following vehicles. The observation is represented as: $\Omega = (d_{ll}, v_{ll}, d_{lc}, v_{lc}, d_{lr}, v_{lr}, d_{fl}, v_{fl}, d_{fc}, v_{fc}, d_{fr}, v_{fr})$. Regarding intersections, our approach does not rely on prior knowledge of the type, the observation contains the longitudinal velocity and position of the vehicles. Consequently, in the roundabout scenario, the observation vector is defined as: $\Omega = (d_e, v_e, d_1, v_1, d_2, v_2)$, where d_e and v_e refer to the ego vehicle's position and velocity, and d_1, v_1, d_2, v_2 correspond to the two nearest adversarial vehicles. The vector is structured such that the closest relevant vehicle always occupies the first position, as illustrated in Figure 4f. In the merge scenario, the observation vector is defined in the same way, as shown in Figure 4g. Finally, in the crossroad scenario, where more than two vehicles are considered, the observation vector is defined as: $\Omega = (d_e, v_e, d_1, v_1, \ldots, d_4, v_4)$, encompassing the observations of the ego vehicle and the two closest adversarial vehicles in each lane, which is represented in Figure 4h.

3) Action Space: Our DM system has four possible actions: drive, stop, change left and change right, so we propose a discrete set of actions for each scenario. In the lane change scenario, the action space is determined by three high-level actions: 'change left', 'continue straight' and 'change right'. These actions can be executed at anytime while the agent is driving in a road with more than one lane. The action space in this scenario is defined as follows: $a = (change\ left, drive, change\ right)$. Furthermore, the action space for navigating intersections consists of two high-level actions: 'stop' and 'drive'. These actions are strategically executed before entering the intersection and are designed to guide the vehicle on when to merge into the intersection and when to yield to other vehicles. The action space for this system is thus defined as follows: $a = (stop, drive)$.

4) Reward Function: The objective of an RL algorithm is to optimize the expected value of the discounted future reward. The purpose of the reward function, in these use cases, is to perform the rapid and safe navigation of the ego vehicle through an scenario, avoiding collisions with adversarial vehicles. Collisions, result in a negative reward, while successful navigation results in a positive reward. To further encourage the vehicle's forward progression, a small cumulative reward based on longitudinal velocity is proposed. Additionally, at the end of each episode, a small negative reward is assigned proportionally to its duration, where t represents the episode duration and t_{out} represents the timeout time. The reward function is defined by a reward based on the velocity, $k_v \cdot v_{ego}$; a reward for crossing the intersection, $+1$; a penalty for collisions, -2; and a penalty relative to the episode duration, $-0.2t/t_{out}$. Here, $k_v = 2 \times 10^{-3}$. Under this setup, the episode reward ranges between $[-2, 1.1]$.

C. Parallel Execution Implementation

To bridge the gap between simulation and real-world applications, we develop an agent capable of translating the vehicle's movements from the real environment into the simulation. This approach enables to apply decisions obtained from the simulated environment directly to a physical vehicle, thereby facilitating a seamless transition from virtual to real-world testing. The real vehicle is mirrored in the simulator, and the simulation data feeds the decision system. This behaviour is achieved through two principal agents (real and twin). The interface connecting these two agents with the simulation is depicted in Figure 6.

Real Agent: This agent processes input from a GNSS system to create a localization pose within the real scenario. The localization data is then fed into the operative level, which generates control commands, including target linear velocity and target steer angle, which are sent to the DBW module [38] at a frequency of 20 Hz. This module is responsible for translating target commands into electric signals to move the real vehicle at a frequency of 100 Hz. This is done using a PID controller for each target signal.

Twin Agent: The Twin Agent receives the vehicle's location data, provided by the real-world localization module, and poses the simulated vehicle in the same position but in the simulated environment. Meanwhile, the DM module processes the observations corresponding to the adversarial simulated vehicles and generates the corresponding actions, which are sent back to the Real Agent.

VI. EXPERIMENTS

This section presents the quantitative results of our study, highlighting the performance metrics achieved by our proposed methodology. Additionally, visual qualitative results are available for further exploration on our GitHub repository [1].

A. Results in SUMO

We have conducted a study in SMARTS scenarios using the SUMO simulator to compare the various algorithms proposed in this work. In this ablation study, we focus on evaluating the agent's learning of the first policy $\pi_{j,s}$, which involves taking high-level actions, while the control signals are generated by the simulator. The complete hybrid DM system is not evaluated in this phase. The agent's performance is evaluated using the success rate, which is a straightforward indicator of the agent's efficacy. Additionally, the average duration of an episode is calculated. The evaluation metrics are defined as follows:

- Success Rate (%): $success\ [\%] = \frac{n_s}{n_e} \times 100$
- Average Episode Time (s): $t_{avg}\ [s] = \frac{\sum t_e}{n_e}$

Here, n_e represents the number of evaluation episodes. n_s represents the number of success episodes, and t_avg represents the average simulation time of an episode.

1) Ablation Study: The 'Driving SMARTS 2022' benchmark, launched in the NeurIPS 2022 Driving SMARTS competition [39], offers a variety of scenarios to evaluate DRL proposals for AD. For our purposes, we have selectively focused on four scenarios, specifically targeting our urban environment needs. A representation of these scenarios is provided in Figure 7. In these scenarios, the traffic flow is consistently managed by the SMARTS simulator, with vehicles being generated at systematic intervals between one to three seconds and achieving maximum velocities of up to 15 m/s (54 km/h). This setup ensures a uniform testing environment across different driving situations, providing a controlled yet challenging context for evaluating the performance of our DRL proposal.

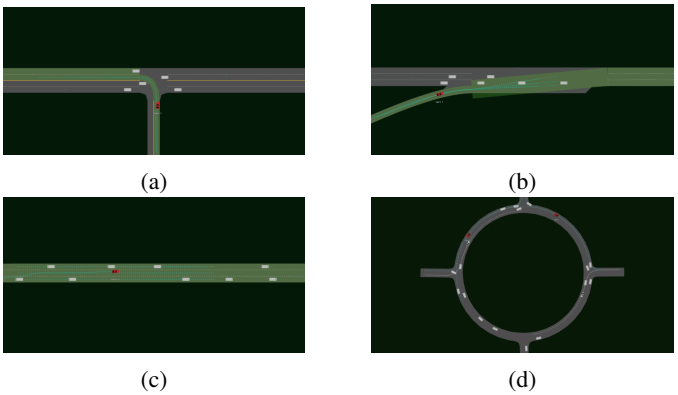


Fig. 7: The designed scenarios within the SMARTS framework. **(a)** Unprotected left turn. **(b)** Three lane merge. **(c)** Three lane road. **(d)** Roundabout.

[1]https://github.com/RobeSafe-UAH/curriculum_based_reinforcement_learning

Figure 8a illustrates the training process progression for the agents in the unprotected left turn scenario. While the convergence point for all agents is comparable, a substantial disparity is observed in the average mean reward, with the TRPO agent achieving a notably higher mean reward. The DQN agent emerged as the fastest. Notably, the TRPO agent distinguished itself by achieving the highest success rate at 99.2% with a competitive average time, as shown in Table I.

The progression of the training process in the three lane merge scenario is depicted in Figure 8b. The TRPO agent emerges as the top performer, with the PPO and DQN agents demonstrating performances that are also competitive. The outcomes detailed in Table I shows that the fastest agent is the DQN with an average time of 5.81 seconds, However, the TRPO agent achieved the highest performance, showing a 92.9% success rate.

As shown in Figure 8c, in the three lane road scenario the maximum mean reward per episode is set slightly over 1 by the TRPO agent, which suggests its success in identifying an optimal policy. As depicted in Table I, the TRPO agent achieves a remarkable success rate of 95.3%. Despite this, all four agents showed comparable average episode durations, being the A2C the quickest and the TRPO the slowest.

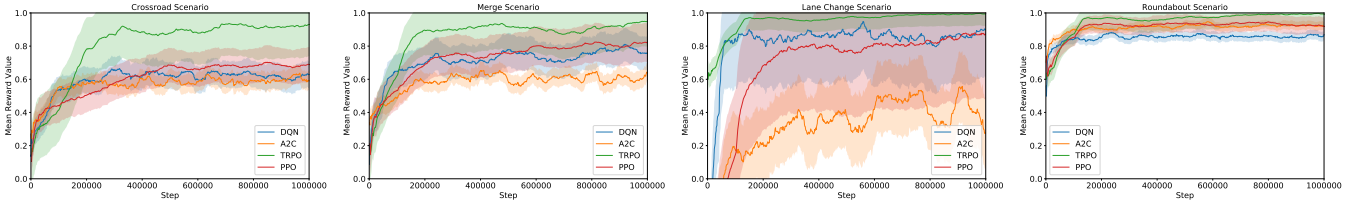
A comparative analysis of training rewards in the roundabout scenario is presented in Figure 8d. The TRPO agent shows superior performance, characterized by a higher mean reward. All agents achieve convergence, typically around 200k time-steps. During the testing phase, with the results presented in Table I, the TRPO agent outperformed others, achieving the highest success rate at 91.3%. Conversely, the DQN exhibited the lowest success rate.

2) SOTA Comparison: We conduct training and testing for our TRPO agent, which was identified as the top performer among the proposed DRL algorithms. We compare the performance of our proposal in the selected SMARTS scenario with two representative methods of the SOTA: [19] employs a Transformer-based scene representation alongside an actor-critic DRL approach. Moreover, [40] is based in three ingredients: expert demonstration, policy derivation, and DRL. The analysis is presented in Table I.

In the three lane merge scenario, our framework outshines with a success rate of **98.40%**, surpassing the 96.00% achieved by [19] and excelling in a domain where [40] provides no comparative data. Moreover, our framework's performance is further evidenced in the roundabout scenario, recording a success rate of **91.70%**. This performance substantially exceeds the 76.00% success rate by [19] and the 84.00% by [40]. Efficiency, measured through the average completion time, further distinguishes our framework. Notably, in the three lane merge scenario, it accomplishes the task in **21.9 seconds**, markedly quicker than the 28.60 seconds required by [19]. Although [40] achieves a marginally higher success rate in the unprotected left turn scenario, our framework maintains competitive success rates across all scenarios while consistently offering more efficient manoeuvrer execution.

TABLE I: The results of the ablation study and the comparison between our approach and the SOTA proposals in the SAMRTS scenarios. Success rate (sr) and average time (at) are presented.

Scenario	Ablation Study								SOTA Comparison					
	DQN		A2C		TRPO		PPO		Ours		[19]		[40]	
	sr	at	sr	at	sr	at	sr	at	sr	at	sr	at	sr	at
Unprotected left turn	88.5	**11.13**	94.6	11.76	**99.2**	13.87	97.3	12.81	95.3	**12.22**	94.0	12.50	**96.0**	14.26
Three lane merge	82.1	**5.81**	82.8	5.61	**92.9**	8.84	86.7	7.18	**98.4**	21.9	96.0	28.60	-	-
Three lane road	83.4	17.12	81.2	**16.32**	**95.3**	18.24	88.3	17.92	93.6	24.34	-	-	-	-
Roundabout	81.5	16.12	89.9	13.97	**91.3**	14.75	90.1	**12.45**	**91.7**	37.47	76.0	56.60	84.0	**36.62**



(a) (b) (c) (d)

Fig. 8: Evolution of the mean rewards during the training process for the DRL agents: DQN(blue), A2C(orange), TRPO(green) and PPO(red). **(a)** Unprotected left turn. **(b)** Three lane merge. **(c)** Three lane road. **(d)** Roundabout.

B. Results in CARLA

In this section, we evaluate the performance of our AD stack, which includes our hybrid DM architecture, within the CARLA simulator.

1) Urban Scenarios for Reinforcement Learning: The urban scenarios simulate realistic, uncontrolled traffic conditions, where vehicles spawn every 3 to 5 seconds and move at speeds of 5 to 15 m/s (18 to 54 km/h). For lane change scenarios, we use the *Town04* map in CARLA, which includes a 400-meter road with four lanes. The intersection scenarios are set in the *Town03* map. Here, a roundabout scenario uses a 30-meter radius roundabout; a merge scenario positions vehicles on a 60-meter lane perpendicular to the ego vehicle's path; and a crossroad scenario has two 50-meter roads intersecting, with adversarial vehicles generated on both sides of the intersection. A detailed analysis is conducted on safety, comfort, and efficiency using the following metrics: Success Rate (%), which measures the percentage of successful episodes and reflects safety; 95th Percentile of Jerk (m/s^3), indicating the smoothness of driving and passenger comfort; Maximum Jerk (m/s^3), which tracks the highest jerk experienced; 95th Percentile of Acceleration (m/s^2), representing the vehicle's acceleration and its impact on comfort and efficiency; and Episode Completion Time (seconds), which measures the time taken to complete an episode and serves as an indicator of efficiency.

To evaluate our proposal, we compare it against the CARLA Autopilot [4]. This operates under the management of the TM module, being adversarial vehicles aware of its pose and actions. In this way, the TM uses a PID controller to generate the trajectories for all vehicles (including the ego vehicle) avoiding collisions. Table III presents a comprehensive evaluation of our AD stack compared to the CARLA Autopilot across various driving scenarios: lane change, roundabout, merge, and crossroad. While the Autopilot achieves a success rate of 100% in all scenarios due to the centralized management carried out by the TM for all vehicles, and the access of the ego vehicle to the complete environmental data, our AD stack shows competitive performance with high success rates in all scenarios, demonstrating its robustness and effectiveness. The jerk dynamics significantly favour our AD stack, with substantially lower 95th percentile and maximum jerk values across all scenarios. This indicates a smoother and more comfortable ride compared to the CARLA Autopilot, which exhibits higher jerk values. Acceleration dynamics further underscore our system's efficiency, with our AD stack maintaining lower 95th percentile acceleration in most scenarios, signifying gentler and safer acceleration and deceleration patterns. Our system completes scenarios in significantly less time than the CARLA Autopilot. Additionally, our AD stack maintains higher average speeds across scenarios. To provide a qualitative assessment of our proposal's performance, we present the temporal response across the four concatenated scenarios, as illustrated in Figure 9.

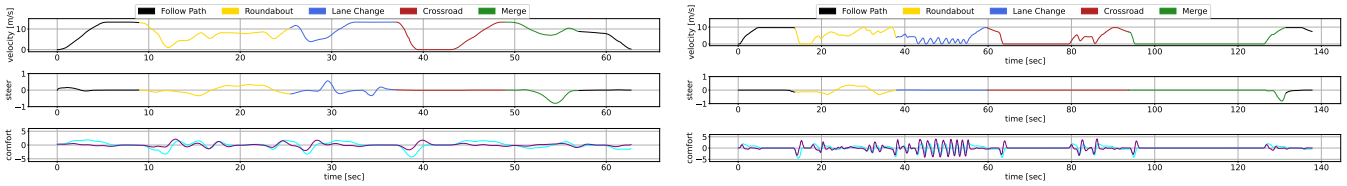
2) Digital Twins: We get the characteristics of our real vehicle and we take measures of the scenario to replicate a realistic configuration in CARLA obtaining our DT. In this work we focus on the merge scenario for the DT and the PE.

To create the University Campus DT, the process begins with acquiring the campus map from OpenStreetMap, which is imported into the RoadRunner tool and subsequently into Unreal, allowing the generation of the virtual environment within CARLA. While this map accurately represents the roads, it does not extend to environmental elements. Adversarial vehicles are generated on a lane perpendicular to the ego vehicle's lane. These vehicles are subsequently destroyed when they reach the end of the scenario. A depiction of the scenario is presented in Figure 10.

To build our ego vehicle's DT, we calculate the parameters of the model to ensure that the simulated vehicle responds as safely as the real one when subjected to the same control commands. The model is defined by a mass of 1030 kg, a

TABLE II: Comparison between our agent and Autopilot across different driving scenarios in CARLA. Metrics include success rate (sr), 95th and max jerk, 95th acceleration, time, and speed.

Metric	Lane Change		Roundabout		Merge		Crossroad	
	Ours	Autopilot	Ours	Autopilot	Ours	Autopilot	Ours	Autopilot
sr [%] ↑	91.20	**100**	95.10	**100**	96.40	**100**	87.90	**100**
95th Jerk (m/s^3) ↓	**1.58**	9.12	**1.73**	5.93	**2.67**	3.63	**1.83**	14.6
Max Jerk (m/s^3) ↓	**5.64**	13.56	**2.20**	12.16	**3.83**	9.98	**2.16**	22.8
95th Acceleration (m/s^2) ↓	**1.53**	3.65	**1.61**	2.67	2.53	**2.51**	**1.55**	3.88
Time (sec) ↓	**68.94**	128.56	**20.32**	30.23	**25.83**	34.16	**23.14**	38.84
Speed (in m/s) ↑	**9.05**	3.61	**5.83**	5.45	**2.45**	1.92	**4.26**	0.89



(a) (b)

Fig. 9: Temporal response within a concatenated scenario of our AD stack and the CARLA Autopilot. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics are illustrated in the bottom chart.

maximum torque of 126 N·m, a drag coefficient of 0.60, a damping rate of 0.2, and a delay response of 0.50 seconds. Additionally, the vehicle has a maximum RPM of 5000, a moment of inertia (Moi) of 0.05 kg·m^2, a tire friction of 0.85, and a maximum steer angle of 40 degrees.

(a) (b)

(c)

Fig. 10: Merge intersection within the DT approach. (**a**) Visual representation of the traffic flow. (**b**) Bird-eye-view of the scenario in CARLA. (**c**) Bird-eye-view of the real world intersection within our university Campus.

3) Evaluation of the Curricular Training: To evaluate the curricular approach three different trainings were executed: training exclusively in SUMO (in the Three Lane Merge Scenario), training in CARLA using the prior SUMO model (in the *Town03* merge scenario), and training the model using our DTs using the prior model in CARLA (in our simulated campus scenario). Besides we estimate the results of training our DT from scratch without the curriculum approach. The outcomes of the experiments are depicted in Table III.

As anticipated, the prior model in SUMO reduces its performance when tested in CARLA, primarily due to the dynamic environment present in this simulator. Training in the *Town03* scenario and testing in CARLA results in a competitive performance with an 88.30% success rate. However, the best resulst are obtained with the fine-tuning process, achieving similar success rates as those observed in previous sections, specifically a 91.80% success rate. On the other hand, the curriculum learning strategy, employing pre-training in SUMO and fine-tuning in CARLA, enables us to achieve model convergence 67 times faster compared to training the model from scratch.

TABLE III: A comparison in terms of performance and training time of the proposed training approaches. Success Rate (sr), Average time (av), Episodes Convergence (ec) and Training Time (tt) are represented.

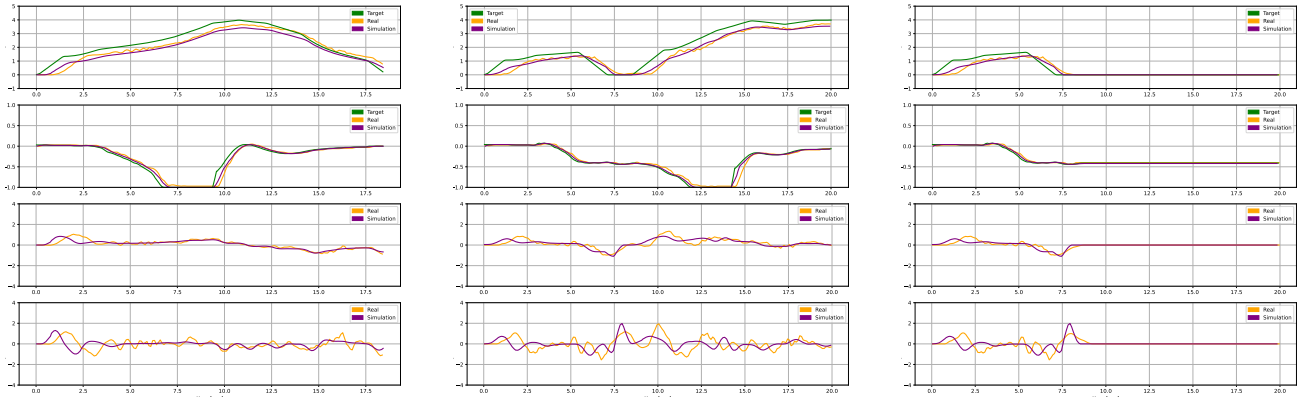
Metric	Pre-training in SUMO	Training in CARLA	Fine-tuning in CARLA	From Scratch
sr [%]	75.60	88.30	91.80	-
at (sec)	21.53	20.33	19.98	-
ec	1M	1M + 10K	1M + 15K	1M
tt (h)	5	21.5	24.75	1650

C. Parallel Execution

For the PE experiments, our focus is on identifying the discrepancies between real and simulated signals. To this end, we execute identical scenarios using the DT only in CARLA and the PE with the Real and Twin agents. We explore three distinct traffic situations in the merge scenario, each varying in vehicle density and behaviour.

1) Low Traffic Flow: The ego vehicle begins its until it reaches the optimal velocity for executing the curve. Due to the low flow of adversarial vehicles, the DRL agent allows our vehicle to continue without interruption, enabling it to merge onto the main road. The comparison reveals high similarity, with the real signal appearing slightly sharper. Both signal sets demonstrate a delayed response, due to the dynamics of the real vehicle as well as modelled in the DT. The control signals are illustrated in Figure 11a.

2) Mixed Traffic Flow: The control signals are depicted in Figure 11b for this mixed flow use case. In this scenario, the ego vehicle initiates its movement similarly to the previous case. However, the presence of adversarial vehicles prompts



(a) (b) (c)

Fig. 11: Control signals is demonstrated during a Parallel Execution (virtual and real) within the merge scenario: (**a**) Low Traffic Flow. (**b**) Mixed Traffic Flow. (**c**) High Traffic Flow. Identical control signals are provided to the real and simulated vehicles. Linear velocity is represented in the top graph, steer in the second graph, acceleration in the third graph and jerk in the bottom graph. The target (green), real (orange), and simulated (purple) signals are illustrated.

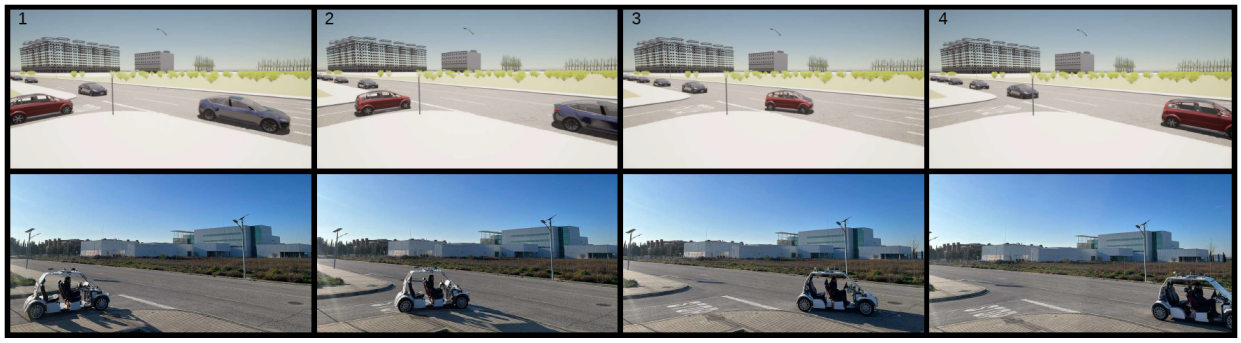


Fig. 12: Parallel Execution of the mixed traffic flow merge scenario. 1) Ego starts moving. 2) Ego stops to yield. 3) Ego starts moving again merging into the intersection. 4) Ego reaches the end of the scenario.

the DRL agent to select the stop action, leading to a reduction in velocity until the vehicle stops. Once a gap is identified, the drive action is initiated, causing the velocity signal to increase until it reaches the nominal velocity set by the low-level controller. The vehicle then merges behind an adversarial vehicle and follows it. In this scenario, the delay in vehicle response is more pronounced due to the signal changes, yet the jerk and acceleration signals remain comparable between the real and simulated responses. A visual representation of this experiment is depicted in Figure 12.

3) High Traffic Flow: As depicted in Figure 11c, the ego vehicle starts its movement in a similar way as in the previous case. The presence of continuous adversarial vehicles causes the DRL agent to select the stop action, causing the vehicle to decelerate until it stops. In this case, the adversarial vehicles present no gaps for merging, resulting the vehicle remains stationary until the episode concludes. Acceleration and jerk signals mirror those observed in the previous scenario.

VII. Conclusions and Furure Works

We have developed a hybrid Decision Making architecture real-world environments, following a Curriculum Learning methodology. This involves an initial training in a lightweight simulator (SUMO), employing a Digital Twins method in a highly realistic simulator (CARLA) to fine-tuning the model, and finally testing the complete AD stack in real-world scenarios with Augmented Reality (AR) observations through Parallel Execution. This allows us to simulate a complex scenario without the safety and economic limitations inherent to real-world environments. With this experiment, we can conclude that by using DT and PE we can validate our Decision Making architecture in a real environment.

Despite the advancements presented we identify potential lines of research for future work: The use of **Transfer Learning** for developing generalizable models that are trained in one scenario and then transferred to another. Besides, other techniques such as **Inverse Reinforcement Learning** can be used to replace the manual specification of rewards. A compelling direction for research is to create an end-to-end Reinforcement Learning architecture that learns from our hybrid DM architecture until it achieves equivalent performance. Testing with real vehicles as adversaries, incorporating **vehicle-to-vehicle** systems instead of relying solely on simulation ground truth data. Experimenting with real sensors, transitioning from simulated sensors to actual hardware to assess the performance and reliability of perception modules in real-world conditions. This would involve integrating sensors such as LiDAR, cameras, and radar. Conducting tests in a wider range of scenarios to include diverse real-world conditions.

REFERENCES

[1] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, 2020.

[2] X. Hu, S. Li, T. Huang, B. Tang, R. Huai, and L. Chen, "How simulation helps autonomous driving:a survey of sim2real, digital twins, and parallel intelligence," *CoRR*, vol. abs/2305.01263, 2023.

[3] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in *in SIMUL 2011, The Third International Conference on Advances in System Simulation*, pp. 63–68, 2011.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning* (S. Levine, V. Vanhoucke, and K. Goldberg, eds.), vol. 78 of *Proceedings of Machine Learning Research*, pp. 1–16, PMLR, 13–15 Nov 2017.

[5] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, P. Revenga, and L. M. Bergasa, "Decision making for autonomous driving stack: Shortening the gap from simulation to real-world implementations," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, pp. 3107–3113, 2024.

[6] M. Zhou, J. Luo, J. Villella, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen, A. C. Huang, Y. Wen, K. Hassanzadeh, D. Graves, D. Chen, Z. Zhu, N. Nguyen, M. Elsayed, K. Shao, S. Ahilan, B. Zhang, J. Wu, Z. Fu, K. Rezaee, P. Yadmellat, M. Rohani, N. P. Nieves, Y. Ni, S. Banijamali, A. C. Rivers, Z. Tian, D. Palenicek, H. bou Ammar, H. Zhang, W. Liu, J. Hao, and J. Wang, "Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving," 11 2020.

[7] P. Wang, S. Gao, L. Li, S. Cheng, and H. xia Zhao, "Research on driving behavior decision making system of autonomous driving vehicle based on benefit evaluation model," *Archives of Transport*, 2020.

[8] C. Vallon, Z. Ercan, A. Carvalho, and F. Borrelli, "A machine learning approach for personalized autonomous lane change initiation and control," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1590–1595, 2017.

[9] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," *CoRR*, vol. abs/1807.00412, 2018.

[10] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2156–2162, 2018.

[11] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. V. Gool, "End-to-end urban driving by imitating a reinforcement learning coach," 2021.

[12] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," 2021.

[13] M. Moghadam and G. H. Elkaim, "A hierarchical architecture for sequential decision-making in autonomous driving using deep reinforcement learning," 2019.

[14] D. Kamran, C. F. Lopez, M. Lauer, and C. Stiller, "Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning," 2020.

[15] T. Tram, I. Batkovic, M. Ali, and J. Sjöberg, "Learning when to drive in intersections by combining reinforcement learning and model predictive control," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 3263–3268, 2019.

[16] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, M. U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," *CoRR*, vol. abs/1909.11538, 2019.

[17] Z. Qiao, K. Muelling, J. M. Dolan, P. Palanisamy, and P. Mudalige, "Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1233–1238, 2018.

[18] T. Tram, A. Jansson, R. Grönberg, M. Ali, and J. Sjöberg, "Learning negotiating behavior between cars in intersections using deep q-learning," *CoRR*, vol. abs/1810.10469, 2018.

[19] H. Liu, Z. Huang, X. Mo, and C. Lv, "Augmenting reinforcement learning with transformer-based scene representation learning for decision-making of autonomous driving," 2023.

[20] H. Seong, C. Jung, S. Lee, and D. H. Shim, "Learning to drive at unsignalized intersections using attention-based deep reinforcement learning," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 559–566, 2021.

[21] A. Aksjonov and V. Kyrki, "A safety-critical decision making and control framework combining machine learning and rule-based algorithms," 2022.

[22] Y. Liu, B. Sun, Y. Tian, X. Wang, Y. Zhu, R. Huai, and Y. Shen, "Software-defined active lidars for autonomous driving: A parallel intelligence-based adaptive model," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 8, pp. 4047–4056, 2023.

[23] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Transactions on Pattern Analysis amp; Machine Intelligence*, vol. 44, pp. 4555–4576, sep 2022.

[24] Y. Song, H. Lin, E. Kaufmann, P. A. Duerr, and D. Scaramuzza, "Autonomous overtaking in gran turismo sport using curriculum reinforcement learning," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9403–9409, 2021.

[25] S. Akhauri, L. Zheng, and M. C. Lin, "Enhanced transfer learning for autonomous driving with systematic accident simulation," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5986–5993, 2020.

[26] S. Sharma, J. E. Ball, B. Tang, D. W. Carruth, M. Doude, and M. A. Islam, "Semantic segmentation with transfer learning for off-road autonomous driving," *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[27] A. Niaz, M. U. Shoukat, Y. Jia, S. Khan, F. Niaz, and M. U. Raza, "Autonomous driving test method based on digital twin: A survey," in *2021 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube)*, pp. 1–7, 2021.

[28] K. Voogd, J. P. Allamaa, J. Alonso-Mora, and T. Duy Son, "Reinforcement learning from simulation to real world autonomous driving using digital twin," *IFAC-PapersOnLine*, vol. 56, pp. 1510–1515, 01 2023.

[29] J. Wu, Z. Huang, P. Hang, C. Huang, N. De Boer, and C. Lv, "Digital twin-enabled reinforcement learning for end-to-end autonomous driving," in *2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI)*, pp. 62–65, 2021.

[30] T. Liu, X. Wang, Y. Xing, Y. Gao, B. Tian, and L. Chen, "Research on digital quadruplets in cyber-physical-social space-based parallel driving," *Chinese Journal of Intelligent Science and Technology*, vol. 1, no. 1, pp. 40–51, 2019.

[31] F.-Y. Wang, N.-N. Zheng, D. Cao, C. M. Martinez, L. Li, and T. Liu, "Parallel driving in cpss: A unified approach for transport automation and vehicle intelligence," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 577–587, 2017.

[32] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, J. Araluce, and L. M. Bergasa, "Reinforcement learning-based autonomous driving at intersections in carla simulator," *Sensors*, vol. 22, no. 21, 2022.

[33] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, F. Sánchez-García, and L. M. Bergasa, "Enhancing autonomous driving in urban scenarios: A hybrid approach with reinforcement learning and classical control (in submission)," *Sensors*, 2024.

[34] A. Diaz-Diaz, M. Ocaña, A. Llamazares, C. Gómez-Huélamo, P. Revenga, and L. M. Bergasa, "Hd maps: Exploiting opendrive potential for path planning and map monitoring," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022.

[35] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, J. Araluce, and L. M. Bergasa, "Reinforcement learning-based autonomous driving at intersections in carla simulator," *Sensors*, vol. 22, no. 21, 2022.

[36] N. Abdeselam, R. Gutiérrez-Moreno, E. López-Guillén, R. Barea, S. Montiel-Marín, and L. M. Bergasa, "Hybrid mpc and spline-based controller for lane change maneuvers in autonomous vehicles," in *2023 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, 2023.

[37] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, J. Araluce, and L. M. Bergasa, "Reinforcement learning-based autonomous driving at intersections in carla simulator," *Sensors*, vol. 22, no. 21, 2022.

[38] J. F. Arango, L. M. Bergasa, P. A. Revenga, R. Barea, E. López-Guillén, C. Gómez-Huélamo, J. Araluce, and R. Gutiérrez, "Drive-by-wire development process based on ros for an autonomous electric vehicle," *Sensors*, vol. 20, no. 21, 2020.

[39] A. Rasouli, R. Goebel, M. E. Taylor, I. Kotseruba, S. Alizadeh, T. Yang, M. Alban, F. Shkurti, Y. Zhuang, A. Scibior, K. Rezaee, A. Garg, D. Meger, J. Luo, L. Paull, W. Zhang, X. Wang, and X. Chen, "Neurips 2022 competition: Driving smarts," 2022.

[40] Z. Huang, J. Wu, and C. Lv, "Efficient deep reinforcement learning with imitative expert priors for autonomous driving," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 7391–7403, 2023.