Universidad de Alcalá

PhD. Program in Electronics: Advanced Electronic Systems. Intelligent Systems

Hybrid Online POMDP and Deep Reinforcement Learning in Decision Making for Autonomous Driving.

PhD. Thesis Presented by

Rodrigo Gutiérrez Moreno

2024

Universidad de Alcalá

PhD. Program in Electronics: Advanced Electronic Systems. Intelligent Systems

Hybrid Online POMDP and Deep Reinforcement Learning in Decision Making for Autonomous Driving.

PhD. Thesis Presented by

Rodrigo Gutiérrez Moreno

Advisors

Dr. Luis M. Bergasa Pascual

Dra. María Elena López Guillén

Alcalá de Henares, May 27th, 2024

Acknowledgements

Entre todas las posibilidades existentes, tuve la fortuna de nacer en una época moderna y en una sociedad próspera. A lo largo de mi vida, las decisiones que debía tomar eran limitadas y, aunque enfrenté algunos obstáculos, el camino resultó ser relativamente sencillo. Siguiendo la ruta establecida, conseguí un buen empleo, tenía buenas amistades y una familia maravillosa. Sin embargo, me encontraba en un punto donde, a pesar de tenerlo todo, sentía un vacío. Fue entonces cuando, impulsado por mi amigo Miguel, decidí regresar a la universidad. Entonces conocí a Luis Miguel, mi supervisor, con quien acordé unirme al grupo de investigación RobeSafe, considerando la posibilidad de emprender una tesis doctoral. Después de cuatro años de dedicación, hemos concluido con éxito el trabajo previsto. De una forma u otra, esta experiencia ha transformado mi vida.

Encontrar un propósito no es tarea sencilla; muchas veces nos dejamos llevar sin tener una dirección clara. Hoy en día, habiendo alcanzado aquel objetivo, me encuentro sin un destino específico a seguir, pero lo que sí tengo es un camino. Un camino que disfruto cada día, compartiendo con personas que me enriquecen y me importan, manteniendo mi mente activa y curiosa. Esta es la crónica de cómo terminé realizando una tesis doctoral y cómo, muy probablemente, me convertiré en doctor. Quiero aprovechar este momento para agradecer a todas las personas que me han acompañado en este viaje.

Mis agradecimientos se extienden primero a mis padres, Arantxa y Juanjo, por haberme educado desde pequeño con amor y dedicación. También recuerdo con aprecio a todos mis profesores, tanto los buenos como los malos, desde la escuela infantil hasta el instituto, cuya enseñanza ha sido fundamental en mi formación. Durante mi educación obligatoria recuerdo estudiar con Sacha. Ya en la universidad, las tardes pasadas con Gon, Pablo, Juanito, Miguel y Edu fueron cruciales; sinceramente, creo que sin ellos, no habría llegado a ser ingeniero. Un agradecimiento especial merece Gon (como administrador de Dropbox), con quien además compartí el máster. En lo que respecta a mi etapa como estudiante de doctorado, estoy agradecido por el apoyo diario de mis compañeros Felipe, Javi, Carlos, Alex, Santi, Miguel, Fabio, Navil y Pablo, así como de los profesores del grupo, Ángel, Manuel y Pedro. Por último, pero no menos importante, quiero expresar mi gratitud hacia Rafa, por sus ideas y ayuda en el día a día; Elena, por su aporte teórico y meticuloso; y Luis Miguel, por su guía y sus exhaustivas correcciones. Sin su valiosa colaboración, la tesis no habría alcanzado la calidad deseada.

Más allá del ámbito académico, deseo expresar nuevamente mi gratitud a mis padres por el simple hecho de serlo, por su presencia constante y su cuidado. Con el paso del tiempo, he descubierto que esta educación se transforma en un proceso bidireccional, en el cual aprendemos mutuamente. Mi hermano Arturo, a quien siempre he querido servir de ejemplo (aunque a veces no del mejor modo), también me enseña diariamente cómo ser una mejor persona. Considero que la fortuna de tener una familia como la mía es algo no muy común y por ello, me siento profundamente agradecido. También mencionar a mis tíos, primos y abuelos, quienes han sido y son parte de mi vida.

Mis amigos. Siempre me ha costado abrirme y confiar en las personas, por lo cual valoro profundamente a aquellos que considero mis amigos, creyendo firmemente que son los mejores. No puedo dejar de mencionar, dentro de este círculo tan especial, tanto a mis amigos de la universidad, con los que he compartido tardes, viajes y experiencias que no cambiaría por nada; como a mis colegas del laboratorio. Hay personas, como Adri, Ro y Arturo, que te acompañan a lo largo de toda la vida y me parece muy bonito viajar a León a los 30 años con las mismas personas con las que compartías mate a los 15 y salías a pedir chuches en Halloween a los 6. También me emociona pensar que personas que conocí en el instituto, como Vicky, Fati, Luis, Boni y Aitor, así como aquellos que se han ido sumando a lo largo del camino, como Antón, Bianca y Nois, continúen siendo parte de mi vida cotidiana 15 años después. Hemos compartido desde nuestras primeras fiestas y viajes solos hasta llegar a las primeras bodas y nacimientos. La vida en cierto modo es un regalo. Por último, quiero expresar un agradecimiento especial a la persona que ha estado a mi lado día tras día, cuidándonos mutuamente y aprendiendo el uno del otro: mi compañera de vida, María.

Gracias a ti, la persona que está leyendo este documento. Espero que lo disfrutes.

Resumen

La toma de decisiones es un tema fundamental en el ámbito de los vehículos inteligentes, donde la conducción autónoma representa desafíos significativos debido a los distintos comportamientos de los vehículos encontrados en la carretera y a los variados escenarios posibles en un entorno urbano. Esta tesis doctoral se centra en la implementación de un módulo de toma de decisiones híbrido en una pila de conducción autónoma para estos entornos, aprovechando una combinación de aprendizaje por refuerzo profundo basado en POMDP para las decisiones de alto nivel y metodologías de control tradicionales para la planificación local.

El objetivo principal de esta tesis es desarrollar una arquitectura realista, validable en un vehículo real, que combine la fiabilidad de las técnicas tradicionales con la adaptabilidad de los enfoques de aprendizaje profundo. La arquitectura híbrida de toma de decisiones propuesta comprende distintas capas estratégica, táctica y operativa. La capa estratégica se encarga de la generación de una trayectoria a seguir, mientras que la capa táctica toma decisiones de alto nivel. Se evaluaron cuatro algoritmos de aprendizaje por refuerzo profundo (DQN, A2C, TRPO y PPO), resultando TRPO el más eficiente para esta aplicación. La capa operativa aplica un controlador LQR (Linear-Quadratic Regulator) para el seguimiento de la trayectoria y un controlador MPC (Model Predictive Controller) para ejecutar las maniobras. La integración en línea de estos dos controladores permite la ejecución segura y confortable de las acciones de alto nivel. Esta arquitectura se evaluó en varios escenarios urbanos, como cambios de carril, rotondas, incorporaciones y cruces, utilizando las plataformas de simulación SUMO (Simulation of Urban MObility) y CARLA (Car Learning to Act). Esta propuesta no solo resuelve casos de uso complejos de manera individual, sino también en secuencias concatenadas.

Además, este trabajo investiga la transición desde entornos de entrenamiento simulados a experimentos reales de la arquitectura de conducción autónoma modular. Esta transición se realiza mediante un aprendizaje curricular a través de la implementación de gemelos digitales y tecnologías de inteligencia paralela, reduciendo significativamente la brecha entre simulacion y realidad. La viabilidad de este enfoque se evidencia a través de una ejecucion paralela, donde las pruebas simuladas y las reales se realizan de forma sincronizada.

Nuestro módulo táctico se compara cuantitativamente en SUMO con arquitecturas del estado del arte dentro del marco estándar SMARTS (Scalable Multi-Agent Reinforcement Learning Training School), logrando resultados competitivos en varios escenarios. Además, el rendimiento de la arquitectura completa de navegación autónoma se evalúa en CARLA utilizando métricas como la tasa de éxito, las dinámicas de aceleración, la sobreaceleración y el tiempo para completar escenarios, comparando nuestro sistema con el Autopilot de CARLA. Los resultados demuestran que, aunque las tasas de éxito son similares, nuestro sistema destaca significativamente en términos de suavidad y eficiencia en la conducción.

En última instancia, esta tesis sienta las bases para un sistema de decision autónomo más versátil, seguro y eficiente. Muestra el potencial de las arquitecturas híbridas para impulsar la vanguardia de la investigación en conducción autónoma y sienta las bases para futuras exploraciones en la integración del entrenamiento basado en simulación con la aplicación en el mundo real.

Palabras clave: Conducción autónoma, Aprendizaje Profundo por Refuerzo, Toma de Decisiones, Control de Vehículos, Gemelos Digitales, Simulador CARLA.

Abstract

Decision-Making (DM) is a fundamental topic in the domain of intelligent vehicles, where Autonomous Driving (AD) poses significant challenges due to the variable behaviours of surrounding vehicles and the wide array of encountered scenarios. This doctoral thesis delves into the implementation of a hybrid DM module within an Autonomous Driving (AD) stack for urban environments, leveraging a combination of Deep Reinforcement Learning (DRL) of POMDP models for high-level actions and classical control methodologies for local planning.

The primary aim of this thesis is to develop a realistic architecture able to be validated on a real vehicle that marries the reliability of traditional techniques with the adaptability of Deep Learning (DL) approaches. The proposed hybrid DM architecture encompasses strategy, tactical, and operative levels. The strategy level generates the trajectory to be followed, while the tactical decision level utilizes a DRL algorithm for high-level decisions. Four DRL approaches (DQN, A2C, TRPO, PPO) are compared under identical conditions, resulting the TRPO the most efficient for this application. The operative level, on the other hand, applies a Linear-Quadratic Regulator (LQR) for trajectory tracking and a Model Predictive Controller (MPC) for executing manoeuvres. The online integration of these two controllers allows the execution of high-level actions in a comfortable and safe way. This architecture is thoroughly evaluated in various urban driving scenarios, such as lane changes, roundabouts, merges, and crossroad intersections, utilizing the Simulation of Urban MObility (SUMO) and Car Learning to Act (CARLA) simulation platforms. This approach not only solves individual complex urban scenarios but also handles concatenated scenarios.

Furthermore, this work investigates the crucial transition from simulated training environments to real-world deployment of AD stacks. This transition is carried out through a Curriculum Learning (CL) approach, facilitated by the deployment of Digital Twins (DT) and parallel intelligence (PI) technologies, significantly narrowing the Reality Gap (RG) and enhancing the applicability of learned behaviours. The viability of this approach is evidenced through a Parallel Execution (PE), wherein simulated and real-world tests proceed concurrently.

Our tactical decision proposal is quantitatively compared in SUMO against State-of-the-Art (SOTA) architectures within a standard framework such as Scalable Multi-Agent

Reinforcement Learning Training School (SMARTS), achieving competitive results across various scenarios. Additionally, the whole AD stack, which includes the proposed hybrid DM architecture, is quantitatively assessed in CARLA using metrics such as success rate, jerk dynamics, acceleration, and time to complete scenarios, comparing our system with the CARLA Autopilot. The results demonstrate that while success rates are comparable, our system significantly excels in terms of smoothness and efficiency.

Ultimately, this thesis lays the foundation for a more versatile, safe and efficient DM architecture. It shows the potential of hybrid architectures to drive the cutting edge of AD research and lays the groundwork for further explorations into the integration of simulation-based training with real-world application.

Keywords: Autonomous Driving, Deep Reinforcement Learning, Decision-Making, Vehicle Control, Digital Twins, CARLA Simulator.

Contents

List of Figures

List of Tables

List of Acronyms

A2C	Advantage Actor Critic.
ACC	Adaptative Cruise Control.
AD	Autonomous Driving.
ADAS	Advanced Driver Assistance Systems.
AVs	Autonomous Vehicles.
CARLA	Car Learning to Act.
CL	Curriculum Learning.
CNNs	Convolutional Neural Networks.
DBW	Drive-by-Wire.
DL	Deep Learning.
DM	Decision Making.
DNNs	Deep Neural Networks.
DQN	Deep Q-Network.
DRL	Deep Reinforcement Learning.
DT	Digital Twins.
FSM	Finite State Machine.
HD	High Definition.
IPC	Inter-Process Communication.
IRL	Inverse Reinforcement Learning.
ITS	Intelligent Transportation Systems.
LGP	Lane Graph Planner.
LQR	Linear–quadratic regulator.
LWP	Lane Waypoint Planner.

MDP	Markov Decision Process.
MDPs	Markov Decision Processes.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
MPC	Model Predictive Control.
NN	Neural Network.
NNs	Neural Networks.
PE	Parallel Execution.
PI	Parallel Intelligence.
PN	Petri Nets.
POMDP	Partially Observable Markov Decision Process.
POMDPs	Partially Observable Markov Decision Processes.
PPO	Proximal Policy Optimization.
RG	Reality Gap.
RL	Reinforcement Learning.
RNNs	Recurrent Neural Networks.
ROS	Robot Operating System.
SB3	Stable Baselines3.
SMARTS	Scalable Multi-Agent Reinforcement Learning Training School.
SOTA	State of the Art.
SUMO	Simulation of Urban MObility.
SVM	Support Vector Machine.
TM	Traffic Manager.
TraCI	Traffic Control Interface.
TRPO	Trust Region Policy Optimization.

Chapter 1

Introduction

Toma este momento. Sumérgete en sus detalles. Responde a esta persona, este desafío, esta acción. Deja las evasiones. Deja de buscar problemas innecesarios. Es hora de vivir; para habitar por completo la situación en la que te encuentras ahora.

Epicteto

1.1. Aim

The use of Deep Learning (DL) algorithms in the domain of Decision Making (DM) for Autonomous Vehicles (AVs) 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 dissertation aims to provide a realistic implementation of a hybrid DM module in an Autonomous Driving (AD) stack, integrating the benefits of DL algorithms and the reliability of classical methodologies.

This thesis encompasses an understanding of learning-based algorithms, the implementation of scenarios in simulated environments, and the integration of AD modules. Specifically, the author addresses the DM problem by employing a Partially Observable Markov Decision Process (POMDP) formulation and offers a solution through the use of Deep Reinforcement Learning (DRL) algorithms. Furthermore, an additional control module to implement the decisions on a real vehicle in a safe and comfortable way through a hybrid architecture is presented. The dissertation introduces a methodology aimed at narrowing the gap between simulation and real-world application, developing a Digital Twins (DT) and validating our proposal on an actual vehicle.

The author's proposal in this dissertation aims to establish a hybrid DM system designed to empower smart vehicles, enabling them to effectively navigate and operate within complex urban environments.

1.2. Motivation

AD unquestionably stands as one of the most revolutionary technologies in recent decades. The multitude of discoveries and innovations in the field has stimulated the interest of researchers in Intelligent Transportation Systems (ITS). This collective interest not only harbours the potential but also the responsibility to mould the future of urban mobility and transportation.

The rapid progression of 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 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, 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, with the objective of developing a framework for autonomous DM in real-world scenarios within the AD context.

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 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 is able to 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, during which an agent incrementally acquires a desired behaviour by continuously interacting with its environment.

However, the application of DRL techniques not always satisfies the requirements of AD. As mentioned before, the main problem to be solved by AVs is related with traffic safety, where DRL approaches may have some difficulties. These techniques are divided

in model-based and model-free. Model-based methods incorporate vehicle dynamics but frequently encounter difficulties, 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 clearly identify high-level behaviour. This is why we propose a hybrid approach in this dissertation. 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, including the vehicle's dynamics.

This thesis 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. A schematic illustration of the hybrid proposal is depicted in Figure 1.1. At the strategy level (planning module), vehicle localization data is received and map information is extracted. This level generates trajectories, encompassing waypoints to follow and contextual data. The perception module provides information about other traffic participants. The tactical level (DM module) uses context information and localization to enable a POMDP fine-tuned for each specific scenario, using perception data to select a high-level action. This chosen action, along with the waypoints, are then processed by the operative level (control module), resulting in the generation of low-level control commands to be sent to the vehicle's Drive-by-Wire (DBW) module.

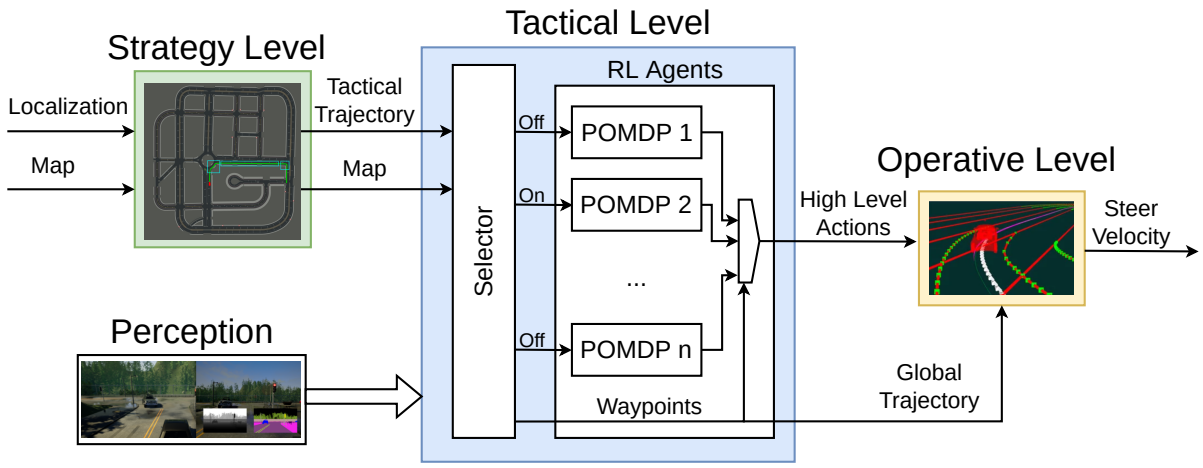


Figure 1.1: An overview of the proposed framework. It includes the strategy, the tactical and the operative levels.

1.3. Contributions of Dissertation

This doctoral thesis focuses on developing 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. In the context of AD systems, this work integrates multiple use cases for learning in the DM. A novel hybrid method is introduced, using DRL to handle the uncertainties of dynamic environments in the high-level decisions, while classical planning, mapping, and control techniques are employed to navigate in a safe and comfortable way taken into account the vehicle's dynamics.

The key contributions of this thesis are summarized as follows:

- **Contribution 1:** This dissertation introduces the design and implementation of a DM module specifically tailored for AD. It proposes a novel 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. This hybrid approach presents a novel advancement compared to the State of the Art (SOTA) RL AD methodologies. Our approach not only solves individual complex urban scenarios but also handles concatenated scenarios. This contribution was 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. This was made public in Sensors journal [10].
- **Contribution 3:** The thesis implements various SOTA DRL algorithms within the domain of AD to evaluate high-level actions. It features the design of four distinct urban scenarios in the Simulation of Urban MObility (SUMO) [11] platform. Within these scenarios, four DRL algorithms are rigorously compared under identical conditions. This comparison aims to identify the algorithm that most effectively enhances the performance of AVs in urban environments. Additionally, we offer a comparative analysis of our approach against other SOTA methodologies within the standard framework Scalable Multi-Agent Reinforcement Learning Training School (SMARTS) [12]. At the 2023 ITSC conference, this contribution was published [13].
- **Contribution 4:** This study also presents a RL framework developed within the Car Learning to Act (CARLA) [14] simulator to evaluate complete vehicle navigation with dynamics. This framework, grounded in the OpenAI Gym [15] toolkit, offers a standardized, open-source environment for both training and testing RL strategies. 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. As such, this framework not only facilitates comprehensive assessment but also establishes a new baseline metric for future RL research in AD. This contribution was released in Sensors journal paper [16].

- **Contribution 5:** The thesis culminates with a substantial advancement towards real-world applicability through the development of DT based on our actual platform. We present an experiment wherein both a simulated and a real vehicle operate simultaneously in the same scenario resulting on a Parallel Execution (PE). The simulated vehicle mirrors the actions of the real vehicle, while the real vehicle makes decisions based on other simulated traffic participants. This approach effectively bridges the gap between simulated models and real implementation, facilitating a more precise and impactful application of research findings to real-world AD scenarios. This contribution has been submitted to the conference IV 2024 [17].

1.4. Organization of Dissertation

The organization of this document has been done as follows:

- **Chapter 2** presents a technical background, mostly focused on DM. A classification of the different methods and a description of various SOTA algorithms is presented.
- **Chapter 3** reviews the existing literature, highlighting the most relevant RL proposals and AVs architectures.
- **Chapter 4** presents our whole DM architecture and reviews various simulation frameworks present in the literature. Additionally, it introduces our own simulation framework, designed to fulfil our research objectives.
- **Chapter 5** presents our low-level controller for trajectory tracking and manoeuvre execution.
- **Chapter 6** evaluates different DM algorithms under urban scenarios in SUMO and describes our DRL proposal.
- **Chapter 7** describes the proposed urban scenarios in CARLA and provides metrics and graphics to validate the AD stack of the research group, which includes our DM proposal, within those scenarios. Furthermore, it details the implementation of a PE that lever our proposal to a real implementation.
- **Chapter 8** Conclusions and Future Works.

Chapter 2

Background

El que es prudente es moderado; el que es moderado es constante; el que es constante es imperturbable; el que es imperturbable vive sin tristeza; el que vive sin tristeza es feliz; luego el prudente es feliz.

Lucio Anneo Séneca

2.1. Introduction

This chapter delves into the foundational concepts and methodologies underpinning DM in AD. The first section provides a comprehensive overview of the AD landscape. It examines the distinctions between end-to-end and modular approaches, focusing on the different modules of a complete AD stack. Moving forward, the second section introduces the mathematical framework used for the DM implementation in this work, that is instrumental in tackling the uncertainty inherent in urban scenarios. The third section explores the cutting-edge realm of ML that intersects with DM. DRL combines the principles of DL and RL to create systems that can learn optimal behaviours through interactions with their environment. This section presents the fundamental concepts of DRL and explains the algorithms used in this dissertation.

2.2. Autonomous Driving

Current AD SOTA identifies two principal software architecture categories: End-to-End and modular. The comprehensive AD architecture spans from initial sensing to the final control of longitudinal (throttle/brake) and lateral (steering angle) movements. These control signals are inputs to DBW systems, which physically manoeuvres the vehicle. 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 [18], [19]. 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 [20], [21].

The hybrid DM architecture proposed in this thesis is integrated into the AD stack of the Robesafe group [22]. Figure 2.1 illustrates the modular approach adopted by the group and identifies the main modules developed in this thesis (outlined with red dashed lines).

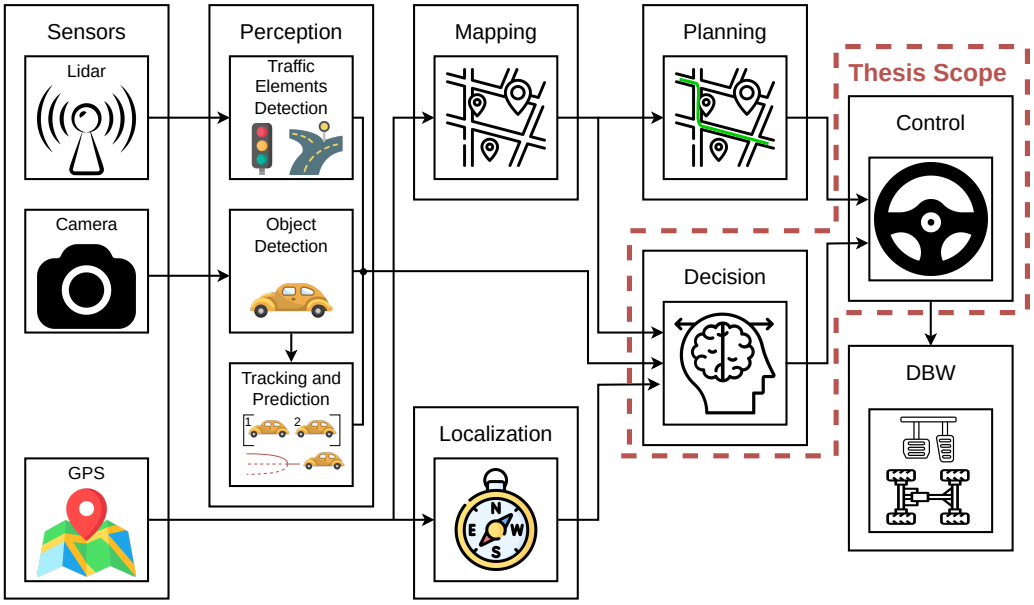


Figure 2.1: Autonomous Driving Stack modular pipeline.

In this AD stack we identify the following modules:

- **Localization module:** Accurately positions the vehicle on a map in real-time, primarily using GNSS, IMUs, wheel odometry, and cameras.
- **Perception module:** Interprets the environment surrounding the vehicle using sensor data. It may involve multi-stage processes including traffic elements detection, obstacle detection, tracking and trajectory prediction.
- **Mapping module:** Creates a detailed environmental model through which the vehicle navigates, often employing a High Definition (HD) map graph.
- **Planning module:** Includes route planning. The route planner determines the optimal path using localization and mapping data. This module is often known as Global Planning.

- **Decision module:** Manages high-level driving behaviours based on the global route, perception outputs, and current location. This module is commonly referred to as Behavioural Planning.
- **Control module:** Responsible for generating control commands based on waypoints from the trajectory planner and high-level actions from the decision module. Frequently, this module is called Local Planning.
- **Drive-by-Wire module:** Generates the physical signals required by the vehicle's actuators to follow the received control commands.

This thesis 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 [23]:

- **Rule-Based Systems:** Use a predefined set of rules to guide the vehicle's decisions. While straightforward to implement, they can be inflexible and may not perform well in complex or unforeseen scenarios.
- **Finite State Machines:** Model the DM process as a series of states and transitions. They are more adaptable than rule-based systems but can become difficult to manage as complexity grows.
- **Behaviour Trees:** This approach structures the DM process in a tree-like hierarchy, allowing for modular and scalable systems. Behaviour trees can handle a variety of scenarios but may require significant tuning and maintenance.
- **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 patterns and experiences. They offer adaptability and can handle complex environments but require substantial data and computational resources, mainly in DL systems.
- **Reinforcement Learning:** is a subset of ML, characterized by its focus on training an agent through experiments to interact effectively with its environment. RL handles the uncertainty in dynamic and complex driving environments.

Among the various approaches, RL stand 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.

2.3. 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 [24]. 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.

2.4. Reinforcement Learning

In this section, we present the knowledge acquired by the author during the dissertation, alongside the concepts derived from [25]. 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:

$$P(s_{t+1}|s_t, s_{t-1}, \ldots, s_0, a_t) = P(s_{t+1}|s_t, a_t). \tag{2.1}$$

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:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

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:

$$V^\pi(s) \triangleq \mathbb{E}[G^\pi | s = s_0] \tag{2.3}$$

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:

$$Q^\pi(s, a) \triangleq \mathbb{E}[G^\pi | s = s_0, a = a_0] \tag{2.4}$$

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:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V^\pi(s')], \tag{2.5}$$

and for the state-action value function:

$$Q^\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s', a')]. \tag{2.6}$$

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:

$$V^*(s) \triangleq V^{\pi^*}(s) = \max_\pi(s) V^\pi(s) = \max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V^\pi(s')\right] \tag{2.7}$$

The optimal state-action value function is defined as:

$$Q^*(s,a) \triangleq Q^{\pi^*}(s) = \max_{\pi}(s) Q^{\pi}(s,a) = \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} Q^*(s', a') \right] \tag{2.8}$$

and the optimal policy is defined as:

$$\pi^*(s) = \arg\max_{a} Q^*(s,a) \tag{2.9}$$

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. We differentiate two principal families of RL algorithms: Tabular Solution Methods, which are quintessential for discrete and manageable state spaces, and Approximate Solution Methods, essential for handling large or continuous state spaces.

2.4.1. Tabular Solution Methods

Dynamic programming methods, such as Policy Iteration and Value Iteration, rely on the Markov Decision Process (MDP) framework and are effective in environments with perfect state information. They iteratively evaluate and improve policies based on the Bellman equations. Monte Carlo methods, in contrast, do not require a complete model of the environment. They learn directly from episodes of experience by averaging the returns following each state, thereby estimating the value function based on sample returns. Temporal difference learning, a blend of the previous methods, learns directly from incomplete episodes by bootstrapping, i.e., updating estimates based in part on other learned estimates. Temporal difference methods, are powerful as they can learn before knowing the final outcome and can update estimates online after each step.

2.4.2. Approximate Solution Methods

In the field of RL, the distinction between on-policy and off-policy methods is pivotal. On-policy methods involve learning the value of the policy being executed by the agent. This approach ensures that the policy evaluation is consistent with the policy being improved. Off-policy methods decouple the policy being learned from the policy being executed. These methods enable learning an optimal policy independently of the agent's actions, offering greater flexibility and efficiency, especially in environments where exploration is crucial.

2.5. 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. 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 thesis, 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) [26], which is value-based, and Advantage Actor Critic (A2C) [27], Trust Region Policy Optimization (TRPO) [28], and Proximal Policy Optimization (PPO) [29], which are policy-based.

Before detailing the algorithms, we introduce the loss function concept. This function is used during the training of the NN that approximates either the policy or the value function. This function measures the difference between the predicted values and the target values. By minimizing this loss, the NN 's predictions are refined, leading to more accurate estimations and better policy decisions.

2.5.1. Deep Q-Network (DQN)

The basic idea behind Q-learning is to use the Bellman optimality equation as an iterative update $Q_{i+1}(s, a)$, and it can be shown that this converges to the optimal Q-function, i.e., $Q_i \to Q^*$ as $i \to \infty$.

For most problems, it is impractical to represent the Q-function as a table containing values for every combination of s and a. Instead, a function approximator is used, such as a NN with parameters θ, to estimate Q-values, i.e., $Q(s, a; \theta)$. This is done by minimizing the following loss at each step i:

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right] \tag{2.10}$$

where $y_i = \mathbb{E}_{s'\sim E}[r + \gamma \max_{a'} Q(s',a';\theta_{i-1})|s,a]$ is the target for iteration i and $\rho(s,a)$ is the behaviour distribution. The parameters from the previous iteration θ_{i-1} are held fixed when optimising the loss function $L_i(\theta_i)$. Note that the targets depend on the network weights.

Q-learning is an off-policy algorithm that learns about the greedy policy π while using a different behaviour policy to act in the environment. This behaviour policy is usually an ϵ-greedy policy that selects the greedy action with probability $1-\epsilon$ and a random action with probability ϵ to ensure good coverage of the state-action space.

2.5.2. Advantage Actor Critic (A2C)

The A2C algorithm is a DRL approach that involves two main components: an actor and a critic. The actor decides the actions to take, modelled by the policy π_θ. The critic evaluates the actions taken by the actor, represented by the value function, which estimates the expected return from state.

The advantage function measures the quality of an action relative to the average action at a given state.

$$\hat{A}_t = A(s_t,a_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.11}$$

The critic applies the MSE between the target and the current state.

$$L(\theta) = (r_{t+1} + \gamma V(s_{t+1}) - V(s_t))^2 \tag{2.12}$$

To calculate the loss for an actor, an action is selected from the policy. Then, the loss is determined using the negative log likelihood of this probability, adjusted according to the advantage.

$$L(\theta) = \hat{\mathbb{E}}_t[log\pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.13}$$

where π_θ is a stochastic policy parameterized by θ, $\hat{A}_t$ is an estimator of the advantage function at timestep t and $\hat{E}_t$ is the expectation . The policy is updated by minimizing the loss function, often using gradient ascent methods. A2C also updates the value function to reduce the variance of the policy gradient.

2.5.3. Trust Region Policy Optimization (TRPO)

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:

$$\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot|s), \pi_\theta(\cdot|s)] \leq \delta \tag{2.14}$$

where δ is a small positive number that defines the size of the trust region.

The loss function in TRPO is defined as follows:

$$L(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \tag{2.15}$$

Here, $\pi_\theta(a_t|s_t)$ represents the probability under the new policy parameters θ 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.

TRPO, being on-policy, explores by sampling actions from its stochastic policy. The policy becomes progressively less random over training, focusing on exploiting known rewards, which may lead to local optima.

2.5.4. Proximal Policy Optimization (PPO)

PPO is considered a strong alternative in the DRL research community due to its impressive results in both benchmarks and real-world applications. It is an on-policy, actor-critic method, that employs trust region methods and clipping to practically ensure improvement in the policy behaviour in every training iteration. The loss function, whose gradient is the policy gradient estimator, has the general form:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t[log\pi_\theta(a_t|s_t)\hat{A}_t] \tag{2.16}$$

where $\mathbb{E}_t$ is the expectation, π_θ is the policy, $\hat{A}_t$ is an estimator of the advantage function at time step t.

While it is appealing to perform multiple steps of optimization on this loss function $L^{PG}(\theta)$, doing so is not well-justified, and empirically it often leads to destructively large policy updates.

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \tag{2.17}$$

where c_1, c_2 are coefficients, S denotes an entropy bonus, and $L_t^{VF}(\theta)$ is a squared-error loss $(V_\theta(s_t) - Vt^{targ})^2$. L_t^{CLIP} is the clipped surrogate objective, which takes the form:

$$L_t^{CLIP}(\theta) = \hat{\mathbb{E}}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.18}$$

where epsilon is a hyper-parameter and $r_t(\theta)$ is the probability ratio $r_t(\theta) = \pi_\theta(a_t|s_t)/\pi_{\theta_{old}}(a_t|s_t)$. The value of L_t^{CLIP} is the minimum of the clipped and the unclipped objective, so the final objective is a lower bound on the unclipped objective. Large policy updates are avoided by clipping this probability ratio inside the interval $[1-\epsilon, 1+\epsilon]$.

PPO's capability for incremental learning and its adaptability to continuous action spaces make it a potent solution for various complex problems, including those in the field of robotics and other domains requiring efficient learning.

2.6. Summary

This chapter provides a comprehensive background on AD and DM in AD, focusing on simulation-based RL methodologies. It begins with an overview of AD architectures, comparing End-to-End and modular approaches, and highlighting the focus on software components in AD. The chapter emphasizes the decision module in AD, crucial for safe and efficient navigation, exploring various approaches including rule-based systems, finite state machines, behaviour trees, ML, DL and RL.

The mathematical framework of POMDPs is introduced, extending the MDP framework by incorporating uncertainty in observations. The chapter also delves into RL, discussing concepts such as policies, value functions, and the Bellman equation, essential for training optimal DM strategies in RL.

Deep DRL is explored in detail, highlighting its ability to handle POMDPs effectively without explicit feature engineering, thanks to DNNs. Various DRL algorithms, such as DQN, A2C, TRPO, and PPO, are reviewed, focusing on their value-based and policy-based categorizations.

Chapter 3

Literature Review

Cada vez que estés a punto de señalar un defecto en otra persona, hazte la siguiente pregunta: ¿Qué defecto en mí se parece al que estoy a punto de criticar?

Marco Aurelio

3.1. Introduction

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. In the realm of DM, two primary approaches are prevalent: classical methods and learning-based methods. Classical methods rely on predefined rules and algorithms, offering reliability and predictability, but they may lack adaptability in unforeseen scenarios. On the other hand, learning-based methods, leveraging new computational technologies, have gained significant popularity. These methods excel in handling the unpredictability and complexity inherent in dynamic driving environments. The detailed advantages and limitations of each approach are compared in Table 3.1.

Acknowledging the presence of numerous surveys on DM for AVs, this chapter aims to provide a comprehensive overview that encompasses both classical and learning-based approaches. Particularly, it places a stronger emphasis on exploring DRL methods, delving deeper into the literature to provide a more detailed understanding of these advanced techniques within the context of AD.

Table 3.1: Characteristic of Different Methods for Decision-Making

Methods		Pros	Cons
Classical Methods	Rule-based Methods	Strong interpretability, adjustability, and feasibility of implementations. Good decision-making breadth.	Difficult to handle complex driving conditions. Poor robustness in dynamic environments.
	Optimization Methods	Optimized decisions can be generated with better modeling of interactions between traffic participants.	"Optimal strategy" often inconsistent with practical applications.
	Probabilistic Methods	Convenient to combine with other methods.	Low computational efficiency. Difficult to generate optimal decisions in complex environments.
Learning-Based Methods	Statistic Learning-Based Methods	Good versatility and suitable for simple scenarios with sufficient environmental information.	Requires extensive training datasets. Low decision-making accuracy.
	Deep Learning-Based Methods	High decision-making accuracy in specific scenarios. Utilizes environmental information fully.	Poor universality in dynamic scenarios. Quality of training datasets influences effectiveness. Lack of explainability.
	Reinforcement Learning-Based Methods	Learning from interaction, no need for labeled data. Flexible algorithm framework. Adaptability and goal-oriented.	Great dependence on the establishment of a reward function. Sample inefficiency.
	Deep Reinforcement Learning-Based Methods	Handling high-dimensional inputs with improved generalization. Flexible algorithm framework with high expandability.	High computational cost. Overfitting risk to certain scenarios. Data inefficiency and stability issues.

3.2. Classical Methods

DM methods in classical contexts can be categorized into three main types: rule-based, optimization, and probabilistic methods.

3.2.1. Rule-Based Methods

Rule-based DM methods utilize a database of rules that incorporate traffic laws, driving experiences, and knowledge. These methods base strategies on the various statuses of vehicles. A key example is the Finite State Machine (FSM) method, which employs a mathematical model with discrete inputs and outputs. Actions are generated in response to external events, leading to transitions in the states of agents. Research in this area includes the development of hybrid flow diagrams for DM based on environmental con-

ditions, multi-point turn frameworks for vehicle steering, and hierarchical frameworks for generating tactical and strategic behaviour. These methodologies are further categorized into tandem, parallel, and hybrid types [30] according to their logical structure.

FSM methods have been widely used in AVs projects. For instance, the tandem type was utilized in Talos [31]; the parallel type was employed in Junior [32] and Bertha [33]; while the hybrid type was implemented in Odin [23]. Apart from typical FSM methods in vehicles, research has also focused on specific rule bases. A hybrid flow diagram was designed for DM based on environmental conditions, selecting feasible trajectories without detailed prior maps, and tested on the "AUTOPIA" instrument vehicle [34]. A "multi-point turn" decision framework was proposed in [35], which minimized steering widths for vehicle control. Furthermore, a hierarchical framework for tactical and strategical behaviour generation was established in [36], offering high generality and expandability for various scenarios.

A further development from FSM are Petri Nets (PN). While FSM contains only a single state, PN can have one or more tokens, representing a concurrent net. A state in a PN is executed as soon as all inputs have the required number of tokens. PN have been widely utilized in AD. Previous research, including our own proposal [22], has concentrated on the DM module within the AD stack. Similar strategies have been adopted for AVs as evidenced in [37], while other studies have explored cooperative behaviours [38] and traffic flow control [39].

3.2.2. Optimization Methods

Optimization methods for DM typically rely on a reward or utility function. MPC is a notable method in this category. It has been used for controlling overall traffic situations, influencing behaviours of vehicles, and considering vehicle-pedestrian interactions [40], [41]. Other approaches involve combining MPC with Inverse Reinforcement Learning (IRL) to establish suitable cost functions [42]. Game Theory is also used in this domain, where the optimal strategies of agents are assumed and actions are based on these strategies. This includes applications in multi-vehicle DM [43], cooperative behaviours in AVs platoons[44], and vehicle cooperation in dense urban traffic scenarios[45]. Additionally, evaluation-based methods like multiple attribute-based DM with AHP and TOPSIS have been utilized to select optimal driving behaviours [46].

3.2.3. Probabilistic Methods

Probabilistic methods in DM rely on probability theory. These methods require the establishment of a probabilistic model for behaviour determination. Examples include the use of Probabilistic Graphical Models for estimating intentions in merging scenarios and generating motion commands [47]. There is also the development of Two-Sequential Level

Bayesian Decision Networks for lane change [48], involving risk assessments and ensuring safe behaviour generation. Works in this area extend to the combination of methods like the Extended Kalman Filter for enhanced safety in behaviour prediction [49].

3.3. Learning-based Methods

Learning-based methods for AD DM utilize artificial intelligence technologies, typically requiring established driving data samples. These methods can be broadly classified into four categories: statistical learning-based, DL-based, RL-based methods, and DRL-based methods.

3.3.1. Statistical Learning-Based Methods

Statistical learning-based methods enable AVs to acquire human-like DM skills through extensive training data. Common approaches include Support Vector Machine (SVM) and AdaBoost. In [50], SVM was employed for lane change DM, integrating MPC with safety constraints. Subsequently, Bayesian parameter optimization enhanced the lane change model's accuracy [51]. AdaBoost has been applied in "Cut-In" scenarios [52] for risk assessment, demonstrating its effectiveness in ensuring safe manoeuvres.

3.3.2. Deep Learning-based Methods

DL frameworks, while similar to traditional ML, distinguish themselves by employing NNs for data feature learning. End-to-end systems, particularly advantageous in image processing, have been pivotal in DM. For instance, Convolutional Neural Networks (CNNs) have been used for end-to-end DM, utilizing front-camera images to generate steering commands [53], [54]. "DriveNet" [55] further demonstrated the efficacy of CNNs under varied lighting conditions. Some research also combines visual images with measurements and high-level commands for comprehensive control instructions, as seen in [56], [57]. Lidar point cloud data has also been instrumental, with CNNs integration yielding promising results [58], [59].

3.3.3. Reinforcement Learning-based Methods

RL is a widely adopted approach in learning-based methods, aiming to maximize returns through trial-and-error strategies. Typical solvers have been used in various contexts, including pedestrian-vehicle interactions [60] and driving ethics considerations [61]. POMDPs have been effectively utilized in scenarios like occluded intersections [62], [63], with various techniques improving computational efficiency and safety.

Recent advancements have utilized DNNs in the role of function approximators, leading to the development of the DRL paradigm. These DRL algorithms have demonstrated performance on par with or surpassing human capabilities in challenging tasks, including playing Atari games [64] and mastering the board game Go [65]. A comprehensive analysis of DRL's application in AD is presented in the following section.

3.3.4. Deep Reinforcement Learning-based Methods

In recent years, DRL has emerged as the predominant methodology in the domain of DM for AVs. For DRL implementation, three fundamental elements warrant meticulous consideration: the definition of the state space, the articulation of the action space, and the construction of the reward function. Each of these components plays a pivotal role in the effective modelling and execution of DRL algorithms, influencing the learning process and the overall performance of the system in complex, real-world scenarios.

The issue of state representation is paramount in the field of RL-based AVs. Traditional approaches to state representation typically revolve around lower-dimensional features, such as proximity to obstacles, lane positioning, and vehicular velocities [9], [66]. These models exhibit robust behaviour in complex situations, demonstrating strong resilience and adaptability. However, their ability to accurately represent the dynamics and interactions of urban driving contexts may be somewhat limited. In response to this limitation, several innovative approaches have been developed. These include the adoption of higher-dimensional, such as Bird-Eye-View imagery [67], image augmentation techniques [68], and the use of occupancy grids [69]. The utilization of CNNs [70] has significantly contributed to the convergence, enhancing the network's invariance to alterations in the state order. Additionally, Recurrent Neural Networks (RNNs) [71] have been effectively employed to process historical information from an episode, thereby enriching the DM context by incorporating temporal data sequences. Transformer-based models have emerged as a focal point due to their proficiency in capturing long-term dependencies and complex interactions within sequential data. Specifically, within the AVs domain, transformers have shown their utility in streamlining computational demands in end-to-end models [72], as well as in facilitating forward-looking, prediction-aware planning strategies [73]. These developments underscore a significant stride towards more sophisticated and efficient state representation techniques, pivotal for the advancement of AD technologies. These advanced methods have demonstrated considerable success in enhancing the generalizability and resilience of DM mechanisms within AD systems. However, they pose challenges in implementation, require substantial computational resources, and are often difficult to train effectively.

The definition of the action space constitutes another critical aspect in the realm of AVs decision systems. In the existing literature, two approaches are predominant. In the first approach, high-level actions mirror human decisions and requires conversion by

an intermediate module into operational commands. Examples include directives such as "stop," "drive slow," or "drive fast" [74], and decisions like "take way" or "give way" [75], which primarily influence the vehicle's longitudinal velocity. Additionally, some others are focused on lane changing with commands like "change left," "idle," and "change right" [76]. This approach, while easier to implement, requires the integration of a low-level controller to manage the local planning. In the second approach, low-level actions directly control the vehicle's actuators, with actions like setting longitudinal velocities in meters per second [77] or managing both lateral and linear controls [78], involving specific commands for steering, throttle, and braking. This approach enables the vehicle to learn more complex behaviours and interact more extensively with the environment. However, in practice, this often leads to the vehicle generating sharp control signals focused primarily on task completion as quickly as possible. This can result in a driving style that prioritizes efficiency over comfort and safety.

The reward function stands as a fundamental component in the design of POMDPs. Although diverse in its formulation, the design of this function generally adheres to a standardized practice. Commonly, a positive reward is given for successfully ending an episode, while a collision incurs a negative reward. In [79], a positive reward is linked to the velocity of the vehicle, providing an incentive to move. A different perspective, given by [71], assigns a negative reward based on the simulation time of each episode. Moreover, the integration of risk parameters into the reward function is also considered, as demonstrated in some works such as [80], [81].

3.4. Autonomous Driving Architectures

Thus far, we have presented several works where diverse DM techniques have been applied to specific scenarios. These studies primarily offer insights into high-level actions without detailing the complete trajectories executed by the vehicle. Since the goal of this thesis is to validate a complete AD stack, this final section reviews some representative AD architectures in the literature, particularly in the context of DM and DRL.

3.4.1. Transformer-based Reinforcement Learning

The use of transformer-based algorithms has been gaining relevance in recent years in the context of AD [82]. With various works following this approach [83], [84]. One of the most relevant and representative framework in this line employs a Scene-Rep Transformer to enhance RL DM capabilities [85]. 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: $s_t = [M_t; K_t]$. The

actions proposed in this work are the longitudinal velocity of the ego vehicle and a lane change signal, executed by the SUMO simulator.

The proposed architecture is depicted in Figure 3.1. Given perception-processed vectorized scene inputs, the multistage Transformer encodes the multi-modal information with interaction awareness. In the training phase, a sequential latent Transformer conducts representation learning using consecutive latent-action pairs for future consistency. The soft-actor-critic module utilizes the latent feature vector to make driving decisions for downstream planning and control tasks. Experiments are carried out in both hyper-realistic and interactive driving environments.

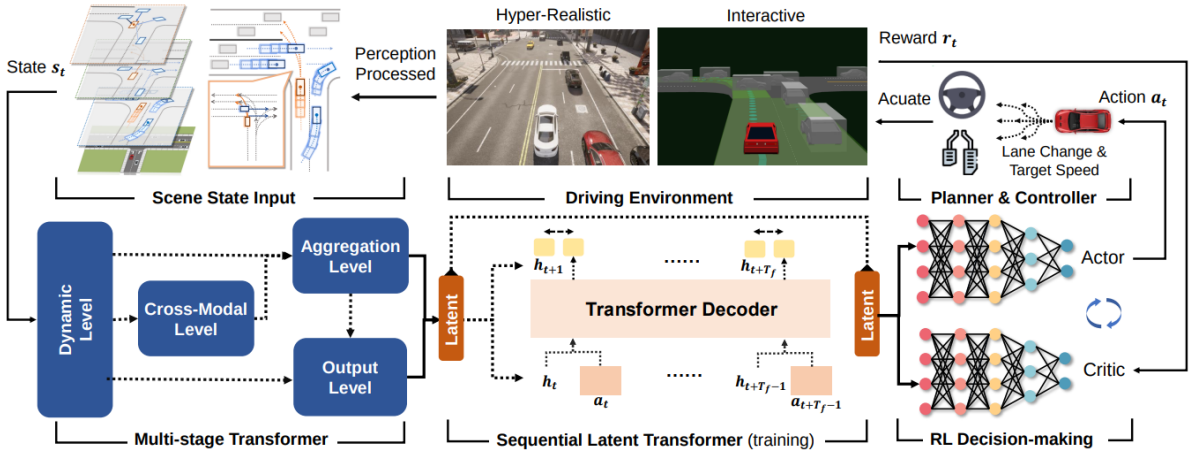


Figure 3.1: An overview of the framework with Scene-Rep Transformer. *Source: Augmenting Reinforcement Learning with Transformer-based Scene Representation Learning for Decision-making of Autonomous Driving [85] (2023).*

This work addresses three distinct urban scenarios independently but fails to offer a framework capable of navigating through these scenarios in a concatenated manner. The results presented in this work predominantly focus on RL metrics, lacking detailed information on low-level signals such as vehicle velocity or steering angles. Additionally, there is a notable absence of metrics related to comfort and navigation. While the methodology and results are presented clearly, the work is exclusively conducted in simulation environments, with no progression towards real-world implementation. This suggests a primary emphasis on contributions to DRL rather than advancements in AD systems.

3.4.2. Attention-based Deep Reinforcement Learning

Another trend is attention-based architectures [86]–[88]. Specifically, the authors of [88] propose an attention-based driving policy for managing unprotected intersections, employing DRL. This proposal places a greater emphasis on realistic implementation within an AD architecture. The general architecture is presented in Figure 3.2. 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 of generating the throttle

and brake signals from this target velocity. A representation of the different scenarios addressed by this work is presented in Figure 3.3.

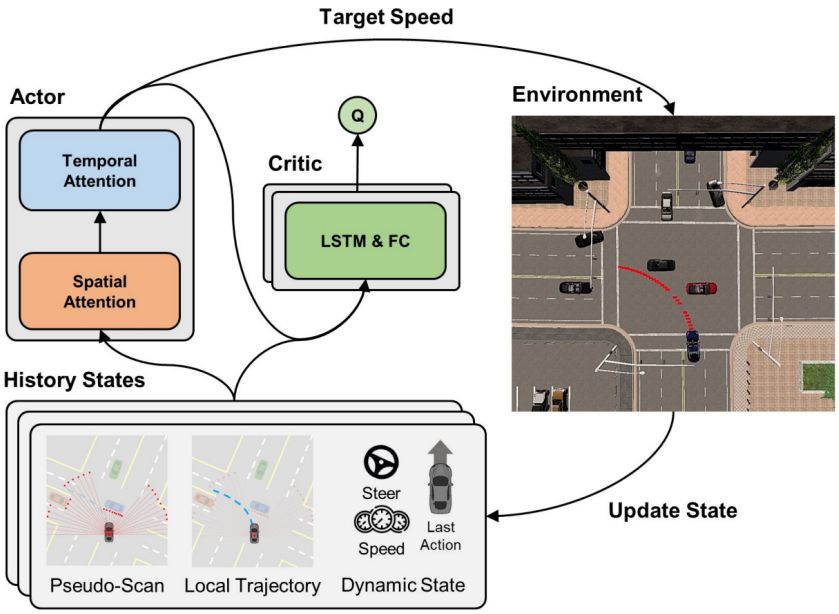


Figure 3.2: A representation of the paper proposed framework, which includes a policy model composed of spatial and temporal attention modules. *Source: Learning to Drive at Unsignalized Intersections using Attention-based Deep Reinforcement Learning [88] (2021).*

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.

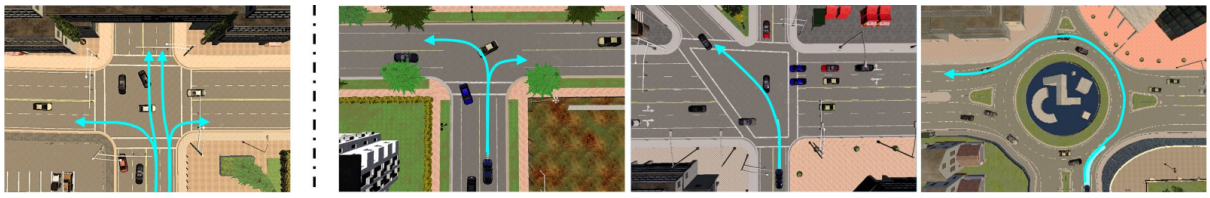


Figure 3.3: Scenarios addressed by this work. *Source: Learning to Drive at Unsignalized Intersections using Attention-based Deep Reinforcement Learning [88] (2021).*

The study demonstrates good results in terms of success rate primarily focusing on various intersection scenarios. However, velocity profiles, as depicted in Figure 3.4, indicate the presence of abrupt control commands, leading to significant fluctuations in the target velocities. Notably, the research lacks comfort metrics to assess the smoothness and practical applicability of the proposed implementation.

They present an implementation on a real vehicle platform, reinforcing the concept that effective simulation can lead to practical application. Nevertheless, the traffic flow in this real-world experiment is quite minimal, involving only two vehicles. As a solution to the challenges associated with conducting experiments in real traffic, this thesis proposes PE as an innovative alternative.

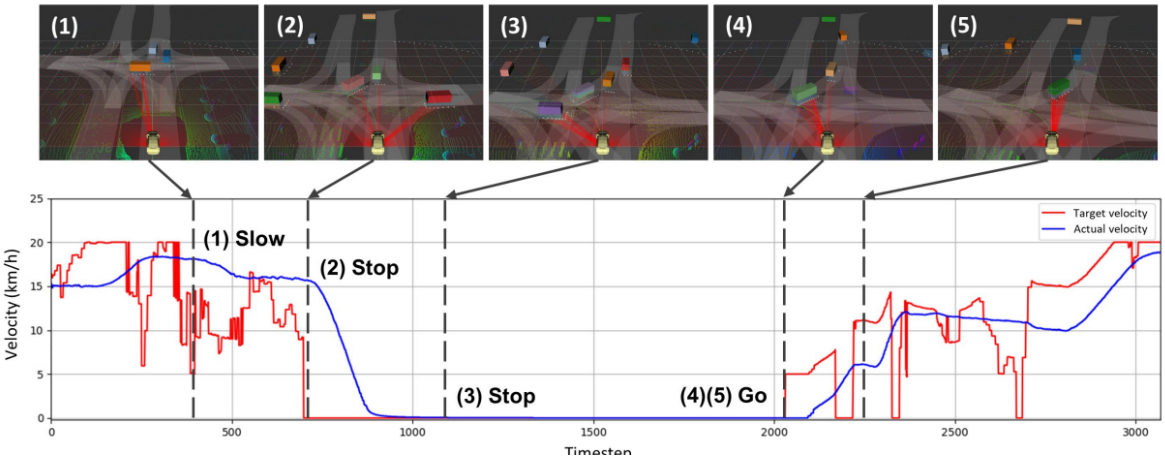


Figure 3.4: Recorded trajectories of the target velocity and actual ego vehicle velocity during an unprotected left turn at the 4-way intersection. *Source: Learning to Drive at Unsignalized Intersections using Attention-based Deep Reinforcement Learning [88] (2021).*

3.4.3. 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 [89], [90]. The authors in [91] present a practical implementation, with emphasis on trajectory generation. Figure 3.5 illustrates the specific scenario addressed and the developed framework.

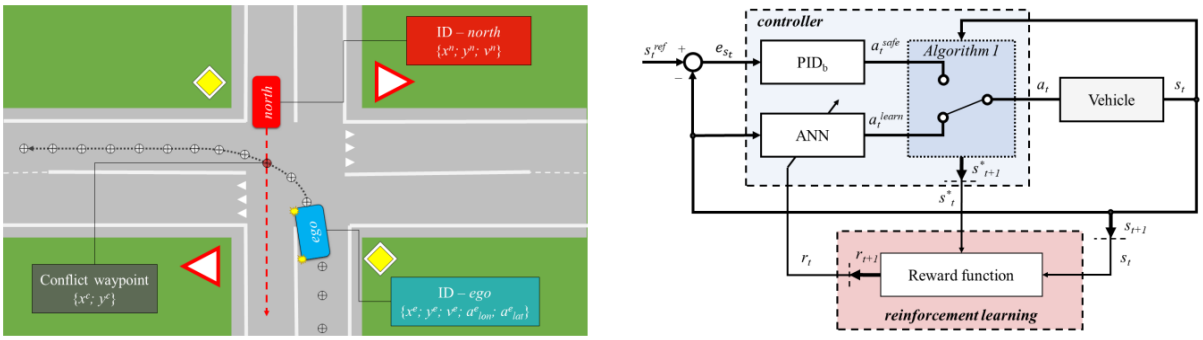


Figure 3.5: A representation of the paper scenario and the proposed framework are represented. *Source: A Safety-Critical Decision Making and Control Framework Combining Machine Learning and Rule-based Algorithms [91] (2022).*

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.

3.4.4. Tactical Behaviour Planning

Other works focus on solving various scenarios using a tactical behaviour planning [92], [93]. The authors of [94] propose high-level DM for AD. They formulate the problem as a continuous POMDP, combining the advantages of two SOTA solvers Monte Carlo Value Iteration and Successive Approximations of the Reachable Space under Optimal Policies. They address a variety of concatenated scenarios, which are presented in Figure 3.6.

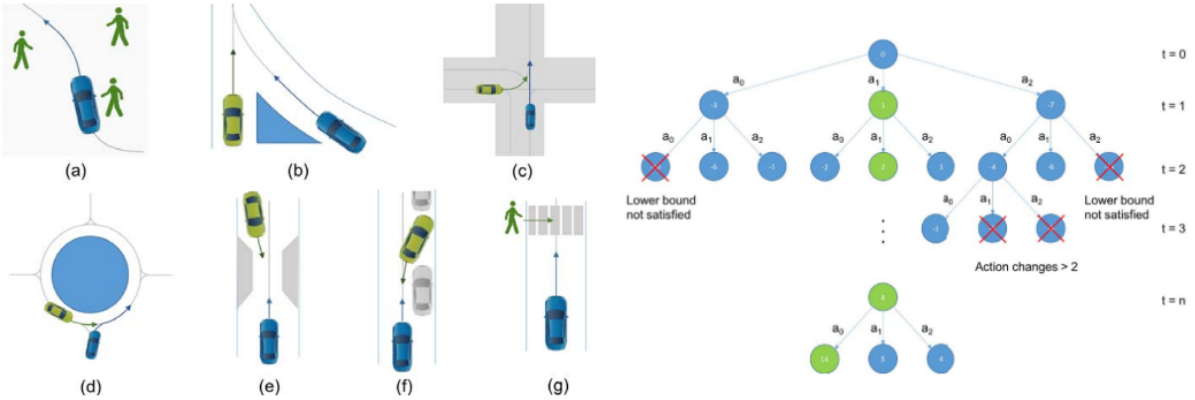


Figure 3.6: Multiple scenarios addressed by this work and Decision Tree. *Source: Towards tactical behaviour planning under uncertainties for automated vehicles in urban scenarios [94] (2017).*

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 is illustrated in the graph shown in Figure 3.7, demonstrating 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.

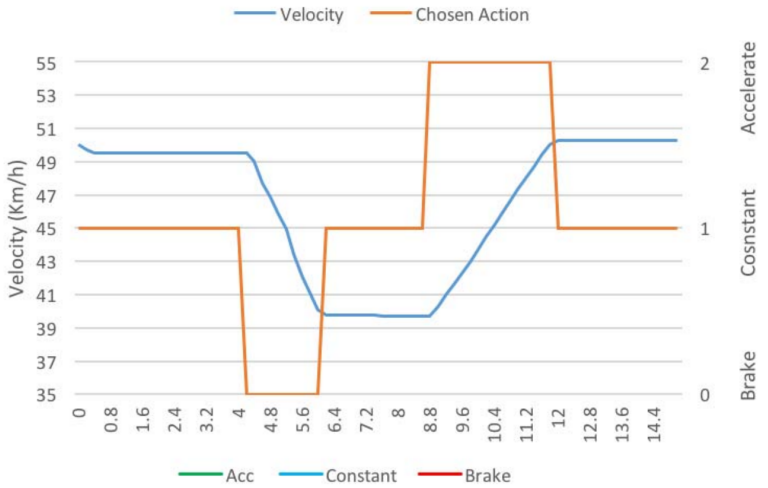


Figure 3.7: Vehicle velocity following the high-level command. *Source: Towards tactical behaviour planning under uncertainties for automated vehicles in urban scenarios [94] (2017).*

This approach shows effective behaviour across various scenarios, but the utilization of these algorithms presents challenges such as high computational costs and scalability issues. These challenges have been addressed in DRL approaches, which offer more efficient solutions in terms of computational resources and scalability, making them more viable for extensive and complex applications. Moreover, this study lacks results related to real applications.

3.4.5. Discussion

The comparative analysis presented in Table 3.2 delineates the distinctive attributes and efficacies of the previously introduced DRL approaches, including Transformer-based, Attention-based, Decision-Control, Tactical Behaviour, and our proposed methodology. This discussion highlights the distinct features and evaluates the overall effectiveness and feasibility of these architectures in the AD context.

- **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.

Table 3.2: Comparison between the Deep Reinforcement Learning Architectures.

Architecture	Transformer-based	Attention-based	Decision-Control	Tactical Behaviour	Ours
Ref	[85]	[88]	[91]	[94]	-
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	×	✓	×	×	✓

In conclusion, our AD architecture presents a significant advancement in the DM field by offering a comprehensive solution that addresses several critical aspects of autonomous vehicle control and navigation. 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. This capability is crucial for real-world applications where diverse driving scenarios are encountered routinely. Moreover, we underscore the practical applicability of our approach through a real-world implementation. This not only validates the effectiveness of our system in actual driving conditions but also demonstrates its application in a real vehicle, a step that many theoretical or simulation-based studies do not reach. 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. In summary, through this innovative integration of technology and practicality we ensure that our proposal is safer, more efficient, and more comfortable than the revised architectures in this section.

3.5. Summary

This chapter explores DM models for AD, comparing classical and learning-based approaches. Classical methods, such as rule-based, optimization, and probabilistic techniques, are noted for their reliability and structured DM but lack adaptability. Learning-based methods, including statistical, DL, RL, DRL, are praised for their ability to handle dynamic environments and learn from data, offering more flexible and human-like capabilities. The chapter reviews existing AD architectures, contrasting various frameworks like

Transformer-based and Attention-based methods with the thesis's approach that combines the strengths of classical control and learning-based DM. Our approach stands out by providing smooth control signals evaluated by comfort metrics, driving in concatenated scenarios, offering real implementation, and managing complex situations with low dimensionality and computational cost, thereby ensuring a safer, more efficient, and comfortable AD system compared to existing literature.

Chapter 4

Hybrid Decision Making Architecture and Simulation Framework

Sólo hay un bien: el conocimiento.
Sólo hay un mal: la ignorancia.

Sócrates

4.1. Introduction

This chapter 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. The detailed exploration of the decision and control modules is reserved for subsequent chapters, where they are discussed in greater depth. The hybrid DM architecture was partially published in the conference IV 2023 [9]: "Hybrid Decision Making for Autonomous Driving in Complex Urban Scenarios".

After introducing the architecture, the chapter shifts focus to a critical aspect in the development of AVs: simulation. This section underscores the importance of simulation for training and validating algorithms and frameworks. It offers a survey of various simulators available in the literature and discusses different AD benchmarks that have been proposed.

Having established the foundational tools and methodologies for training and testing an AD stack, the chapter concludes by presenting the specific framework selected for developing the proposed architecture. This strategic choice sets the groundwork for the practical implementation and evaluation phases of this dissertation.

4.2. Our Architecture Approach

As elaborated in Chapter 3, the current SOTA showcases various research directions, including the application of DL-based methods as evidenced by studies such as [83], [84], [87], the utilization of traditional algorithms as seen in works by [92]–[94], and the formulation of hybrid approaches as demonstrated by [89]–[91]. Our architecture has been inspired by and developed in line with these hybrid proposals. In this way, this thesis 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.

The architecture of our system is structured into four levels, as illustrated in Figure 4.1. 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 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 thesis contributes mainly at the tactical (DM module) and operative (control module) levels, with a small contribution at the strategy level (planning module).

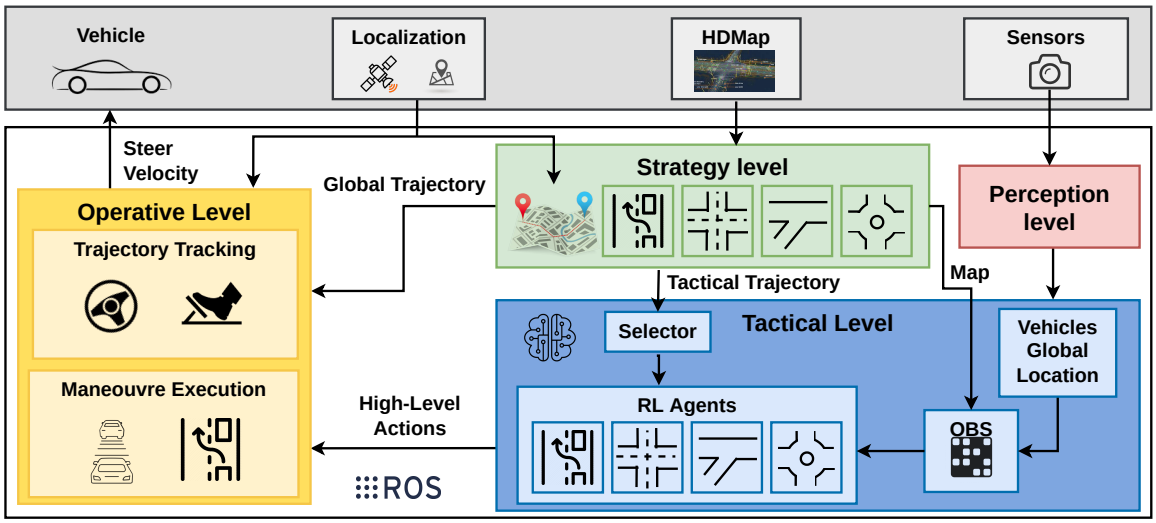


Figure 4.1: The proposed hybrid architecture. The strategy level defines a tactical trajectory with the map information and the ego vehicle location. The tactical level executes high-level actions in correlation with the perception ground truth. The operative level combines the trajectory and the actions, calculating the driving commands.

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

4.2.1. 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 4.2.

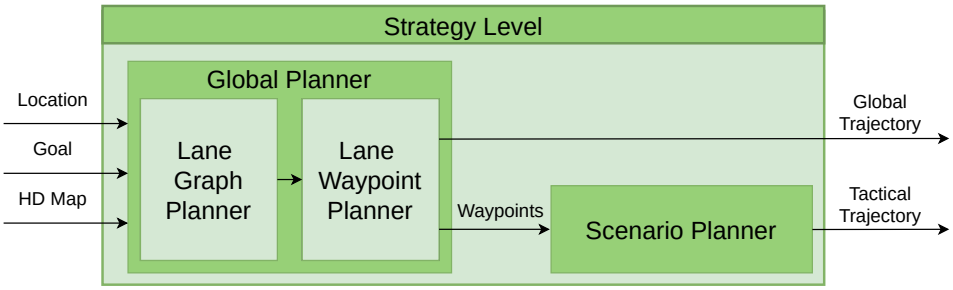


Figure 4.2: Strategy level. The global planner calculates the global trajectory while the scenario planner generates the tactical trajectory.

The strategy level is based on prior research by the Robesafe group [95], presented at the IV 2022 conference, where the global planner module was developed. For this thesis, 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 route between them as a list of waypoints centred in the lanes. 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.

- **Lane Graph Planner:** LGP creates a directed graph (DiGraph) of roads and lanes from the HD map. The graph is constructed using edges that represent connections between nodes, where a node is a tuple of road ID and lane ID. Each edge in the graph is a Python set containing the input node, output node, and weight. The weight signifies the cost of travelling from the input node to the output node, represented in road lengths in meters or travel time in seconds, calculated using the maximum speed allowed for each segment. The LGP assesses connections for every road/lane from the HD map. Once the road graph is built, the route is calculated using the Dijkstra

algorithm. The output is a topological route as a list of tuples: (road, lane, action), where action can be: lane follow, lane change right, or lane change left. Although this data could be input to a perception-based controller, our implementation requires a detailed list of waypoints.

- **Lane Waypoint Planner:** LWP is responsible for calculating a list of waypoints for each road-lane determined by the LGP. It utilizes road-lane geometries from the HD map and a mathematical discretization process to define the waypoints centred in the lane. The waypoints are spaced apart by a predetermined distance, aligning with the route calculated by the LWP.

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.).

An illustration of the strategy level in CARLA is depicted in Figure 4.3, where a comprehensive method for working with OpenDRIVE-based HD maps is presented. This method encompasses map generation, parsing the map file, and finally leveraging all this information for map monitoring and path planning.

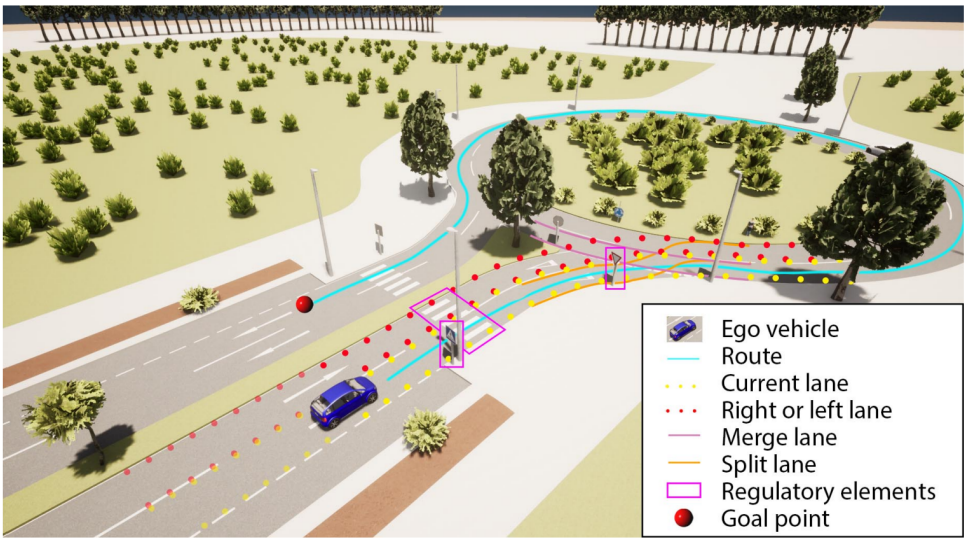


Figure 4.3: Monitored Area and Path Planning in CARLA simulator

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.

4.2.2. Tactical Level

The tactical level is responsible for processing and evaluating information received from other levels to make decisions. The inputs for the tactical 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 other 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. The decisions made at the tactical level are then executed by the operative level. An overview of the tactical level is illustrated in Figure 4.4.

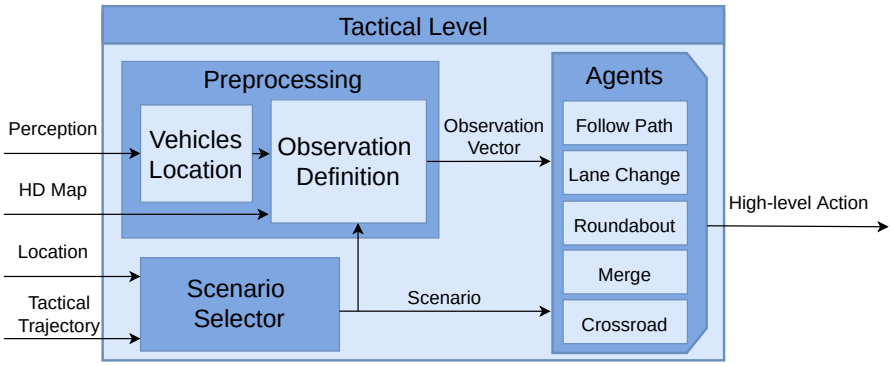


Figure 4.4: Tactical level. Perception data is processed in conjunction with the HD map to formulate the observation vector. The system selects a specific scenario based on the tactical trajectory and the current location. A decision module is selected, which is responsible of executing decisions.

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 an 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 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). This agent then takes actions based on the corresponding observation vector. A detailed description of the functioning of these agents is provided in Chapter 6.

4.2.3. 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. The second task is executed using MPC controllers to incorporate lateral and longitudinal constrains, ensuring smooth movements during manoeuvres. This section introduces the architecture of the operative level, as illustrated in Figure 4.5. The diagram highlights the interaction among the two controllers within this level.

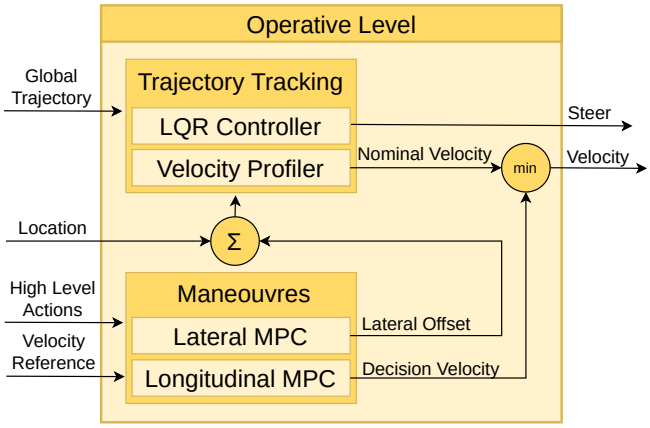


Figure 4.5: Operative level. The global trajectory is meticulously followed using a Linear Quadratic Regulator controller. Additionally, manoeuvres are executed with the aid of a Model Predictive Control system.

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, 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.

A detailed description of the operation of these controllers is provided in Chapter 5.

4.3. Simulation Frameworks

This section justifies the need of a simulator for validating RL-based frameworks. The final goal is to build a realistic environment for AD development. This environment serves as an initial step towards developing approaches that can be integrated into real-world implementations, providing a safe way of training various algorithms.

More specifically, this section delves into three main topics: the principal AD simulation platforms identified in the literature, the various AD simulated benchmarks, and the prospective strides in simulation towards real-world implementation.

4.3.1. Autonomous Driving Simulators

The careful design of driving scenarios is crucial for developing safe and robust AD technology. Contemporary simulators have evolved beyond mere vehicle dynamics simulation to encompass a wide range of complex functionalities, including sensor models, path planning, control systems, and more. In order to meet the testing needs of advanced AD technology, simulators must fulfil a comprehensive set of requirements, extending from simulating physical car models to various sensor models and beyond. The following SOTA simulators exemplify this trend.

CarSim [96] is a widely-recognized vehicle simulator, popular in both academic and industrial spheres. The latest version of CarSim has enhanced features supporting the simulation of moving objects and sensors, which are crucial for scenarios involving self-driving technology and ADAS. An interesting aspect of CarSim is its capability to link these moving objects to 3D models, including embedded animations. This feature is especially useful for simulating dynamic entities such as vehicles, cyclists, and pedestrians.

Another notable simulator is **PreScan** [97], which offers a comprehensive framework designed for the development of self-driving cars and ADAS. A key feature of PreScan

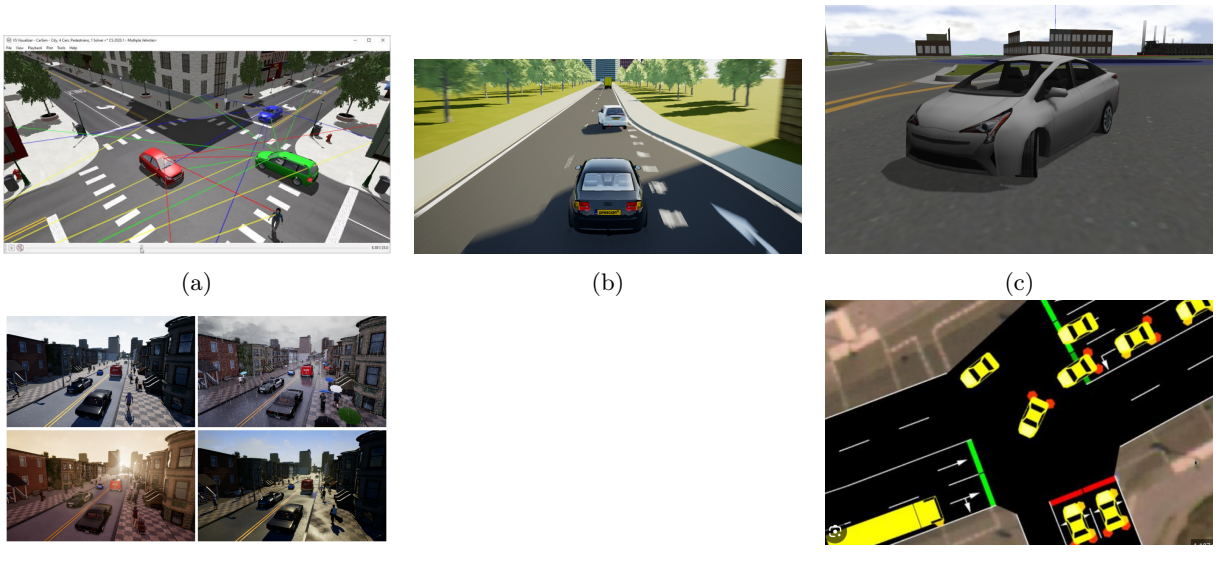
is its automatic traffic generator, which allows manufacturers to test and validate their autonomous navigation systems under a variety of realistic environmental and traffic conditions. Moreover, PreScan supports Hardware-in-the-Loop simulation, a methodology extensively employed for evaluating Electronic Control Units in real-world applications.

CARLA, an open-source autonomous driving simulator, is implemented as a layer over Unreal Engine 4. CARLA benefits from UE4's ecosystem of interoperable plug-ins, realistic physics, and state-of-the-art image quality. Designed as a server-client system, CARLA leverages UE4's capabilities to render and run simulations on the server. Its environment includes 3D models of both static (buildings, infrastructure, vegetation) and dynamic (pedestrians, cyclists, vehicles) objects. These models use low-weight geometric textures while maintaining visual realism through variable levels of detail and meticulously crafted materials. A key advantage of CARLA is the ease of modifying vehicle onboard sensors and their features for accurate data collection, alongside the ability to create realistic traffic scenarios.

Another notable simulator is **Gazebo** [98], a scalable, open-source, flexible, and multi-robot 3D simulator. Gazebo excels in simulating both outdoor and indoor environments. It defines its 3D scene (world) and any 3D object (model) within it, using a Simulation Description File. The default physics engine in Gazebo is the Open Dynamic Engine.

SUMO is an open-source, highly portable, microscopic and continuous traffic simulation package designed to handle large road networks. It allows for intermodal simulation including pedestrians and comes with a large set of tools for scenario creation.

An overview of the simulators discussed in this section is presented in Figure 4.6.



(a) (b) (c)

(d) (e)

Figure 4.6: An overview of the simulation platforms. (**a**) CarSim Simulator. (**b**) PreScan Simulator. (**c**) Gazebo Simulator. (**d**) CARLA Simulator. (**e**) SUMO Simulator.

Selecting the right simulator involves considering various criteria, such as perception (sensors and weather conditions), multi-view geometry, traffic infrastructure, vehicle con-

trol, traffic scenario simulation, 3D virtual environment, 2D/3D ground truth, scalability via a server multi-client architecture, and open-source availability. A comprehensive comparison of these simulators is provided in Table 4.1.

Table 4.1: Comparison of some *state-of-the-art* simulators for Autonomous Driving. GT stands for Ground-Truth. ✓and × indicate that the corresponding requirement is supported or not respectively. U = Unknown, TL = Traffic Light, SS = Stop Signal, INT = Intersections, IN = Indoor, OUT = Outdoor, ROS = Robot Operating System. MCA = Multi-Client Architecture. Hyphens "-" indicate that attributes are either not applicable, or not available.

Requirements	**CarSim**	**PreScan**	**CARLA**	**Gazebo**	**SUMO**
Sensor models supported	✓	✓	✓	✓	×
Different weather conditions	×	✓	✓	×	×
Camera Calibration	×	✓	✓	×	×
Path Planning	✓	✓	✓	✓	✓
Proper vehicle control dynamics	✓	✓	✓	✓	×
3D Virtual Environment	✓	✓	✓, OUT (Urban)	✓, IN & OUT	×
Traffic Infrastructure	✓	✓	✓ (including TLs, INTs, SSs)	✓	✓ (including TLs, INTs, SSs)
Simulate different dynamic objects:	✓	✓	✓	×	×
2D/3D ground-truth	×	×	✓	-	✓
Interfaces to other software	✓, with MATLAB	✓, with MATLAB	✓, with ROS, Autoware	✓, with ROS	✓, with CARLA
Scalability via a server MCA	-	-	✓	✓	-
Open Source	×	×	✓	✓	✓
Stability	✓	✓	✓	✓	✓
Portability	✓	✓	✓, Windows & Linux	✓, Windows & Linux	✓, Windows & Linux
Flexible API	✓	-	✓	✓	✓

In our research, we opted for open-source simulators to explore AD scenarios. We selected CARLA for its high degree of realism and popularity in the AD research community, making it an ideal platform for implementing our system. However, due to CARLA's significant computational demands, we also choose SUMO for developing our approach and conducting the preliminary training phase of our DRL algorithms. This dual-simulator strategy allows us to leverage the strengths of both platforms: the realistic simulation environment of CARLA and the efficiency and scalability of SUMO.

4.3.2. Autonomous Driving Simulated Benchmarks

In the scientific community, these simulators are well-recognized, and various research groups have introduced different benchmarks for validation by other scientists. Such practices are prevalent in supervised learning research. However, in the realm of RL, the situation is more complex. Researchers often design unique experiments with varying

simulation setups, making the translation of these works to standard benchmarks less common.

In urban scenarios, the application of DRL encounters a unique set of challenges, prominently due to the intricate and dynamic nature of these environments. Urban landscapes are characterized by a variety of traffic situations, each demanding different DM strategies. Among the most common scenarios where DRL plays a crucial role are navigating roundabouts, managing crossroads, executing merging manoeuvres, and performing lane changes. In roundabouts, the primary challenge lies in the continuous and unpredictable nature of traffic flow, requiring sophisticated algorithms to anticipate and react to other vehicles' movements efficiently [99]. At crossroads, the complexity is amplified by the need to comply with traffic signals while also considering the actions of other vehicles [75]. Merging scenarios, particularly in high-traffic urban settings, demand a high level of precision in timing and trajectory planning to ensure safety and fluidity of traffic [100]. Lastly, lane changes in urban environments need a keen awareness of the surrounding vehicular density and the ability to make quick, yet safe decisions [101]. These urban driving scenarios are exemplary of the areas where DRL can significantly enhance the capabilities of AVs, contributing to safer and more efficient urban mobility.

The fidelity of the simulation environment is paramount, as more realistic simulations tend to lead to more effective outcomes. While certain studies develop their own simulation setups [75], a trend towards standardization is observable, with some researchers utilizing established simulators like SUMO and CARLA to develop frameworks and algorithms [85].

Recently, several proposals for testing autonomous driving systems have emerged. In this thesis, we review these approaches in relation to the simulation frameworks we utilize, specifically focusing on the following benchmarks:

- **HighwayEnv** [102]: Although is not strictly a benchmark, it is well-known within the RL community. It comprises a collection of environments for AD and tactical DM tasks. This framework allows the validation of RL algorithms.

- **SMARTS** [12]: This benchmark aims to assess the adaptability and robustness of RL agents in realistic and varied interactions. It features authentic traffic models and extensive scenario coverage. However, it lacks vehicle dynamics and does not support the integration of RL agents with other modules of an AD stack.

- **CARLA Challenge** [103]: This benchmark concentrates on urban driving, offering a range of intricate scenarios. Its high-fidelity graphics and elaborate urban layouts contribute to more realistic testing conditions. The primary goal of this challenge is to validate an holistic AD system, which means it is not specifically designed for RL applications. One notable limitation is the lack of uncontrolled traffic situations.

- **CARLA No Crash** [104]: This benchmark primarily targets driving in urban environments, navigating from one point to another. Its main objective is to validate control systems within CARLA, rather than focusing on DM. While it holds potential, it has not gained significant prominence since its inception. A key limitation is its restricted range of scenarios, coupled with traffic that is randomly generated.

Each of these benchmarks offers unique insights and challenges for RL-based AD systems, highlighting the need for diverse and comprehensive testing approaches. A comparison of these benchmarks is provided in Table 4.2.

Table 4.2: Comparison of some *state-of-the-art* benchmarks for Autonomous Driving. ✓and × indicate that the corresponding requirement is supported or not respectively.

Requirements	**HighwayEnv**	**SMARTS**	**CARLA Challenge**	**CARLA No Crash**
Simulator	Gym	SUMO	CARLA	CARLA
Designed for RL	✓	✓	×	×
Variaty of Scenarios	✓	✓	✓	×
Proper vehicle control dynamics	×	×	✓	✓
Leaderboard	×	×	✓	×

Standardized environments, such as HighwayEnv and SMARTS, offer diverse test scenarios and are compatible with the OpenAI Gym interface, a standard in the community to develop RL frameworks. These standardized frameworks often prioritize RL mechanics over realism, which can be a limitation for realistic AD applications. However, in order to validate our proposal in an open source environment we chose SMARTS, as it is based on sumo and we found several works tested in this benchmark.

4.3.3. Simulation to Real World

The primary goal of simulating AD systems is to speed up the development of models within a versatile and controlled setting, serving as an initial step towards their real-world deployment. Safety and cost are critical factors in the advancement of AD technologies, covering the spectrum from academic research to commercial usage. Simulation followed by real-world adaptation and testing is crucial for the development of AVs. In practice, a comprehensive simulation phase precedes the application of acquired driving knowledge in actual environments. Yet, the transition from simulation to real-world conditions is affected by the Reality Gap (RG), a term that encapsulates the variances in lighting, textures, vehicle dynamics, sensors, and agent behaviours. To address the RG, recent studies [105] have ventured into various methodologiesas: sim2real, DT, and Parallel Intelligence (PI). This section explores these strategies, emphasizing their contributions, breakthroughs and main concepts in the domain of AD.

- **Sim2real**: This is a crucial process in AD, aiming to bridge the RG by transferring strategies and knowledge from simulations to real-world applications. This involves

training autonomous systems in simulations and applying them to actual vehicles. To tackle RG challenges, such as uneven sampling and imperfect dynamics models, sim2real employs various methods like curriculum learning, meta-learning, knowledge distillation, robust RL, and transfer learning, supplemented by domain randomization and system identification. Despite its effectiveness, sim2real faces significant computational cost challenges, particularly in complex and dynamic environments, limiting its scalability.

- **Digital Twins (DT)**: These methods focus on creating a simulation environment that maps real-world physical entities, utilizing sensor data and physical models to reflect their entire lifecycle. DT are used for multi-scale environment and vehicle modelling. Utilizing virtual representations of the real environment allow real vehicles to learn knowledge of their DT by synchronizing data from both the real and simulated worlds in an offline way.
- **Parallel Intelligence (PI)**: This combines the strengths of both sim2real and DT methods for improved management and control in complex systems. The learned knowledge is applied to the real vehicle through PE with real-time interaction between the real and virtual worlds and online feedback.

4.4. Our Simulation Framework

Following the introduction of various simulation platforms and a discussion about the significance of simulation in RL, this section delineates the tools employed in this thesis for our research development. As previously noted, two simulators, SUMO and CARLA, are utilized in a Curriculum Learning (CL) process. To facilitate a seamless domain adaptation of our algorithms, we have implemented an interface module with simulators. This module is designed to extract information from the simulator, thereby providing a state vector to the DRL agent and enabling the execution of the DRL agent's actions within the simulation. This interface module is constructed utilizing the standard libraries provided by the two simulation environments. Once the simulation setup is established, an agent can be trained, thus forming the DM module, employing standard libraries widely recognized in the community. Finally, we offer an overview of our proposal with DT and PI approaches to diminish the disparity between simulations and real-world applications.

4.4.1. Simulation of Urban MObility (SUMO)

SUMO is an open-source, detailed, microscopic, multi-modal traffic simulator designed to model individual vehicle movements through urban road networks. It stands out for its ability to simulate diverse traffic scenarios, supporting deterministic and random elements. Alongside the core simulation engine, SUMO includes a suite of tools for importing

and preparing road networks and traffic data. Its capabilities extend to space-continuous, time-discrete vehicle movement, various vehicle types, multi-lane streets, and dynamic traffic management features like different right-of-way rules and traffic lights. The simulator is notable for its fast execution speed and graphical user interface, handling extensive networks efficiently. The simulator's interface is illustrated in Figure 4.7. This visual representation provides an overview of the simulation environment, showcasing the road networks, traffic elements, and other relevant features to the simulation process.

Figure 4.7: An overview of SUMO simulator.

We leverage the fast execution capabilities of the simulation to accelerate the training process, a critical aspect in DRL. To train and validate the DRL algorithms, different scenarios are generated using the user interface. As previously discussed, an essential component for this purpose is the interface module, which is crafted using Traffic Control Interface (TraCI). TraCI is a sophisticated interface designed for interacting with ongoing road traffic simulations. It enables users to access and modify the behaviour of simulated objects in real-time. Operating on a TCP-based client/server architecture, TraCI allows for seamless connection to SUMO simulations. The simulation setup, as depicted in Figure 4.8, enables the DRL agent to access information from the simulation through TraCI and execute actions accordingly. This setup allows a direct interaction between the DRL agent and the simulation environment.

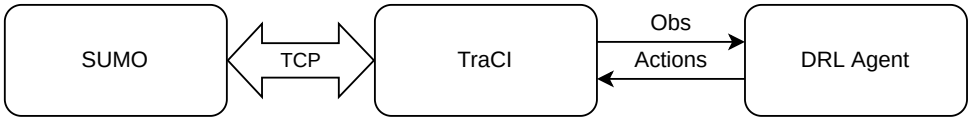


Figure 4.8: The interface module between SUMO simulator and the DRL agent.

With the establishment of a real-time interactive simulation environment, the next step is to create a realistic scenario. The construction of the road infrastructure is facilitated by a tool called netedit. Netedit provides the flexibility to design various aspects of the roads, such as their shape, priorities, and lane velocities. Once the road layout is established, the behaviour of other participants in the simulation needs to be defined.

This is primarily achieved by setting predefined routes for them in the scenario configuration. Furthermore, during the simulation, the behaviours of traffic participants can be dynamically refined using TraCI. This enables the creation of detailed and realistic traffic scenarios, encompassing aspects such as the cooperativeness of drivers, their velocities, lane changing behaviours, and path adherence.

In the experiments presented in Chapter 6, SUMO is responsible for controlling the movements of both the ego vehicle and the adversarial vehicles, encompassing trajectory tracking and manoeuvre execution. In this simulation environment, these movements do not account for the vehicles' dynamics. These vehicles in SUMO follows the model defined by Krauss in [106].

4.4.2. Car Learning to Act (CARLA)

CARLA is a versatile, open-source simulation platform tailored for the development, training, and validation of AD systems. The simulator's interface, as illustrated in Fig 4.9, vividly demonstrates the realism inherent in the CARLA simulation environment.

Figure 4.9: An overview of CARLA simulator.

It provides not only the source code and protocols but also a variety of digital assets such as urban layouts, buildings, and vehicles, all designed specifically for AD research. The platform offers remarkable features, including scalable server multi-client architecture, a flexible API for comprehensive simulation control, and a diverse suite of AD sensors. In contrast with SUMO, which prioritizes speed and efficiency in its simulations, CARLA's simulations tend to be slower due to the complexity of their agent. This complexity allows CARLA to present highly realistic vehicle dynamics, enabling simulations that closely mimic real-world scenarios. This focus on realism is particularly beneficial

for applications requiring detailed and accurate modelling of vehicle behaviours and interactions within a dynamically changing environment. In CARLA, vehicle dynamics are calculated via the "PhysX vehicle SDK" plug-in [107]. This method models vehicles as a collection of sprung masses, each of which incorporates a suspension line and corresponding wheel and tyre specifications. These assemblies are further aligned with a rigid body actor, meticulously reflecting the exact mass, centre of mass and moment of inertia of the individual sprung masses. PhysX vehicles support various drive models, centred on a torque clutch that links the engine to the wheels and enables realistic torque transfer. This model reflects the interaction of the vehicle with the environment and other objects within the PhysX simulation, ensuring accurate and dynamic vehicle behaviour. Some of the parameters, such as mass, maximum torque, maximum rpm, tyre friction, etc., can be modified to create a specific model of the vehicle.

In a similar vein to SUMO, CARLA provides a specialized tool for accessing simulation data, known as the CARLA PythonAPI. This API allows for an intuitive and efficient way of interfacing with the simulation environment. Through the PythonAPI, users can programmatically control various aspects of the simulation, including vehicle behaviours, sensor data collection, and environmental settings. Additionally, CARLA includes a feature called the Traffic Manager (TM). This component is crucial for managing the dynamic of traffic within the simulation. The TM allows for the customization of traffic behaviour, controlling parameters such as vehicle speed, route planning, and collision avoidance. It provides a way to simulate more realistic traffic conditions, making it possible to test AD algorithms under various, often challenging, traffic scenarios. The simulation setup is presented in Figure 4.10.

Figure 4.10: The interface module between CARLA simulator and the DRL agent.

CARLA encompasses a variety of virtual environments, known as "towns," each with unique traits to support a wide range of AD research scenarios. In this dissertation, we leverage these "towns" as the foundational virtual spaces for constructing our RL environments, specifically focusing on two towns:

- **Town03:** Designed to mimic the essence of a European city, *Town03* combines historical and modern architectural elements. Its configuration features narrow streets, roundabouts and diverse intersections, offering intricate urban driving challenges.
- **Town04:** Contrarily, *Town04* portrays a suburban setting with residential neighbourhoods, parks, and commercial areas. It is characterized by broader roads and varied intersection types compared to *Town03*, thereby simulating suburban driving scenarios.

In Chapter 7 we detail the use of PythonAPI, TM, and map information to define the behaviour of other simulation participants.

4.4.3. OpenAI Gym and Stable Baselines 3 (SB3)

The implementation of DRL setup uses two standard frameworks: OpenAI Gym and Stable Baselines3 (SB3). Gym, established as a benchmark in the field of RL, is widely recognized for its role in creating diverse and standardized RL environments. SB3 complements this tool by offering efficient implementations of SOTA algorithms.

In this thesis, we leverage Gym's ability to create custom environments to develop simulations specifically for AD, utilizing its versatile framework for targeted DRL models. Two critical concepts in Gym are the action space and observation space. The action space defines the set of actions an agent can take in the environment. It can be discrete or continuous. The observation space, on the other hand, represents the information available to the agent at each step of the environment. It includes the data that the agent perceives and uses to make decisions, which can range from pixel data in a visual environment to sensor readings in a robotic control task. Essential functions in Gym include step and reset. The step function advances the environment by one time step based on an action taken by the agent,returns the new state, the reward and the done signal. The reset function is used to initialize or restart the environment, typically returning the initial state. In the upcoming chapters, we will delve into defining the specific functions that are instrumental in generating our custom environments.

SB3 is a library that builds upon and complements Gym. While Gym provides a standardized framework for creating and managing different environments, SB3 offers efficient and robust implementations of various advanced DRL algorithms that are compatible with these environments.

The relationship between SB3 and Gym is synergistic. Gym's environments serve as the testing and training grounds for RL agents, and SB3 provides the algorithmic backbone, enabling these agents to learn and interact within these environments. SB3's algorithms are fine-tuned to work seamlessly with the Gym API, ensuring smooth integration and ease of use.

4.4.4. Reducing Reality Gap

To bridge the RG, we introduce a novel approach for the practical application of a DM module within our AD stack, based on a CL strategy. This method is structured into three crucial phases: First, training of the DRL agent to form an initial behaviour model withput dynamics in SUMO. Subsequently, this model undergoes further refinement during the second training phase involving the use of a DT of our actual environment and vehicle including dynamics in CARLA. The culminating phase entails the real-world

validation of our AD stack, augmented by virtual perception through a PE approach. In this setup, simulated and real-world tests are conducted simultaneously in real-time, enabling interaction with virtual adversarial vehicles while moving our real vehicle. A schematic illustration of our PI proposal, which includes a PE of the actual and simulated models, is provided in Figure 4.11.

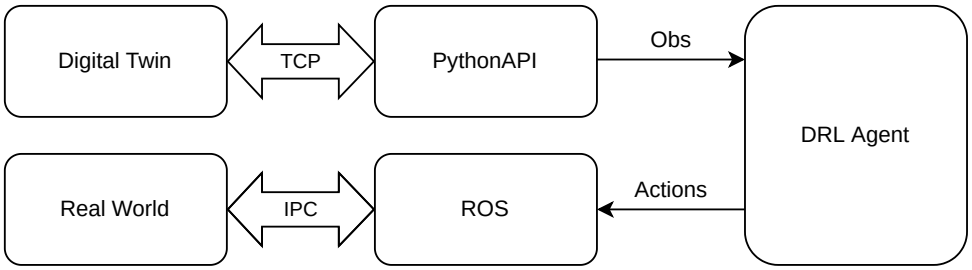


Figure 4.11: An overview of our proposed Parallel Intelligence.

The scenario and the adversarial vehicles are generated using the CARLA PythonAPI. Observations are extracted from the DT by the DRL agent using this API. Subsequently, actions generated by the agent are transmitted to the real vehicle's actuators. The AD stack modules are implemented in Robot Operating System (ROS) [108] nodes and the communication among them is asynchronous and based on topics, through the Inter-Process Communication (IPC). This implementation is described in more detail in Section 7.4.

4.5. Summary

This chapter outlines the proposed hybrid architecture for AD systems, emphasizing its structure and the interactions among the different levels. It highlights the importance of simulation in AD development, comparing various simulators and benchmarks, and discusses transitioning from simulated environments to real-world applications. The architecture is described across strategic, tactical, and operative levels, focusing on route planning, tactical trajectory creation, and the conversion of high-level decisions into vehicle commands. Additionally, the chapter explores methodologies like Sim2real, DT, and PI to bridge the gap between simulation and reality. Finally, it details the development framework using SUMO and CARLA simulators, and Gym and SB3 tools, showcasing an interface module that enables training and validation of RL algorithms in realistic traffic scenarios, reducing the RG.

Chapter 5

Trajectory Tracking and Manoeuvres Execution

Cuanto mayor es la dificultad, mayor gloria en superarla.

Epicuro

5.1. Introduction

As introduced in Section 4.2.3, the operative level of the proposed architecture is responsible for generating the low-level control signals needed, both to follow the nominal trajectory given as a sequence of waypoints, and to execute the manoeuvres indicated by the tactical level. These control signals are the steering wheel angle and the linear velocity required as inputs for the vehicle's DBW system. Classical optimal control methods have been employed for this level of the hybrid architecture, allowing to meet the necessary requirements of reliability, safety and comfort.

Although in recent years MPC has raised high expectations as an optimal control method for trajectory tracking of on-road vehicles [109], its real-time implementation has not yet been achieved due to the high computational requirements involved in using non-linear models and large time horizons appropriate to driving velocities. For this reason, this thesis proposes the use of a hybrid controller, which combines the passenger's comfort associated with the smoothness of spline curves, the low computing time and reliability of LQR control for nominal trajectory tracking, and a fast linear MPC for safe manoeuvres execution. After generating a smooth trajectory from waypoints using spline interpolation, the LQR control allows generating safe and comfortable control signals in real-time at medium velocities, compensating for delays in perception and actuation systems to predict the vehicle's location at the moment when the control signals take effect. In parallel, an MPC controller executes lane change manoeuvres when necessary, by generating lateral displacement through a simple linear model based on a decoupled

chain of integrators, which allows real-time implementation and ensures compliance with certain constraints to achieve manoeuvres safely. Finally, a velocity profiler adjusts the nominal linear velocity of the vehicle to the characteristics of different segments of the trajectory, primarily the curvature radius. This linear velocity is also modified by the MPC controller based on the velocity of adversarial vehicles and the high level actions executed by the decisions module.

A general vision of the approach is presented in Figure 5.1. The ego vehicle is tracking the spline nominal trajectory (red dashed line) using the LQR tracking controller when other slower vehicle blocks its nominal behaviour. After receiving the lane change order from the tactical level, the MPC controller generates a lateral offset that is added to the input of the LQR controller, achieving a smooth lane change manoeuvre (dashed black line in the figure).

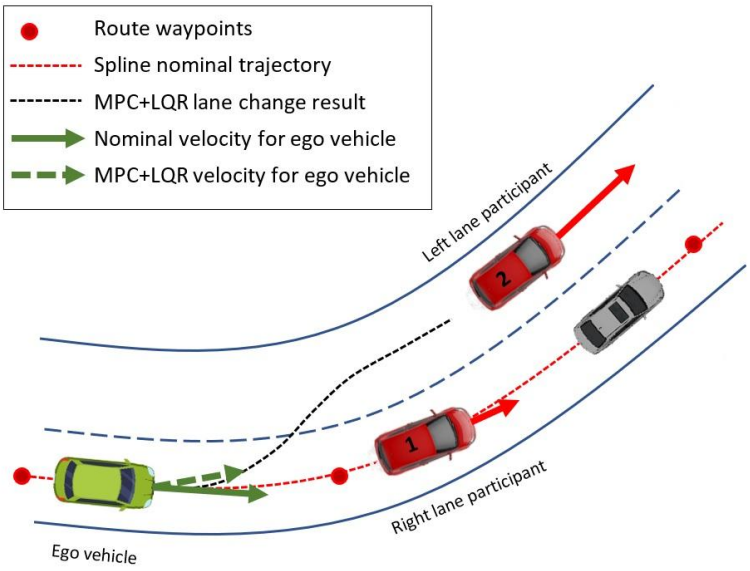


Figure 5.1: Common scenario for trajectory tracking and manoeuvre execution: the ego vehicle is following the nominal trajectory (red line) using LQR control when a slower vehicle blocks its path; for lane changing, a MPC controller generates a lateral offset that, combined with LQR control, generates the final trajectory (black line) for manoeuvre execution.

5.2. Nominal trajectory tracking based on spline curves and LQR control

In this section we describe the controller that allows to generate the control signals for nominal trajectory tracking in order to follow a set of predefined route waypoints, without taking into account the presence of possible obstacles or other vehicles. The work presented in this section was partially published in Sensors journal paper [10], with title "A Waypoint Tracking Controller for Autonomous Road Vehicles Using ROS Framework".

To ensure a smooth and comfortable trajectory tracking a classic controller is utilized, which includes both longitudinal and lateral controllers. The longitudinal controller regulates the vehicle's longitudinal velocity, whereas the lateral controller is responsible for steering the vehicle to ensure path tracking. For paths defined by waypoints, a trajectory interpolator is necessary to produce smooth, optimized movements and continuous actuator references.

The controller architecture is depicted in Figure 5.2. Its inputs include the set of waypoints of the global trajectory, an external velocity specification regarding the maximum velocity of the road segment and the estimated vehicle location. A spline interpolator calculates a smooth trajectory, with its curvature at each point serving as the main input for a linear velocity profiler that adjusts the longitudinal velocity of the vehicle. The lateral control involves an optimal LQR controller that utilizes tracking errors for steering the vehicle. Additionally, a delay compensation system is included to deal with delays introduced by the vehicle's positioning system and the actuators, enabling medium velocities to be achieved without compromising controller stability.

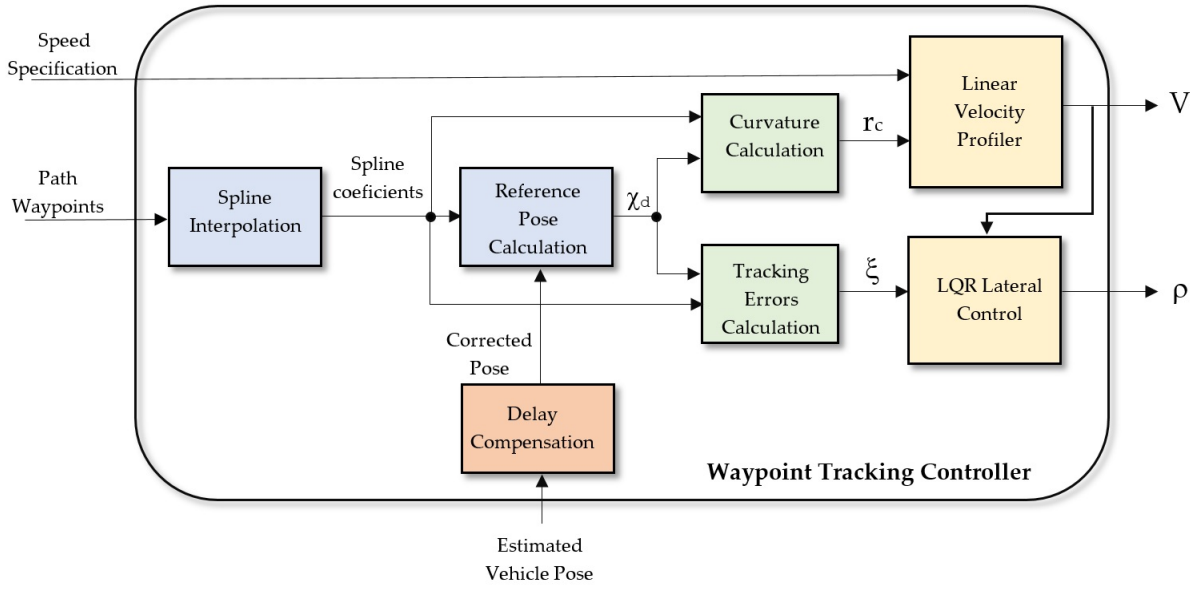


Figure 5.2: Architecture of the Waypoint Tracking Controller. Inputs are: a set of path waypoints, an external speed specification and the estimated vehicle pose. Outputs are: linear velocity V and steering angle ρ. The main subsystems are a Spline Interpolator, a Linear Velocity Profiler, an LQR Lateral Controller and a Delay Compensation System.

5.2.1. Vehicle Model and Limitations

The vehicle model assumes negligible inertia, effective for typical urban driving velocities. A kinematic model, rather than a dynamic one, is utilized for implementing a safe, urban vehicle with limited velocities. The model's inaccuracies are counterbalanced by a feedback control loop for waypoint tracking.

The vehicle's pose (x, y, θ) is defined with respect to an external reference frame and consists of the Cartesian coordinates (x, y) of the central point of the front wheels' axle (control point in Figure 5.3) and the heading angle θ between its longitudinal axis and the external X-axis. The kinematic model facilitates the calculation of the evolution of this pose from the speed $V(t)$ and the angle $\rho(t)$ of a virtual central front wheel at the control point:

$$\begin{aligned} x(t) &= V(t) \cdot \cos(\rho(t) + \theta(t)) \\ y(t) &= V(t) \cdot \sin(\rho(t) + \theta(t)) \\ \theta(t) &= \frac{V(t) \cdot \sin(\rho(t))}{L} \end{aligned} \tag{5.1}$$

with L being the distance between front and rear axles, and steering mechanically limited to $|\rho(t)| < \rho_{\max}$.

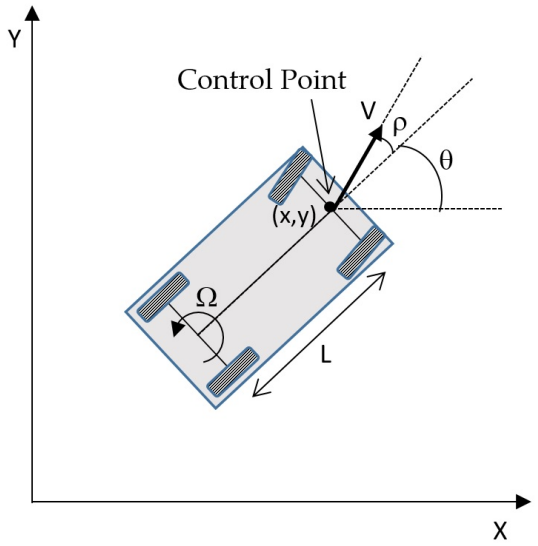


Figure 5.3: Kinematic model for a vehicle with Ackerman configuration: steering centres on the control point, defined as the central point of the front wheels' axle. The kinematic model obtains the evolution of the location of the control point (x, y, θ), as a function of the speed $V(t)$ and the angle $\rho(t)$ of a virtual central front wheel located at the control point.

5.2.2. Waypoints Interpolation

The tracking controller receives a set of path waypoints as input. For a smooth trajectory an interpolation is necessary. In the field of automated driving, different solutions have been proposed. Clothoids have been implemented in the design of highways and railways and are also suitable for car-like robots [110] to obtain smooth transitions between straight segments and curved ones. Polynomial curves [111], [112] have also been used to meet the constraints needed in the waypoints, such as fitting position, angle and curvature constrains among others. Another solution with a low computational cost are the Bézier curves [113], [114], that rely on control points to define their shape. However,

spline curves [115]–[118] have several advantages in the field of autonomous driving, due to their very low computational cost and their high degree of smoothness constraint at the join between the segments of the generated path. In this work we use a spline interpolation, that offers significant advantages in computational efficiency, accuracy, and smoothness. This approach minimizes the curvature of the path, ensuring a trajectory that precisely passes through the input waypoints. In AD, two-dimensional cubic splines guarantee continuity in the first and second derivatives, ensuring smooth angular and linear velocities at joint points.

A two-dimensional parametric cubic spline $Q(U)$ is obtained by combining two splines, $X(U)$ and $Y(U)$, where U is the parameter along the curve. Each spline consists of n segments (cubic polynomials), with $n+1$ waypoints P_i $(i \in [0, n])$ to be joined. To simplify segment handling, U is normalized into $u \in [0, 1]$ for each segment i, resulting in the spline definition $Q(u) = (X_i(u), Y_i(u))$, $i \in [0, n-1]$. The cubic polynomials for each segment are given by:

$$\begin{aligned} X_i(u) &= a_{ix} + b_{ix} \cdot u + c_{ix} \cdot u^2 + d_{ix} \cdot u^3, \quad 0 \leq u \leq 1 \\ Y_i(u) &= a_{iy} + b_{iy} \cdot u + c_{iy} \cdot u^2 + d_{iy} \cdot u^3, \quad 0 \leq u \leq 1 \end{aligned} \tag{5.2}$$

A representation of the cubic splines is depicted in Figure 5.4. The output of the spline interpolator is the set of polynomial coefficients for each path segment. We avoid generating an extensive sequence of points that requires significant computational resources. Instead, we work in a parametric way and we only need to store a few polynomial coefficients because the controller has been formulated in terms of these coefficients.

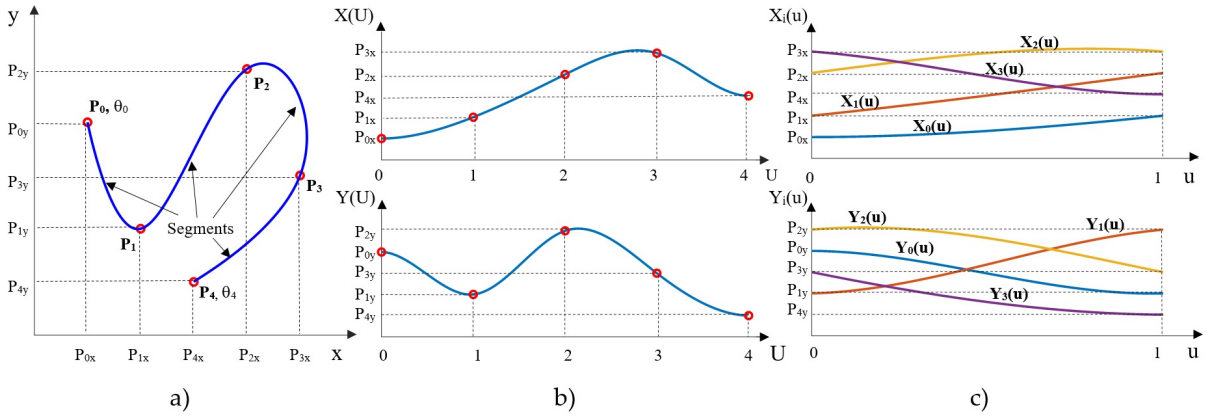


Figure 5.4: Cubic spline definition: a) Cubic spline in cartesian space x-y, with five waypoints (orientation is imposed at start and goal points) and four segments, b) Parametric cubic splines for each coordinate of cartesian space, X(U) and Y(U), c) Polynomials of each segment of the spline with normalized parameter u.

5.2.3. Linear Velocity Profiler and Curvature Calculation

We define a nominal velocity based on the curvature radius of the trajectory to be followed. When dealing with two-dimensional spline curves, the radius of curvature $r_c(u)$ at the point corresponding to a certain value of the parameter u is given by:

$$r_c(u) = \frac{(X'(u)^2 + Y'(u)^2)^{3/2}}{X'(u) \cdot Y''(u) - Y'(u) \cdot X''(u)} \tag{5.3}$$

For road vehicles, it is necessary to reduce velocity in advance, before reaching the segments of high curvature. To achieve this, the velocity profiler first calculates the mean radius of curvature of each segment i of the spline $\overline{rc_i}$. Then, an average speed $\overline{v_i}$ is assigned to each segment of the spline, which is a function of its average radius of curvature:

$$\overline{v_i} = V_{MAX} \cdot \min\left(\frac{\overline{rc_i}}{RC_{MAX}}, 1\right) \tag{5.4}$$

where V_{MAX} is the maximum allowed linear velocity for the vehicle and RC_{MAX} is the maximum value of the radius of curvature that reduces the linear velocity below V_{MAX}. The velocity V_i assigned to each segment of the spline is calculated by averaging the subsequent segment velocities, using a vector λ of normalized weights:

$$V_i = \sum_{n=1}^{N} \overline{v}_{i+n-1} \cdot \lambda(n) \tag{5.5}$$

Finally, a linear interpolator smooths the speed transitions between segments, calculating the velocity command assigned to a value of the parameter u within segment i of the spline in the following way:

$$V_i(u) = \begin{cases} V_i + (u - 0.5) \cdot (V_{i+1} - V_i) & \text{for } 0.5 \leq u \leq 1 \\ V_{i-1} + (u + 0.5) \cdot (V_i - V_{i-1}) & \text{for } 0 \leq u < 0.5 \end{cases} \tag{5.6}$$

5.2.4. Lateral Control

Once the trajectory and linear velocity profile have been defined, a lateral controller provides closed-loop path tracking. Some of the most typical implementations for tracking controllers are are the Pure Pursuit [119] and Stanley [120] controllers. Pure Pursuit is by far the most popular among geometric methods, so it has been a standard benchmark to validate new controllers proposed by researchers [121]. The main shortcoming of Pure Pursuit is the selection of "look-ahead distance" parameter. Dynamic controllers [122]–[124], include the dynamic properties of the vehicles in the control law, but need dynamic feedbacks such as force and torque which require expensive dedicated sensors. Optimal

control [125], [126] has demonstrated to be a suitable choice for robust applications and allows a very intuitive adjustment of its parameters. So, we have implemented an LQR Optimal Controller for its stability characteristics and its easy tuning using cost functions.

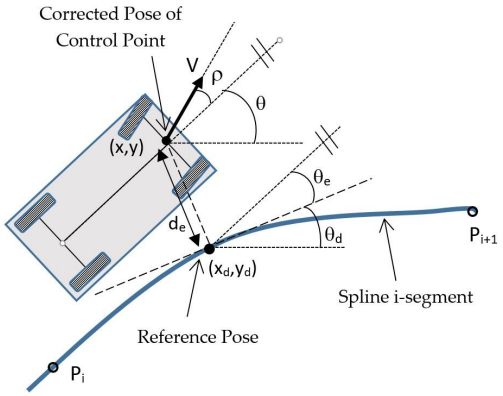


Figure 5.5: Variables involved in lateral control include the reference pose $c_d = (x_d, y_d, \theta_d)$ and the tracking errors, represented by the vector $\mathbf{x} = [d_e \quad \theta_e]^T$.

In order to linearize the non-linear kinematic model of the plant, a convenient selection of the state vector of the controller is needed. The states are chosen to be the trajectory tracking errors $\xi = [d_e \ \theta_e]^T$, defined as the distance from the control point to the closest point of the path (that we call reference pose $\chi_d = (x_d, y_d, \theta_d)$) and the difference between the vehicle heading and the tangent to the path at the reference point (see Figure 5.5), respectively.

As the vehicle travels along the spline segments, the parameter u, that grows from 0 to 1 inside each segment, is a good way to detect segment changes. In this way, the current segment i and its polynomials $Y_i(u)$ and $X_i(u)$ coefficients are always known. To calculate the reference point (x_d, y_d), the value of $u = u_m$ that minimizes the Euclidean distance between the control point (x, y) and the trajectory must be calculated. Setting to zero the first derivative of the distance function, we obtain:

$$A \cdot u_m^5 + B \cdot u_m^4 + C \cdot u_m^3 + D \cdot u_m^2 + E \cdot u_m + F = 0 \tag{5.7}$$

with (for simplicity, the subscript i of the spline coefficients has been omitted):

$$\begin{aligned}
A &= 3d_x^2 + 3d_y^2 \\
B &= 5c_x d_x + 5c_y d_y \\
C &= 2c_x^2 + 4b_x d_x + 2c_y^2 + 4b_y d_y \\
D &= 3c_x b_x - 3x d_x + 3a_x d_x + 3b_y c_y - 3y d_y + 3a_y d_y \\
E &= b_x^2 - 2x c_x + 2a_x c_x + b_y^2 - 2y c_y + 2a_y c_y \\
F &= a_x b_x - x b_x + a_y b_y - y b_y
\end{aligned} \tag{5.8}$$

The solution u_m of the polynomial closest to the last value of u must be chosen, and when this solution overtakes 1, i must be incremented to change to a new spline segment. Therefore, the reference position is given by:

$$\chi_d = \left(X_i(u_m), Y_i(u_m), \arctan\left(\frac{Y_i'(u_m)}{X_i'(u_m)}\right)\right) \tag{5.9}$$

The tracking errors are:

$$\xi = \begin{bmatrix} d_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} (y - y_d)\cdot\cos(\theta_d) - (x - x_d)\cdot\sin(\theta_d) \\ \theta - \theta_d \end{bmatrix} \tag{5.10}$$

Taking the tracking errors ξ as the state vector, and the steering angle ρ as input, we can obtain the derivatives of the states from the kinematics of the vehicle:

$$\dot{\xi}(t) = \begin{bmatrix} \dot{d}_e(t) \\ \dot{\theta}_e(t) \end{bmatrix} = \begin{bmatrix} V\cdot\sin(\rho(t) + \theta_e(t)) \\ \frac{V\cdot\sin(\rho(t))}{L} \end{bmatrix} \tag{5.11}$$

where V is the linear velocity of the vehicle, which is considered as a parameter for the design of the lateral controller. Assuming small errors and steering angles, we obtain the following linear state equation for the system:

$$\begin{bmatrix} \dot{d}_e(t) \\ \dot{\theta}_e(t) \end{bmatrix} = \begin{bmatrix} 0 & V \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} d_e(t) \\ \theta_e(t) \end{bmatrix} + \begin{bmatrix} V \\ \frac{V}{L} \end{bmatrix} \cdot \rho(t) \tag{5.12}$$

and its discrete version, assuming the controller works with a sampling period T_s, is:

$$\begin{bmatrix} d_e \\ \theta_e \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & V\cdot T_s \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} d_e \\ \theta_e \end{bmatrix}_k + \begin{bmatrix} V\cdot T_s + \frac{V^2\cdot T_s^2}{2\cdot L} \\ \frac{T_s\cdot V}{L} \end{bmatrix} \cdot \rho(k) \tag{5.13}$$

The control law is reduced to the following expression:

$$\rho(k) = -K\cdot\xi(k) = -\begin{bmatrix} K_1 & K_2 \end{bmatrix} \cdot \begin{bmatrix} d_e \\ \theta_e \end{bmatrix}_k \tag{5.14}$$

where K is calculated as an LQR so that it minimizes the following quadratic cost function:

$$\psi = \sum_{k=0}^{N} \xi^T(k)\cdot Q\cdot\xi(k) + \rho^T(k)\cdot R\cdot\rho(k) = \sum_{k=0}^{N} \begin{bmatrix} d_e & \theta_e \end{bmatrix}_k \cdot \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix} \cdot \begin{bmatrix} d_e \\ \theta_e \end{bmatrix}_k + \rho(k)\cdot r\cdot\rho(k) \tag{5.15}$$

The parameters q_{11}, q_{22} and r allow an intuitive adjustment of the relative importance of minimizing the lateral error, the orientation error and the control effort, respectively.

5.2.5. Delay compensation

In urban environments where AVs may need to travel at medium velocities, delays become significant and can destabilize the control system. Two primary sources of delay within the control loop must be considered: the delay in localization system readings, and the delay introduced by the actuators. These delays are illustrated in Figure 5.6 and are characterized as follows:

- At time k, when the control signal is computed, it is assumed that the available position reading ξ_r corresponds to the vehicle's position n_p sampling periods before. This delay can result in a substantial error in the position data used for calculating the control signal.
- Similarly, it is assumed that the control signal calculated at time k will take effect on the vehicle after n_c sampling periods. It is evident that the vehicle's position will have changed by this time if it is moving.

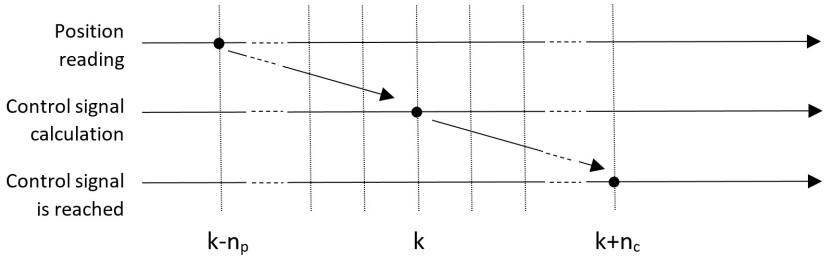


Figure 5.6: Delays involved in the control system.

For this reason, a delay compensation system is included (see Figure 5.2), which makes a prediction of the future position of the vehicle just when the actuation signal will take effect. For this, it is necessary to store in a buffer of size $(n_p + n_c)$ the last speed control signals sent to the actuators. In this way, at the sampling time k, the last known position $\chi(k - n_p)$ will be integrated until the sampling instant $(k + n_c)$ using the vehicle model and the speeds stored in the buffer:

$$\begin{aligned} &\text{for } n = k - n_p \text{ to } n = k + n_c, \\ &\chi(n) = \chi(n-1) + u(n-1) \cdot T_s \end{aligned} \tag{5.16}$$

being T_s the sampling period of the controller.

5.2.6. Experimental Results

This section presents the experimental results validating the proposed controller's performance. Initially, the test bench setup is described, followed by ablation tests to demonstrate the enhancements brought by the linear velocity profiler and delay compensator

subsystems. Finally, a comparative analysis with other state-of-the-art proposals is included.

Test Bench

The validation of the LQR controller is done in the CARLA simulator. The control signals in this simulator are "throttle" and "brake", normalized between [0,1], and steering normalized to [-1,1]. To transform the linear velocity output of our controller into these acceleration commands we implement a classical PID controller, while the steering command is sent directly to the simulated vehicle. Figure 5.7a presents a bird's eye view of the chosen route for controller testing.

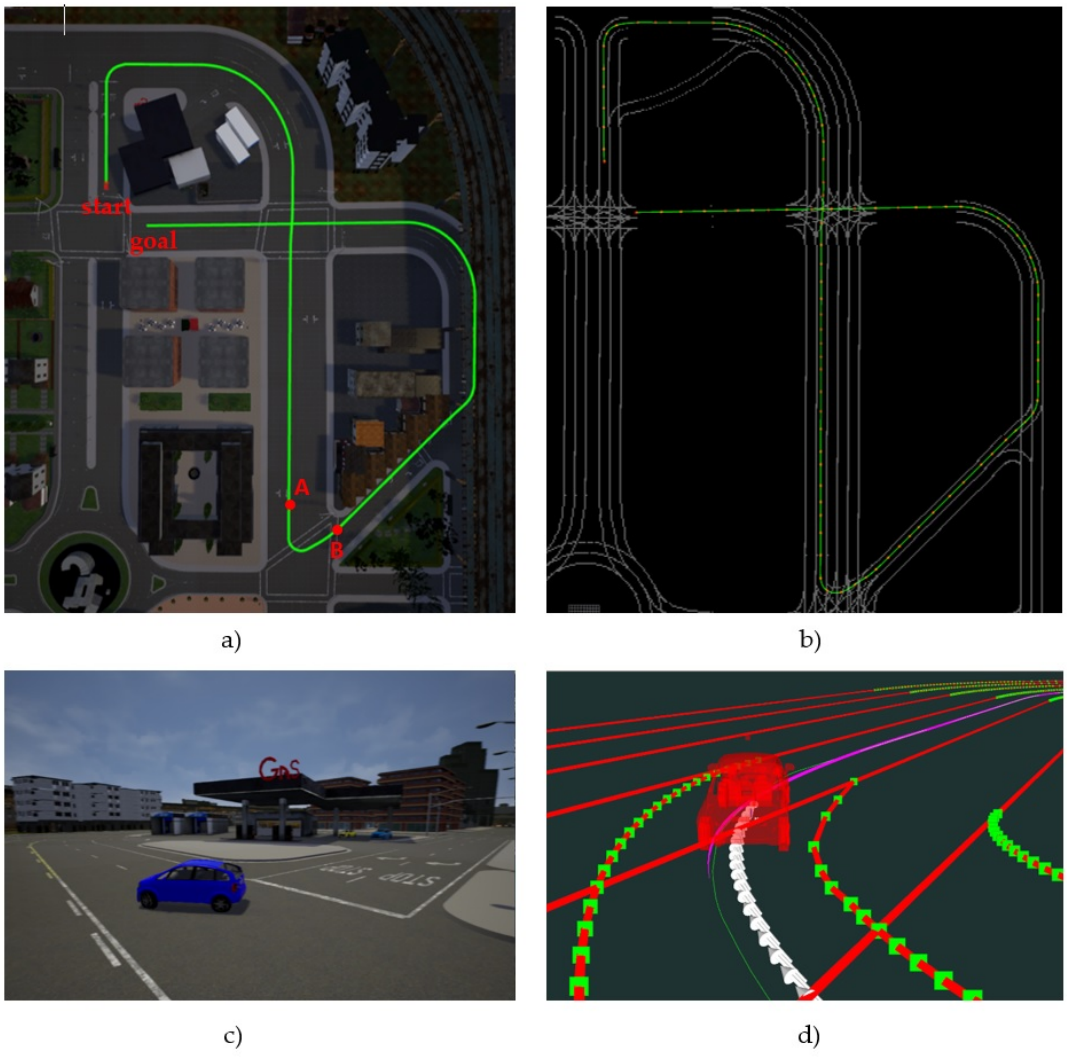


Figure 5.7: Test bench in the CARLA simulator: (a) 610-meter route in Town03 used for the tests. (b) View of the route in Rviz ROS visualizer, with the input waypoints shown as red points. (c) A closer view of vehicle model making the first turn in CARLA. (d) Rviz view of the same turn, with trajectory and reference pose markers for visualization.

A 610 meters path within the Town03 map, encompassing both straight lanes and sharp curves is defined. To define the path, 122 waypoints from start to goal were generated by the Global Planner, with an average distance of 5 meters between them. The waypoints and the interpolated spline are displayed in the Rviz [127] view (Figure 5.7b).

The experimental vehicle is an Audi A2, available within the CARLA simulator. Figures 5.7c and 5.7d show the vehicle navigating a curve in the CARLA simulator and the corresponding Rviz view with the path and reference poses generated by the controller, respectively.

Ablation Test

A series of tests were conducted to evaluate the improvements introduced by the various modules of the proposed controller. The basic LQR controller, lacking both the delay compensator and the velocity profiler, served as the baseline. We incrementally added these modules to the basic control: first the delay compensator, then the velocity profiler, and finally both, to form the complete controller. Figure 5.8 illustrates the lateral error, orientation error, steering angle, and linear velocity along the path for each test case. Table 5.1 shows the RMS value of errors during the complete route and in section AB (sharp curve), as well as the maximum, the average speed and the time to complete the route.

The performance of the basic LQR controller, depicted by a red dot-dashed line in Figure 5.8, was evaluated at a constant speed of 6 m/s (21 km/h). This speed was chosen to ensure the maximum lateral error in bends stayed around 1 m, allowing the vehicle to complete the route without leaving the lane. However, due to the consistent low velocity, the total time exceeded 100 seconds.

Integrating the delay compensator with the basic controller (blue dotted line in Figure 5.8) showed modest improvements at low speeds. Delays of np=10 and nc=8 (using $T_s = 0.1$ s) were estimated for localization measurements and actuation, respectively. The delay compensator enhanced actuation responsiveness, especially in bends, slightly reducing tracking errors and shortening the route completion time.

Incorporating the velocity profiler into the basic lateral control (magenta dashed line in Figure 5.8) allowed for dynamic velocity adjustments based on the path's curvature. With parameters set to $V_{\text{MAX}} = 13.5$ m/s (48 km/h) and $RC_{\text{MAX}} = 20$ m, the average speed increased to 8.65 m/s (31 km/h), though it dropped to 3 m/s (11 km/h) in high-curvature areas. This resulted in faster route completion and reduced tracking errors, particularly the lateral error in the A-B section.

The full controller, integrating both the delay compensator and the velocity profiler (blue continuous line in Figure 5.8), demonstrated further error reduction. The delay compensator's impact was more pronounced in straight sections, reducing oscillations at the end of bends when the vehicle accelerates. The controller completed the 610-meter route in 71 seconds with an average speed of 8.7 m/s (31 km/h), varying between 3 m/s (11 km/h) in sharp bends and 13.5 m/s (48 km/h) in straight sections. The average lateral error was 0.17 m, with a maximum of 0.60 m in bends, showcasing its suitability for autonomous driving in urban environments.

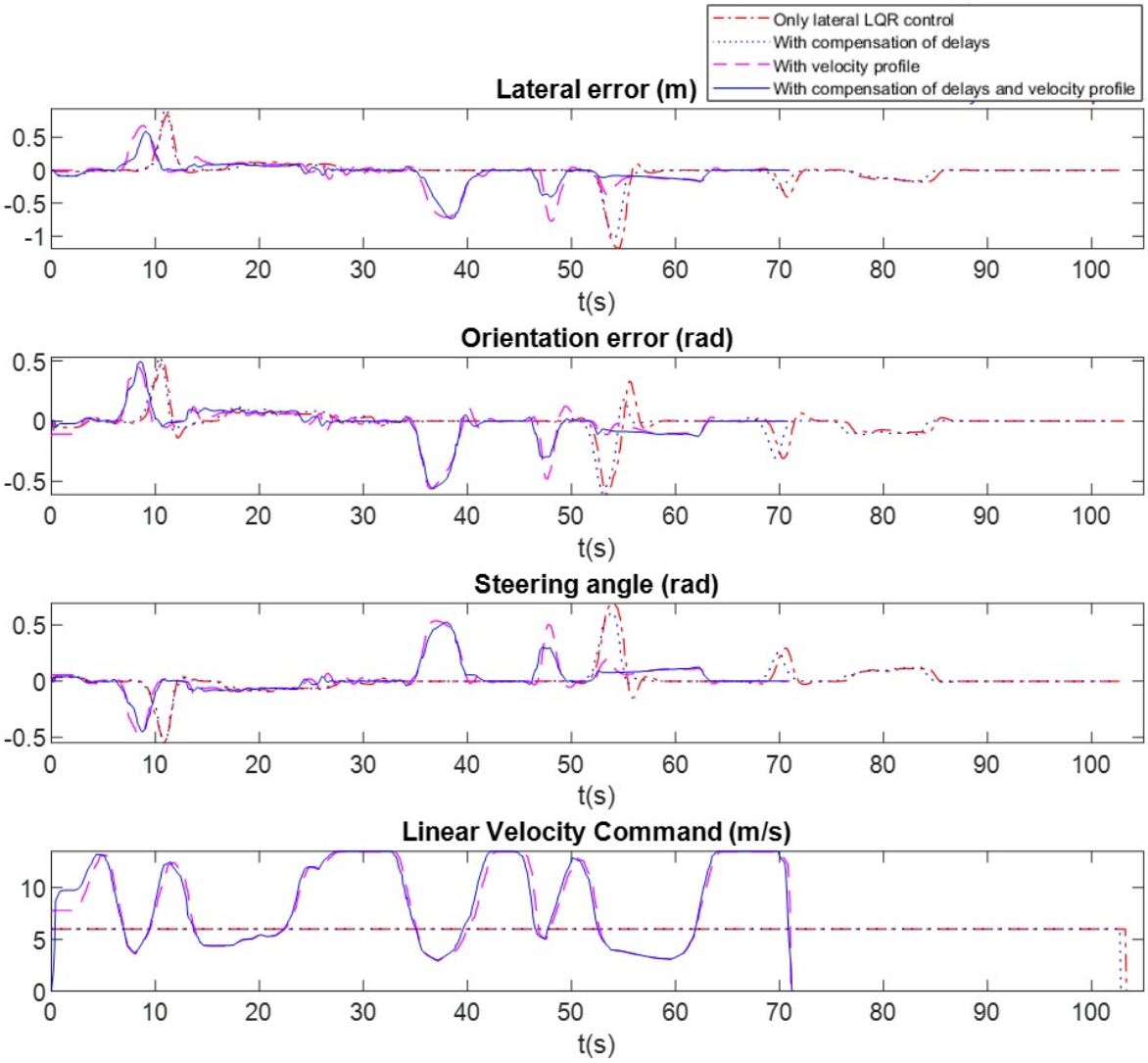


Figure 5.8: Ablation tests: (1) using only the basic LQR lateral controller with 6 m/s (21 km/h) constant speed (red dot-dashed line) (2) using LQR lateral control + delay compensation (blue dotted line) (3) using LQR lateral control + velocity profiler (with VMAX=13.5 m/s (48 km/h)) (magenta dashed line) (4) using the complete controller with delay compensation and velocity profiler (blue line).

Table 5.1: Ablation tests results.

	Basic LQR lateral control	Lateral control with delay compensation	Lateral control with speed profiler	Lateral control with delay compensation and speed profiler
Total lateral error (m)[1]	0.1954	0.1680	0.2041	0.1733
AB Bend Lateral error (m)[1]	0.5019	0.4395	0.3524	0.2924
Total Orientation error (rad)[1]	0.1087	0.1074	0.1509	0.1126
AB Bend Orientation error (rad)[1]	0.2285	0.2068	0.2268	0.2035
Average speed (m/s)	6	6	8.65	8.70
Maximum speed (m/s)	6	6	13.5	13.5
Time to complete the route (s)	103	102	71	70

[1]RMS Value

Comparison with other proposals

We show a comparison of our proposal with Pure Pursuit [128] standard (usually used as benchmark to validate path tracking controllers) and a previous low level controller developed in our research group [129], an efficient and real-time algorithm (BCM-DO) based on the Beam Curvature Method that deals with dynamic obstacles, that treats the lane edges as static obstacles to perform a lane following controller.

These tests have been carried out on the same test bench as the ones in previous subsection, using the same metrics to compare the controllers. The common parameter for the three algorithms is the maximum linear velocity, set to VMAX=13.5 m/s (48 km/h). Figure 5.9 and Table 5.2 show the temporal evolution and quantitative results respectively.

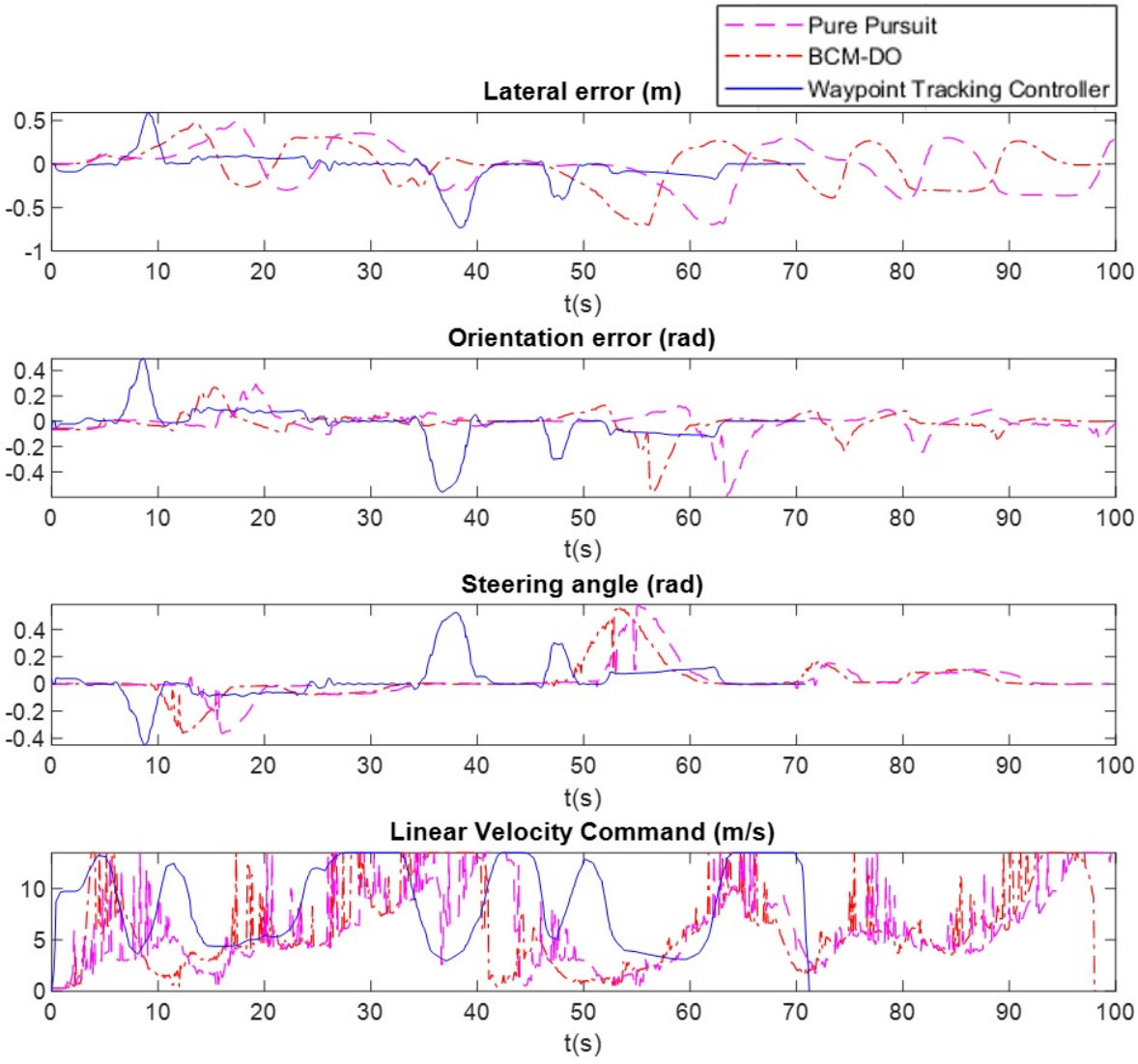


Figure 5.9: Comparison with other tracking controllers: (1) Pure Pursuit (magenta dashed line) (2) BCM-DO (red dot-dashed line) (3) proposed Waypoint Tracking Controller (blue continuous line).

As can be seen, our proposal maintains lower tracking errors with considerably higher average speed, completing the 610-meter route almost half a minute earlier than the other two controllers. An important improvement of our proposal can be seen in the straight sections of the route, drastically reducing oscillations and achieving much lower average errors in the complete route. A much smoother speed profile is also observed than the ones generated by the two other control algorithms, avoiding abrupt changes in the control signals, which contributes to an easier controllability.

Table 5.2: Comparison with other tracking controllers.

	Pure Pursuit	BCM-DO	Proposed Waypoint Tracking Controller
Total lateral error (m)[1]	0.2755	0.2842	0.1733
AB Bend Lateral error (m)[1]	0.3179	0.3125	0.2924
Total Orientation error (rad)[1]	0.1174	0.1055	0.1126
AB Bend Orientation error (rad)[1]	0.1520	0.1471	0.2035
Average speed (m/s)	6.1	6.2	8.7
Maximum speed (m/s)	13.5	13.5	13.5
Time to complete the route (s)	100	98	70

[1]RMS Value

5.3. Model Predictive Control for Manoeuvres Execution

The previous controller allows the tracking of nominal trajectories. However, in the presence of adversary vehicles or other obstacles blocking the nominal route, the decision making system of the tactical level will send the required actions to execute lane change manoeuvres or velocity reduction to adapt the trajectory to the current traffic conditions. These manoeuvres are executed by an MPC controller working in parallel with the previous controller, as it was shown in Figure 5.1. This work was partially published in the conference ITSC 2023 [130]: "Hybrid MPC and Spline-based Controller for Lane Change Maneuvers in Autonomous Vehicles".

The whole low-level control architecture, depicted in Figure 5.10, stands out as hybrid due to its integration of LQR and MPC optimization methodologies. It utilizes the classical LQR controller explained in the previous section (green blocks in Figure 5.10)) for adhering to the nominal spline-based trajectory. In this section, we incorporate two MPC controllers (red blocks in Figure 5.10), one for lateral and another for longitudinal control. This allows real-time modification of control signals without recalculating the nominal trajectory, by combining their outputs with the appropriate signals of the previous nominal controller. This hybrid structure ensures activation of these MPC controllers solely for manoeuvres, leveraging predictive control benefits while addressing its computational time constraints. Lane changes are initiated by setting a lateral offset reference signal (d_{lat_ref}) equal to the lane width $\pm L_{\mathrm{w}}$, determining the direction of the

lane change, to the lateral MPC controller. This controller, using a linear model and constraints, generates a lateral offset (d_{lat}) that, added to the nominal lateral error (d_e), produces a smooth lane change using the LQR controller. If there is a slow leading vehicle, the linear velocity is reduced through the longitudinal MPC module by aligning its reference signal (V_{lon_ref}) with the preceding vehicle's velocity for ACC. Additionally, if there is a stop signal, this longitudinal controller adjusts the vehicle's velocity considering the distance to the intersection point (d_i), that is predefined in the tactical trajectory and corresponds to the beginning of the intersection for each of the possible use cases.

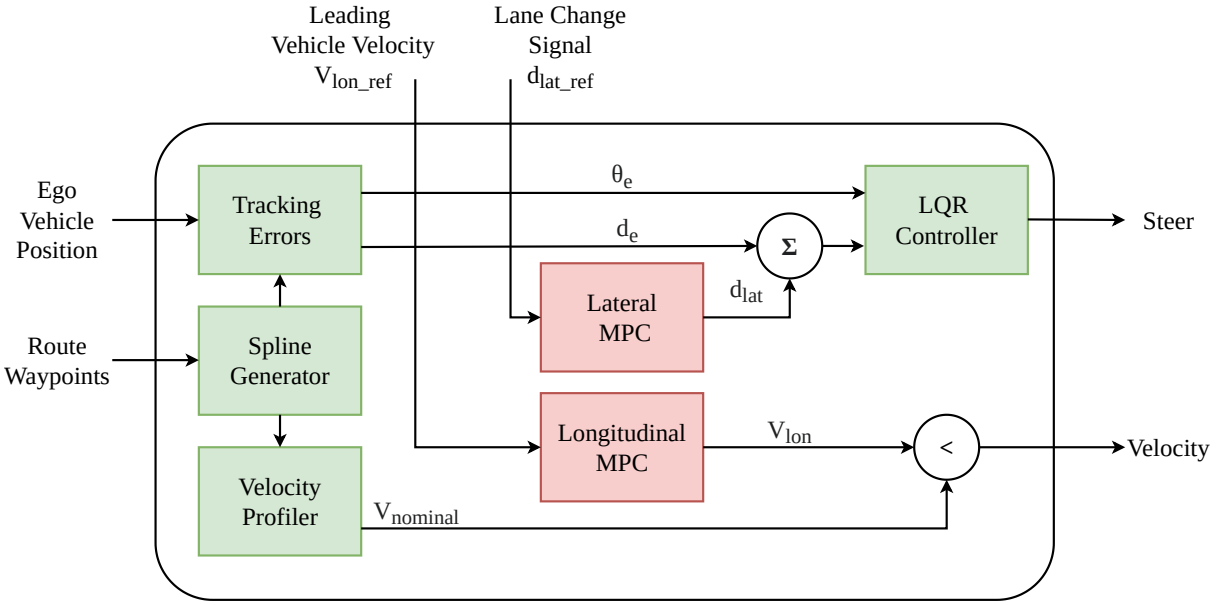


Figure 5.10: Architecture of the hybrid controller: Green blocks correspond to the waypoint tracking controller and red blocks correspond to the MPC controller for manoeuvres execution.

5.3.1. Motion Integral Models and Constrains

In the field of AD, the conventional approach to control using MPC has relied on the kinematic or dynamic model of the vehicle. However, these non-linear models require significant computation time for real-time execution. Our MPC controllers do not rely on the vehicle's model, since the LQR control operates continuously for vehicle trajectory tracking, while MPC controllers only need simple and linear models for lateral displacements and longitudinal velocity adjustments. Additionally, the type of manoeuvres executed (lane change, Adaptative Cruise Control (ACC) and stop) are not compromised by the model precision.

Longitudinal Model

The linear model used is a decoupled integrator chain for longitudinal and lateral dynamics. For longitudinal dynamics, a triple integrator chain based on jerk is used:

$$d_{\text{lon}} = \int\int\int j_{\text{lon}}(t)\, dt^3 \tag{5.17}$$

where d_{lon} is the longitudinal travelled distance and j_{lon} the longitudinal jerk. Additionally, from this integration can be obtained longitudinal speed v_{lon} and the longitudinal acceleration a_{lon}. The discrete linear state-space representation with sample time T_s of this longitudinal model is:

$$X_{lon}(k+1) = \begin{bmatrix} d_{\text{lon}} \\ v_{\text{lon}} \\ a_{\text{lon}} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_{\text{lon}} \\ v_{\text{lon}} \\ a_{\text{lon}} \end{bmatrix}_k + \begin{bmatrix} \frac{T_s^3}{6} \\ \frac{T_s^2}{2} \\ T_s \end{bmatrix} j_{\text{lon}}(k) \tag{5.18}$$

The output of the MPC longitudinal controller is a smooth velocity profile v_{lon} to follow the reference signal $v_{\text{lon\_ref}}$. Moreover, the acceleration changes must be smooth to ensure comfort and safety, and this is the reason to include acceleration and jerk as part of the model.

All these states and control variables are bounded. The jerk is limited to a maximum value of $j_{\max}$ (this parameter and its relationship with comfort is presented in [131]). The acceleration is bounded between maximum deceleration $a_{\text{lon,min}}$ and maximum acceleration $a_{\text{lon,max}}$, and the velocity is bounded between 0 and the maximum velocity given by the velocity profiler of the spline trajectory v_{nominal}. The longitudinal distance is also bounded to avoid a collision with the front vehicle located at a distance D_{front}. These are the constraints for the longitudinal model:

$$\begin{aligned} 0 \leq d_{\text{lon}} &\leq D_{\text{front}}, \\ 0 \leq v_{\text{lon}} &\leq v_{\text{nominal}}, \\ a_{\text{lon,min}} \leq a_{\text{lon,max}} &\leq a_{\max}, \\ -|j_{\max}| \leq j_{\text{lon}} &\leq |j_{\max}| \end{aligned} \tag{5.19}$$

The $v_{\text{lon\_ref}}$ and the D_{front} constraints have to be modified to execute the manoeuvres. The set of longitudinal boundary constraints are referred as $h_{\text{lon}}(k)$.

Lateral Model

The lateral model is a double integrator chain described by:

$$d_{\text{lat}} = \int \int a_{\text{lat}}(t)\, dt^2 \tag{5.20}$$

being d_{lat} the lateral offset (distance) to the nominal spline trajectory and a_{lat} the lateral acceleration. Additionally, v_{lat} is obtained during the integration process and it is the lateral speed. Using as states d_{lat} and v_{lat}, the following discrete state-space representation is obtained:

$$X_{lat}(k+1) = \begin{bmatrix} d_{\text{lat}} \\ v_{\text{lat}} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_{\text{lat}} \\ v_{\text{lat}} \end{bmatrix}_k + \begin{bmatrix} \frac{T_s^2}{2} \\ T_s \end{bmatrix} a_{\text{lat}}(k) \tag{5.21}$$

The lateral output of the MPC is a smooth lateral offset d_{lat}, which is added to the lateral error of the LQR controller to achieve a lane change manoeuvre.

To ensure a smooth lane change without encroaching into adjacent lanes, constraints play a crucial role in lateral control. Being L_{w} the lane width, V_{w} the vehicle width, and $v_{\text{lat,max}}$ and $a_{\text{lat,max}}$ the maximum lateral velocity and acceleration, the lateral constraints to remain in the nominal lane are:

$$\begin{aligned} \left(-\frac{1}{2}L_{\text{w}} + \frac{1}{2}V_{\text{w}}\right) \leq d_{\text{lat}} \leq \left(\frac{1}{2}L_{\text{w}} - \frac{1}{2}V_{\text{w}}\right), \\ -|v_{\text{lat,max}}| \leq v_{\text{lat}} \leq |v_{\text{lat,max}}|, \\ -|a_{\text{lat,max}}| \leq a_{\text{lat}} \leq |a_{\text{lat,max}}| \end{aligned} \tag{5.22}$$

The first constrain must be convenient modified to execute a lane change manoeuvre. These lateral bounded constraints are named as $h_{\text{lat}}(k)$.

5.3.2. Manoeuvres Execution

In our AD architecture we consider three different manoeuvres: **ACC** and **Stop**, executed by the longitudinal MPC controller; and **Lane Change**, executed by the lateral MPC controller.

The ACC behaviour is executed when a slow leading vehicle is encountered and no further action is received. To accomplish this behaviour, the velocity reference $v_{\text{lon\_ref}}$ is determined based on this leading vehicle's velocity. Moreover, D_{front} is established as the minimum distance to maintain from the preceding vehicle, thereby avoiding driving too close and preventing potential frontal collisions.

The stop action is received from the tactical level when the vehicle approaches an intersection and the agent deems it unsuitable to proceed, due to the presence of adversarial vehicles. Under these circumstances, the longitudinal velocity reference $v_{\text{lon\_ref}}$ is set to zero, and the D_{front} constraint is adjusted based on the distance to the intersection d_i. Consequently, the vehicle's velocity decreases until it reaches the start of the intersection, ensuring smooth movements by accomplishing the acceleration and jerk constrains.

For lane change manoeuvres, the lateral reference displacement ($d_{\text{lat\_ref}}$) value is set equal to the lane width L_{w}. When the vehicle is on the nominal trajectory, two actions are possible: a left lane change, which requires a negative lateral reference, and a right lane change, which requires a positive lateral reference. The lateral constrains are conveniently modified considering this reference, ensuring the vehicle's smooth transition between lanes while maintaining acceleration and jerk limits. If the vehicle is not on the nominal lane, setting the reference to zero results in the return to the original path.

Optimization Function

Once the constraints and reference signals necessary to perform the manoeuvres have been defined, as well as their operation, the output signals of the MPC controllers are calculated. The optimization process is resolved using a Quadratic Problem formulation, where it minimizes the function $\Phi(X(k), u(k))$, which is a quadratic cost function of the state vector $X(k) = [X_{\text{lon}} \quad X_{\text{lat}}]^T$ and the control vector $u(k) = [j_{\text{lon}} \quad a_{\text{lat}}]^T$. This problem can be summarized as:

$$\begin{aligned} \text{minimize} \quad & \Phi(X(k), u(k)) \\ \text{subject to} \quad & h_{\text{lat}}(k), \quad h_{\text{lon}}(k) \end{aligned} \tag{5.23}$$

where $h_{\text{lat}}(k)$ and $h_{\text{lon}}(k)$ are the lateral and longitudinal bounded constraints, respectively. The optimization function is defined by:

$$\Phi(X(k), u(k)) = (d_{\text{lat}}(k) - d_{\text{lat\_ref}}(k))^2 + (v_{\text{lon}}(k) - v_{\text{lon\_ref}}(k))^2 \tag{5.24}$$

5.3.3. Experimental Results

In this section, we evaluate the efficacy of our proposed MPC controllers through a series of experiments. First, we present a qualitative analysis of the manoeuvres introduced in this chapter. Then, we discuss the performance of our approach in an ablation study. Finally, a comparative analysis, positioning our controller in the context of existing SOTA methodologies is presented.

Manoeuvres Execution Qualitative Results

Three experiments are conducted to showcase the performance of the manoeuvres execution module. For each one, we present the evolution of linear velocity, steering and comfort metrics (acceleration and jerk) to evaluate the vehicle's response.

The initial experiment demonstrates the effectiveness of our ACC system. The signals, with the target velocity and steer illustrated in red and the vehicle's actual response in blue, are depicted in Figure 5.11. The vehicle encounters a slower leading vehicle and maintains a steady following distance, decelerating to match the leading vehicle's speed. In the 15-second, our vehicle begins to slow down, ultimately matching the leading vehicle's speed of 5 m/s (18 km/h), as opposed to the nominal speed of 14 m/s (50 km/h). The control signals are smoothly followed, exhibiting slight variations during deceleration manoeuvre when the leading vehicle is detected.

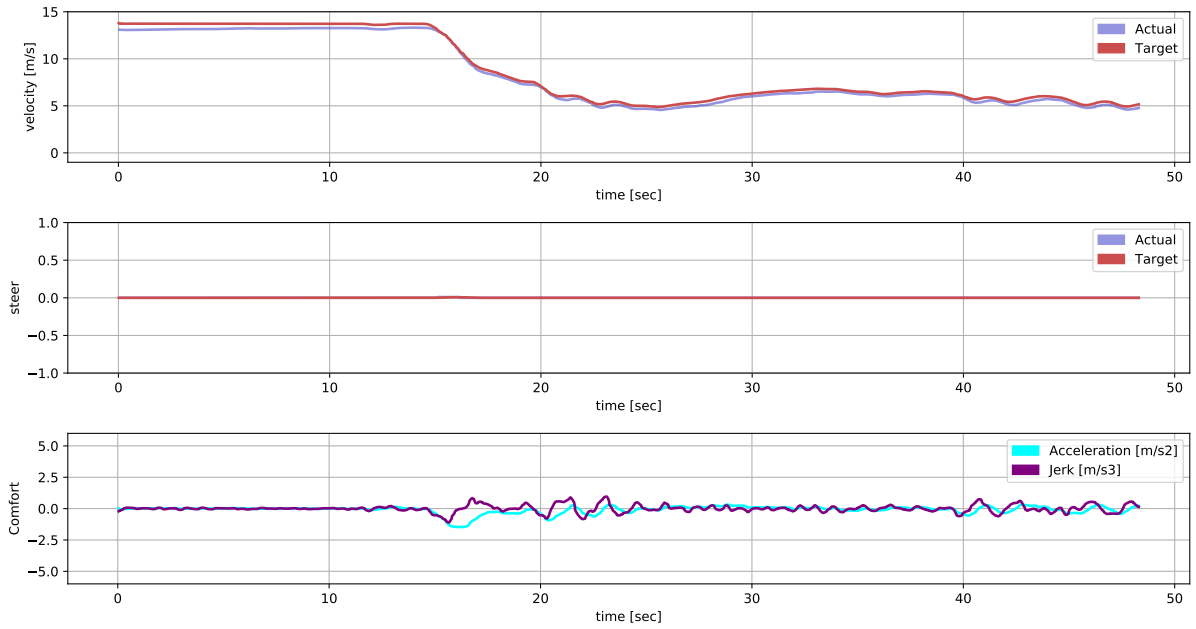


Figure 5.11: Control signals during an ACC manoeuvre. The linear velocity is depicted in the top, steering data is presented in the middle, and comfort metrics (acceleration and jerk) are illustrated in the bottom.

The second experiment illustrates in Figure 5.12 the control signals and the simulated vehicle's response to a stop action. At second 0.8, the vehicle receives a stop command, prompting a gradual decrease in target velocity over 1 second. The vehicle achieves a complete stop by the fourth second.

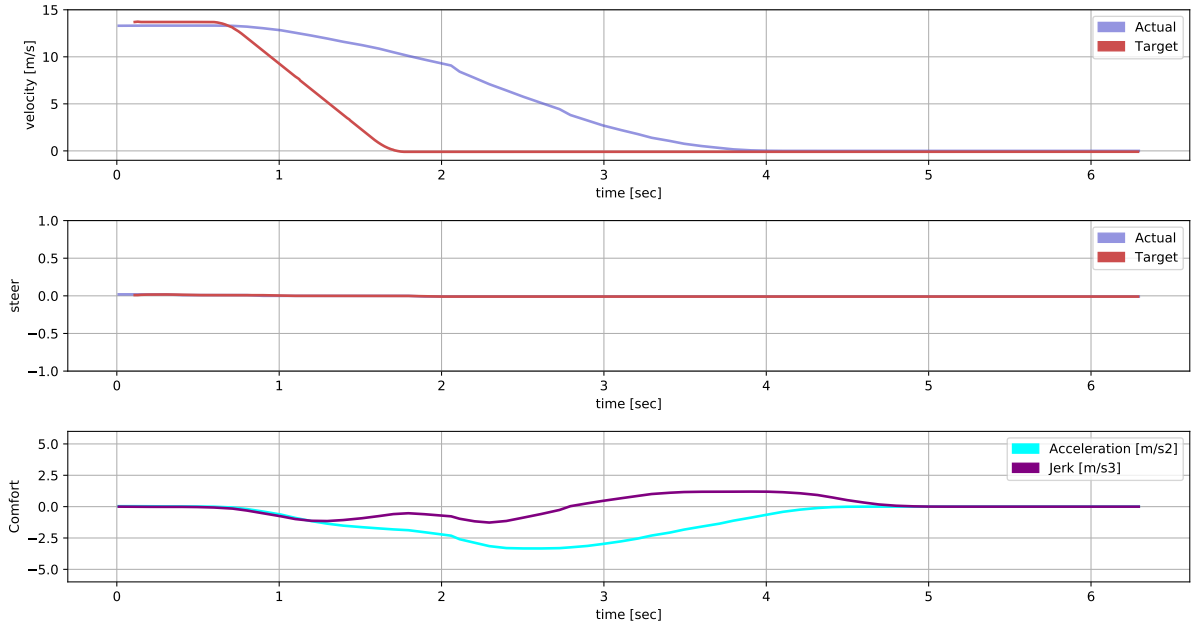


Figure 5.12: Control signals during a Stop manoeuvre. The linear velocity is depicted in the top, steering data is presented in the middle, and comfort metrics (acceleration and jerk) are illustrated in the bottom.

This scenario represents a situation where the vehicle must decelerate from its nominal velocity of 14 m/s (50 km/h) to zero. Despite this, the acceleration and jerk remain within comfortable limits, ensuring a smooth deceleration process. The third experiment illustrates the control signals for a lane change scenario in Figure 5.13, where the vehicle executes a left change (second 1.5), followed by a right change, returning to the nominal lane (second 10).

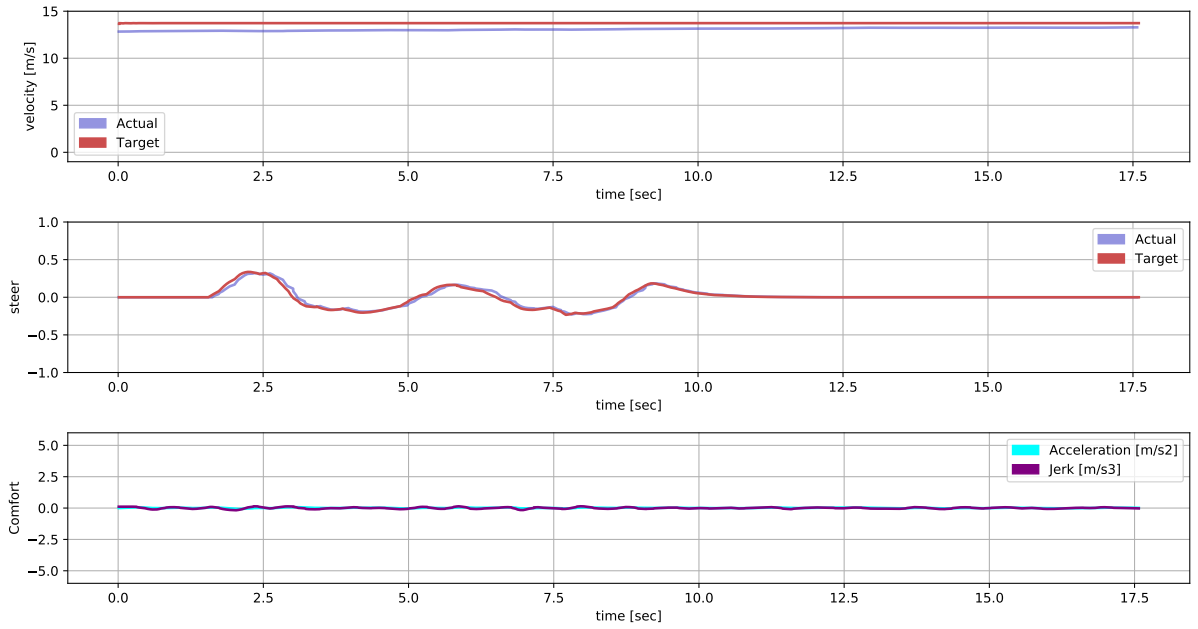


Figure 5.13: Control signals during a Lane Change manoeuvre. The linear velocity is depicted in the top, steering data is presented in the middle, and comfort metrics (acceleration and jerk) are illustrated in the bottom.

The vehicle adheres closely to the target signals, and both the acceleration and jerk signals demonstrate smooth transitions throughout the manoeuvre. It consistently maintains the nominal velocity during the turns.

Ablation Study

In CARLA, we designed an overtaking scenario for an ablation study of our system. This scenario includes a stationary vehicle in the middle of the ego vehicle's lane. Key moments for lane change actions, depicted in Figure 5.14 with dashed blue lines, trigger the execution of the manoeuvre. By conviniently modifying the lateral constrains and considering the acceleration and jerk limits, our system successfully completes the overtaking manoeuvre in a safe and smooth way.

In Figure 5.15, a comparison is presented between our whole proposed method and an alternative approach with the lateral MPC disabled. In the alternative case, the decision-making module's signal is directly introduced as an offset to the lateral error of the LQR controller. As observed, the LQR exhibits inadequate handling of a step signal at the error input, resulting in oscillations in the response. Although the overtaking manoeuvre

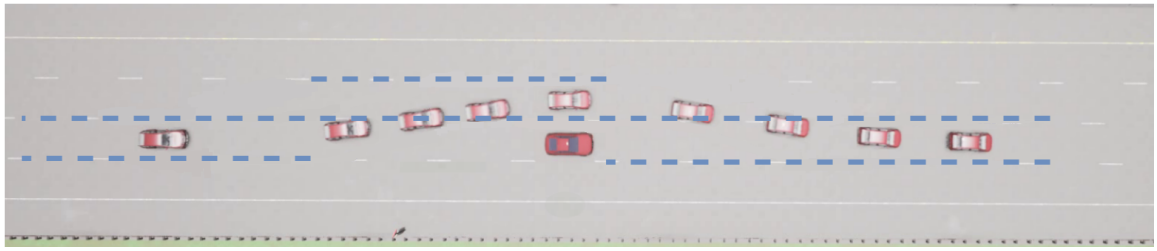


Figure 5.14: Simulation scenario in CARLA. Dashed lines depict MPC constraints. The secondary vehicle is stationary and ego vehicle's movement is represented by multiple frames.

in this particular case is of short duration, it should be noted that the alternative method would not be viable for prolonged lane changes, where extended driving in a lane different from the nominal one becomes necessary.

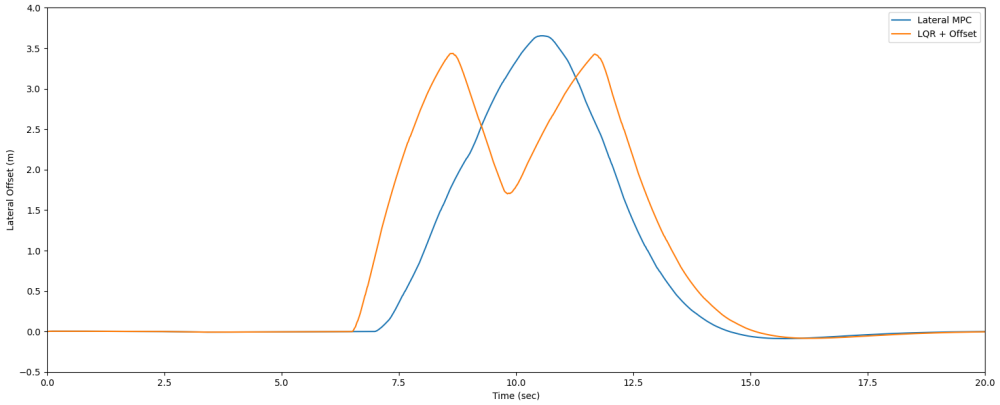


Figure 5.15: Lane change manoeuvre example: The MPC-based overtaking is illustrated in blue, while the results of directly introducing the offset signal to the LQR controller are depicted in orange.

The observed oscillations in the LQR + offset response is partially attributed to the controller's hyper-parameter values. While the calibration of the LQR primarily focuses on enhancing tracking accuracy, it may not adequately address overtaking manoeuvres.

Comparison with other state-of-the-art methods

To evaluate our approach in comparison to the current state of the art, we analyse a common scenario proposed by [132] where traffic lanes are partially occupied by obstacles. The prevailing methods primarily rely on optimization-based planning, mainly focusing on path generation. In Figure 5.16, we showcase the nominal trajectories generated by the SOTA methods with the same well-tuned cost function parameters, juxtaposed with the results of the avoidance manoeuvre implemented in our method. The methods of comparison are: MSM [133], which focuses on creating smooth paths by minimizing the fourth derivative of position, known as snap. PJM [134], that aims to minimize jerk by breaking the trajectory into segments. Lastly, the QP-based [132] method, which employs quadratic programming to find optimal trajectories, minimizing a quadratic cost function while adhering to both dynamic and kinematic constraints.

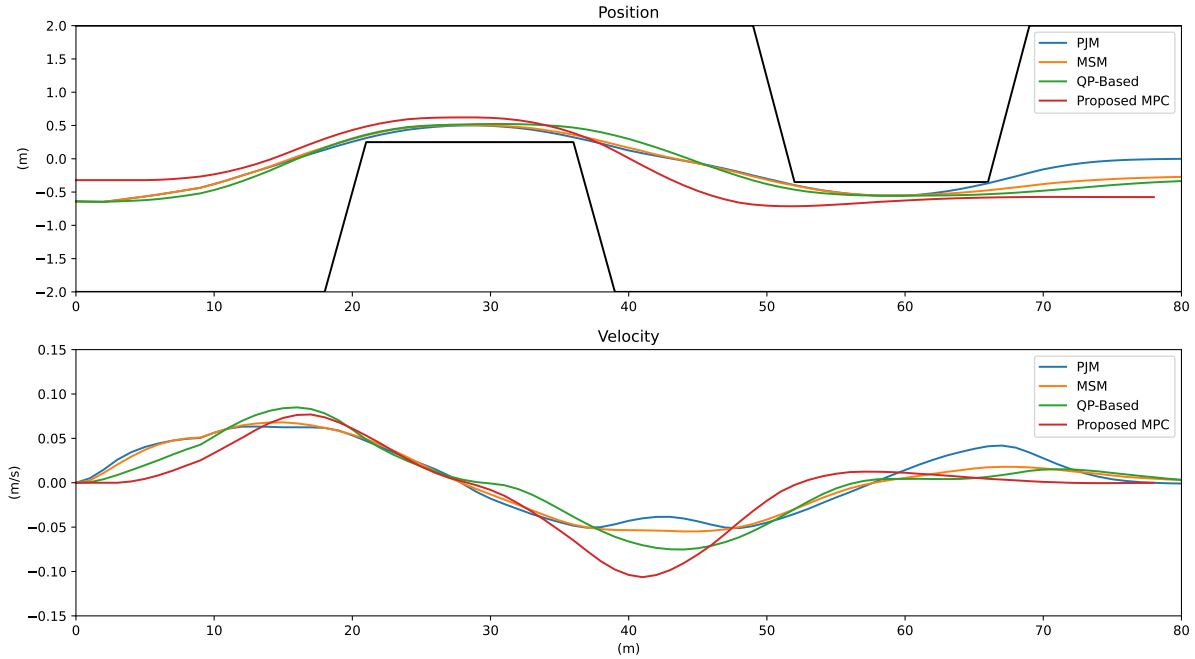


Figure 5.16: Comparison between our method, MSM, PJM, and Two-Phase QP-based: The figure illustrates the optimized paths and lateral velocity.

As indicated in [132], while the MSM method ensures snap continuity, its reliability is heavily dependent on cost function parameters. This reliance on parameter values makes the approach less reliable in complex scenarios. On the other hand, the PJM method does not guarantee jerk continuity, resulting in increased vibration and reduced smoothness in the final trajectory. Regarding our approach to the matter, when considering comfort along the nominal trajectory, the utilization of splines and the velocity profiler effectively highlights their functionality. However, with regards to lateral deviation, as explained in the preceding section, smoothness during manoeuvres is achieved by filtering the ($d_{\text{lat\_ref}}$) signal through the integrator chain and appropriately configuring the operational limits of the state variables.

In contrast to SOTA methods that assume strict adherence of the vehicle to generated trajectories without considering possible errors in the path tracking controller, our proposal takes a different approach. This key distinction ensures that the trajectories are always collision-free, unlike the method proposed by Yuncheng Jiang et al. [132] that may require multiple iterations to achieve this goal.

5.4. Summary

This chapter delves into two pivotal sections: the LQR for trajectory tracking and the MPC for manoeuvres execution, both are crucial for our architecture navigation.

The LQR is a classic control approach that ensures smooth and comfortable trajectory tracking. This section discusses both longitudinal and lateral control mechanisms. The

longitudinal control manages the vehicle's speed, while the lateral control is responsible for steering accuracy, ensuring the vehicle adheres to its intended path. A notable aspect here is the use of spline interpolation for creating smooth trajectories. Besides a velocity profiles and a delay compensation system are described and tested in CARLA simulator.

The second section centres on the MPC, which is used for executing lane changing, stop and ACC manoeuvres. This highlights the benefits of using a nominal coupled smooth trajectory, which simplifies the computational requirements compared to traditional non-linear vehicle models. The MPC approach, with its focus on manoeuvres, complements the continuous operation of the LQR system, together ensuring a comprehensive control mechanism for the vehicle.

Chapter 6

Reinforcement Learning for Decision Making in Urban Scenarios

El primer paso hacia la resolución de cualquier problema es reconocer que existe.

Albert Einstein

6.1. Introduction

This chapter was partially published in the conference IV 2023 [9]: "Hybrid Decision Making for Autonomous Driving in Complex Urban Scenarios", where it was accepted as a regular paper.

Urban driving environments encompass a variety of scenarios. In this thesis, we identify and explore four key scenarios common in many cities: crossroads, merges, roundabouts, and lane change. 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. This chapter details the scenarios, elaborates on the POMDP formulation for each scenario, and discusses the results obtained for different DRL algorithms. All research conducted here is performed using SUMO, where the low-level signals are generated by the simulator. Further experiments are presented in Chapter 7, where the full hybrid architecture is executed and the scenarios are evaluated in CARLA.

The vehicle under control, referred to as the "ego vehicle," is defined as an agent. Building on the concepts introduced in section 2.3, 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. A visual representation of this methodology is illustrated in Figure 6.1. The observations extracted from the simulator are related to distances and velocities of the adversarial vehicles, while

the actions executed by the agent are high-level decisions: driving, stopping, or changing lanes.

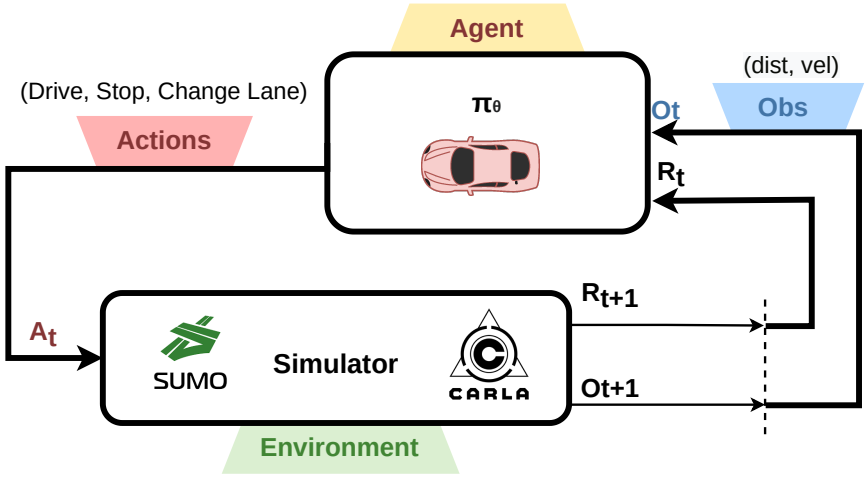


Figure 6.1: The observation vector is extracted from the simulator. The agent selects an action considering this vector and the reward for this action.

In the following sections, we first outline the configuration of the agents and then explore in detail the aspects related to each of the scenarios introduced earlier. The structure includes the following points:

1. An overview of each urban scenario, detailing its relevance and offering a general definition.

2. The formulation of the POMDP for each specific scenario, including states, observations, actions, and the reward function. It is important to note that the transition function is inherent to the simulation.

3. The description of the implementation of each scenario within the SUMO framework. We define the traffic flow, the adversarial vehicles and their behaviour.

4. An outline of the experiments, including the training and testing process. This includes a comparison of the DRL algorithms within the scenario. Besides, 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.

As highlighted in Chapter 3, existing literature primarily focuses on solving individual scenarios. In contrast, our research aims to establish a more holistic architecture able to be adaptable to an entire urban environment. To this end, we have designed specific individual scenarios for agent training, valid for transfer the acquired knowledge to the hybrid approach detailed in Chapter 4. A pivotal aspect of our proposal is the uniform POMDP formulation across these diverse scenarios, characterized by the adoption of low-dimensional observation vectors that encapsulate distances and velocities. Furthermore, to underscore the effectiveness of our methodology, we conduct a comparison with the conventional scenarios introduced in Section 4.3.2, specifically on those incorporated in the SMARTS framework, in Section 6.6.

6.1.1. Agent Configuration

The DRL algorithms in this study are implemented using the SB3 library. For a fair comparison among different DRL algorithms, each is trained over one million time-steps, ensuring the convergence of algorithms. 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 thesis incorporates a feature extraction module, which has proven to enhance the convergence of training [16]. 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. The outputs are then concatenated into a single vector, serving as the input for the DRL algorithms.

- **DRL Algorithms.** As introduced in Section 2.5, 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.

To ensure an equitable comparison among the algorithms, 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 policy-based algorithms configuration is depicted in Figure 6.2.

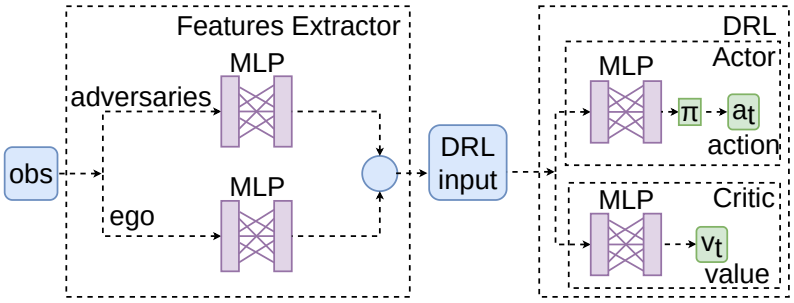


Figure 6.2: A representation of the policy-based algorithms configuration. Features from both adversaries and the ego vehicle are extracted separately. The concatenated extracted information is then input into the actor-critic structure of the DRL algorithm.

The hyper-parameter configuration selected for each agent, previously presented in Section 2.5, has been obtained following the recommendations of the SB3 library and the experience acquired during the research process. The configurations are presented in Tables 6.1, 6.2, 6.3 and 6.4. A concise overview of the parameters is presented as follows:

- **Learning Rate**: Step size at each iteration while moving toward a minimum of a loss function. A lower value results in slower convergence, while a very high value can cause the learning process to diverge.
- **Buffer Size (steps)**: This refers to the size of the memory buffer where the algorithm stores the experiences or transitions.
- **Start of Learning (steps)**: Specifies the point at which the learning algorithm begins processing the stored experiences.
- **Batch Size (samples)**: The number of training samples used in one iteration. It determines how much data will be used to estimate the gradient in optimization algorithms.
- **Discount Factor (γ)**: A value between 0 and 1 that determines the importance of future rewards compared to immediate rewards.
- **Gradient Steps**: Indicates the number of gradient descent steps to be taken for each learning iteration.
- **Target Network Update Interval (steps)**: Determines how frequently the weights of the target network are updated with the weights of the main network.
- **Maximum Gradient Clipping Value**: A technique to prevent exploding gradient by capping the values of gradients during back-propagation.
- **Number of Epochs for Optimizing Surrogate Loss**: Refers to the number of complete passes through the training dataset to optimize a specific loss function.
- **Clip Range**: It's used to limit the update of the policy at each step, ensuring small and incremental changes to the policy.

- **Target KL (Kullback-Leibler divergence)**: A measure to ensure that updates to the policy do not deviate too much from the previous policy.

Parameter	Value
Learning rate	5e-4
Buffer size (steps)	15000
Start of learning (steps)	200
Batch size (samples)	32
Discount factor, γ	0.8
Gradient steps	1
Target network update interval (steps)	50

Table 6.1: Parameters configuration of the DQN agent.

Parameter	Value
Learning rate	7e-4
Discount factor, γ	0.8
Maximun gradient clipping value	0.5

Table 6.2: Parameters configuration of the A2C agent.

Parameter	Value
Learning rate	1e-3
Batch size (samples)	128
Number of epochs for optimizing surrogate loss	10
Discount factor, γ	0.99
Clip range	0.2

Table 6.3: Parameters configuration of the TRPO agent.

Parameter	Value
Learning rate	3e-4
Batch size (samples)	64
Number of epochs for optimizing surrogate loss	10
Discount factor, γ	0.99
Target kl	0.01

Table 6.4: Parameters configuration of the PPO agent.

6.2. Lane Change Scenario

Roads in urban scenarios may have one or multiple lanes. We propose a lane change module to drive in these situations, where a slow or stopped vehicle can hinder vehicle driving. We propose a three lanes road, which can be extrapolated to any number of lanes. Our lane change module is responsible for determining the appropriate timing for lane changes. Meanwhile, SUMO is in charge on maintaining a safe velocity relative to the leading vehicle, thereby preventing frontal collisions.

6.2.1. POMDP formulation

State

The state of a vehicle is defined by its distance to the ego vehicle, its longitudinal velocity, and its driving intention: $s_i = (d_i, v_i, i_i)$, these physical values are normalized between $[0, 1]$ and driving intention of the adversarial vehicles are described by $i \in [0, 2]$. In this scenario, a vehicle may have three intentions: change left ($i = 1$), keep driving in its lane ($i = 0$), or change right ($i = 2$), as illustrated in Figure 6.3a. We define the state of the environment as the collection of the individual states of the adversarial vehicles:

$$s = (s_1, s_2, ..., s_n) \tag{6.1}$$

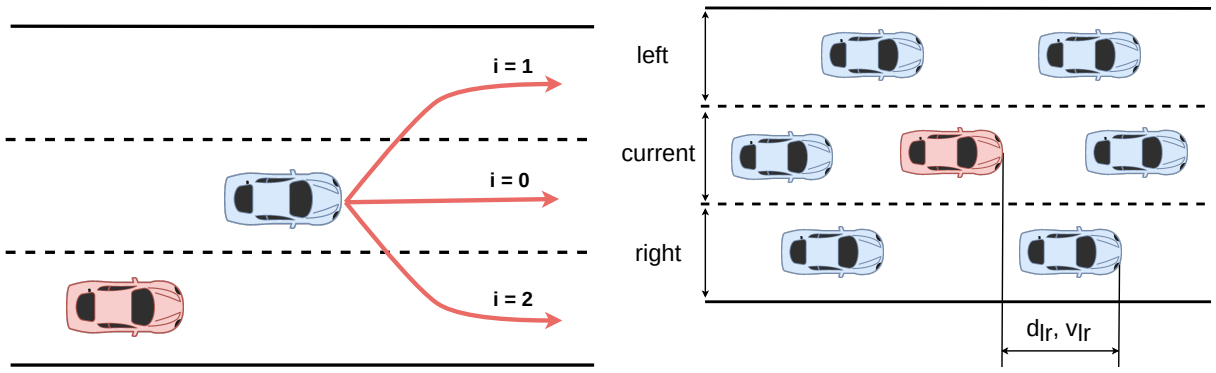
being n the number of adversaries within the lane change scenario. This definition remains consistent across all four scenarios studied in this chapter.

Observation

The observation matrix is defined by the nearest vehicles in the current and contiguous lanes of the ego vehicle. As shown in Figure 6.3b we consider the information of six vehicles, three leading vehicles, and three following vehicles. The observation space is defined by two key components: the relative normalized distances to surrounding vehicles, 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:

$$\Omega = (d_{ll}, v_{ll}, d_{lc}, v_{lc}, d_{lr}, v_{lr}, d_{fl}, v_{fl}, d_{fc}, v_{fc}, d_{fr}, v_{fr}) \tag{6.2}$$

where the indices indicate the lane of the vehicle (left, current, right) and its relative position (leading, following) with the ego vehicle.



(a) (b)

Figure 6.3: The state and observations representations of the lane change scenario. (**a**) The ego vehicle (red) is driving in the right lane. The adversary (blue) may have three possible intentions: stay straight (i=0), change left (i=1) or change right (i=2). (**b**) The observation vector in the lane change scenario is represented by the distance the ego vehicle red, relative to the six surrounding vehicles. Additionally, the velocities of these surrounding vehicles (blue), are also included in the observation vector.

Action

The DRL algorithm only controls the lane-changing decisions while the velocity is automatically controlled by the operative level. This is done by an ACC module, which adapts the ego vehicle velocity to the leading vehicle as it gets closer. In these experiments this task is executed by the simulator.

We propose a discrete set of actions: change to the left lane, continue in the current lane, and change to the right lane:

$$a = (change\ left, stay\ straight, change\ right) \tag{6.3}$$

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 try to drive as fast as possible. In addition, another small positive reward is given when the vehicle is in the right lane, since this lane is the corresponding to the nominal trajectory. 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 based on the right lane: k_i;
- Reward for reaching the end of the road: $+1$;
- Penalty for collisions: -2;

where $k_v = 1 \times 10^{-3}$ and $k_i = 5 \times 10^{-4}$. 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 to measure the vehicle's performance based on the average reward per episode. A reward nearing 1 indicates a successful driving behaviour.

6.2.2. Experiments

We have developed a lane change environment comprising three lanes in a 400-metre road. Every three to five seconds, an adversarial vehicle is randomly generated in each lanes. These vehicles can reach maximum speeds of 5 to 15 m/s (18 to 54 km/h). They are capable of performing a lane change depending on their intention, as outlined in the state definition. A visual representation of the scenario is presented in Figure 6.4.

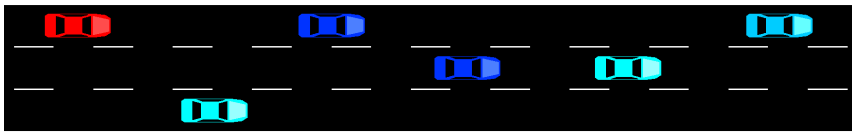


Figure 6.4: A bird-eye-view of the simulation framework in SUMO. The ego vehicle (red) is driving in the road. The colour intensity of the adversarial vehicles (blue) is related to their velocities, being the darker the fastest.

In Figure 6.5, a comparative analysis of the training rewards for the different DRL agents is presented. This analysis highlights the TRPO agent's superior performance, evident through its higher mean reward. Conversely, the A2C agent exhibits the lowest mean reward, indicating its relative ineffectiveness among the agents compared. The performances of the PPO and DQN agents are closely matched, with the DQN agent marginally outperforming the PPO. As the training process concludes, convergence is observed. With the maximum mean reward per episode set slightly over 1, the TRPO agent's performance nearing this threshold suggests its success in identifying an optimal policy for the lane change scenario.

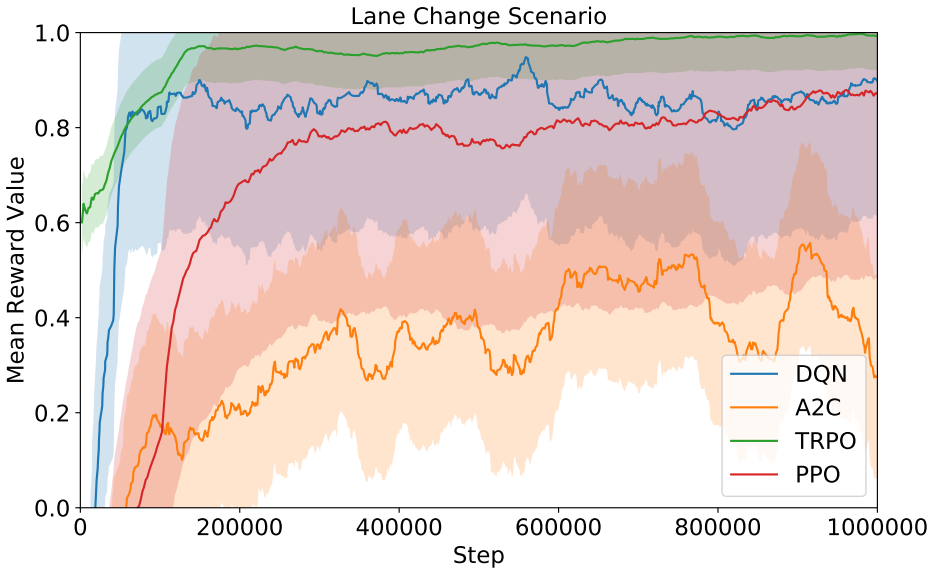


Figure 6.5: Evolution of the mean rewards during the training process for the DRL agents in the lane change scenario: DQN(blue), A2C(orange), TRPO(green) and PPO(red).

The evaluation of the agents was conducted over 1000 new episodes with the NNs weights obtained in the training phase, obtaining the findings detailed in Table 6.5. Throughout the training episodes, the TRPO agent stood out for its exceptional performance, achieving 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. The PPO and DQN agents also demonstrated a high success rate, but the first acts too risky and the second too conservative. The DQN keeps in the right lane without

overtaking slow adversarial and the PPO agent sometimes change lane when it is not necessary. Within the scope of AD, the TRPO agent has been chosen for integration into the hybrid approach.

Table 6.5: A comparison of the four DRL agents in the lane change scenario. The success rate [%] and the average time (sec) are presented.

Metric	DQN	A2C	TRPO	PPO
Success Rate [%]	83.40	81.20	**95.30**	88.30
Average Time (sec)	17.12	**16.32**	18.24	17.92

Figure 6.6 illustrates the sequence of events demonstrating the agent's behaviour. The scenario begins with the ego vehicle travelling in the right lane. Upon encountering an ahead vehicle in the same lane, the agent initiates a lane change to overtake. After that, it finds another vehicle in front of it, and waits until there is a gap in the right lane to return.

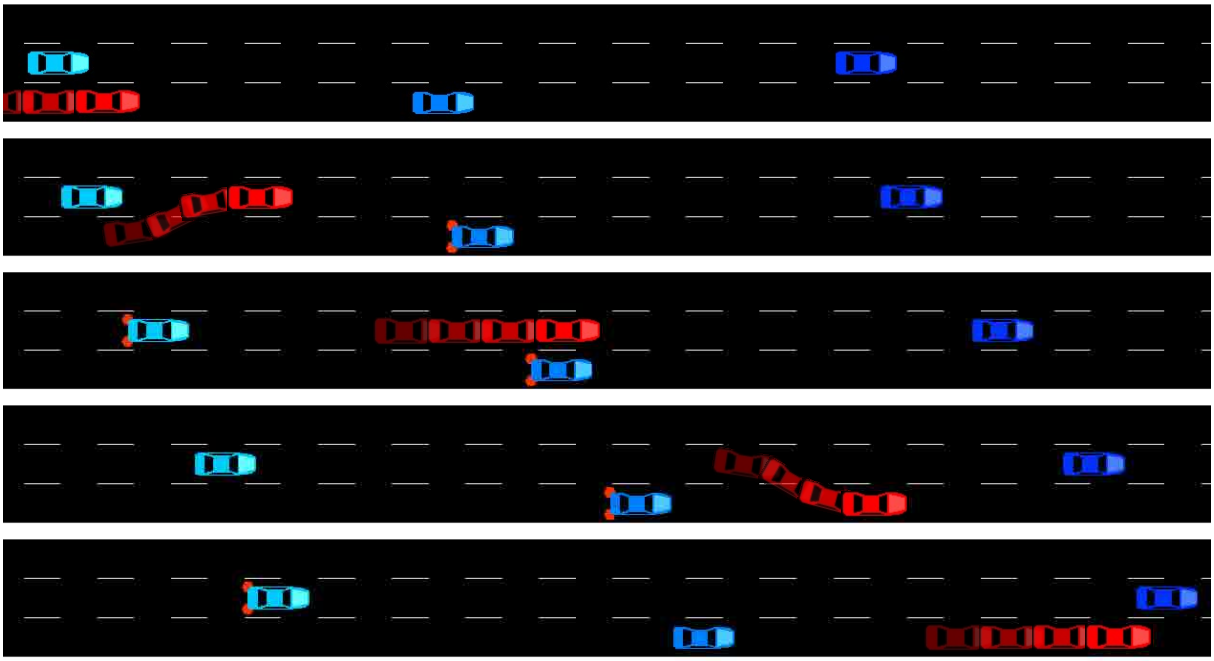


Figure 6.6: A bird's-eye view of a simulated episode in SUMO is presented, describing five scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded. This temporal representation illustrates the behaviour of the TRPO agent within the highway scenario.

6.3. Roundabout Scenario

Cities have various types of intersections, each with unique shape, behaviours, and rules for vehicles. A roundabout is a specific type of intersection designed to manage traffic flow. In a roundabout, vehicles entering must yield to those already inside. For efficient and safe traffic management, it's crucial to anticipate whether the vehicles within

the roundabout will remain circulating or exiting, allowing the incoming vehicles to enter and maintain a smooth and safe flow of traffic. In this section, we define a one lane roundabout that includes four exits. The agent's task is to determine the optimal timing for entering the roundabout, while the trajectory following is managed by the simulator.

6.3.1. POMDP formulation

State

The state of a vehicle is defined by its distance to the intersection point, its longitudinal velocity, and its roundabout exit intention: $s_i = (d_i, v_i, i_i)$, these physical values are normalized between $[0, 1]$ and the driving intention of the adversarial vehicles are described by $i \in [0, 1]$. The intersection point is defined by the location where the centre of the ego vehicle's lane and the centre of the opposing vehicles' lane intersect. In this scenario, this intention is predefined by the route to be followed. The vehicles inside the roundabout will leave it in the first or second exit, depending on this value. These two behaviours are represented in Figure 6.7a. and the state is defined as:

$$s = (s_e, s_1, s_2, ..., s_n) \tag{6.4}$$

We limit the exits to two because once our vehicle enters the roundabout, its velocity is controlled by the SUMO simulator to avoid frontal collisions, and it follows a predefined trajectory. Therefore considering additional exits does not add higher complexity.

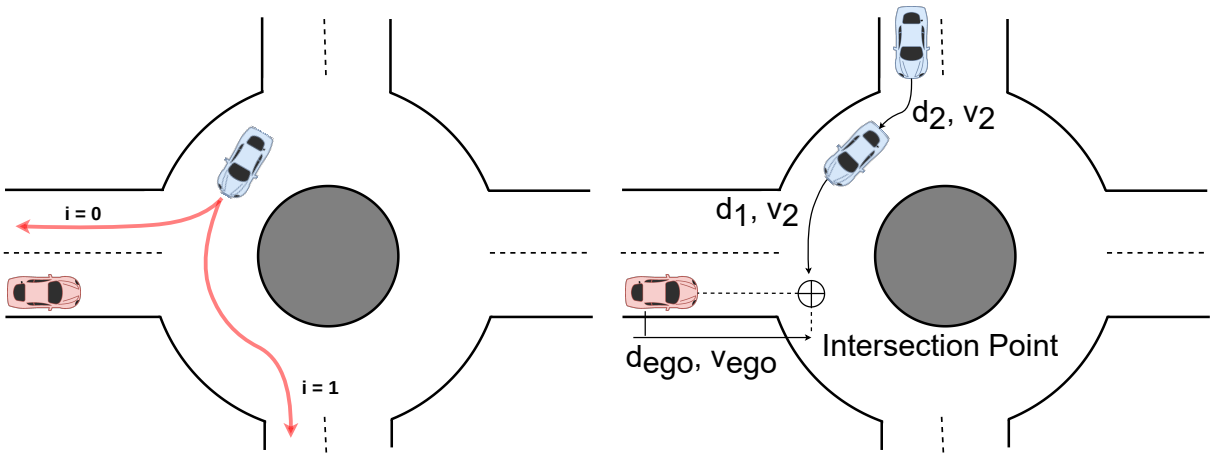
Observation

To simulate realistic experimental conditions, we assume a hypothetical perception module capable of detecting vehicles within a 50-meter radius, encompassing a full 360-degree field of view. Our approach does not rely on prior knowledge of the intersection type. It focuses on the longitudinal velocity and position of surrounding vehicles. Consequently, the observation vector is defined based on these parameters, providing the ego vehicle with essential information about the scenario. This vector is defined as follows, focusing on the ego vehicle and the two nearest adversarial vehicles:

$$\Omega = (d_e, v_e, d_1, v_1, d_2, v_2) \tag{6.5}$$

We consider that this configuration is sufficient for this scenario, as it effectively captures both positional and dynamic information through the distances and velocities of the surrounding vehicles. The observation vector is structured such that the nearest relevant vehicle always occupies the first input position. This configuration aids in emphasizing the significance of the closest vehicle to the DRL agent. When the nearest vehicle exits the relevant range (either by passing or exiting), the focus shifts to the next two vehicles,

ensuring the vector consistently contains the most relevant information at all times. The setup of the observation vector is illustrated in Figure 6.7b.



(a) (b)

Figure 6.7: The state and observations representations of the roundabout scenario. (**a**) The ego vehicle (red) is ready in merge into the roundabout. The adversary (blue) may have two possible intentions: take the first exit (i=0) or continue and take the second exit (i=1). (**b**) A representation of the observation vector within the roundabout scenario. The observation vector is defined by the ego vehicle's (red) distance to the intersection and its velocity. The two closest adversaries' (blue) distances and velocities are also considered in the observation matrix.

Action

The action space for navigating intersections consists of two high-level actions: 'stop' and 'drive'. These actions are strategically executed before entering the roundabout and are designed to guide the vehicle on when to merge into the roundabout traffic and when to yield to other vehicles. The primary focus of our system is to avoid collisions and ensure safe merging into traffic. The DM module is responsible for identifying appropriate gaps in traffic to merge safely. The action space for this system is thus defined as follows:

$$a = (stop, drive) \tag{6.6}$$

This configuration allows for straightforward yet effective DM, facilitating an efficient navigation through roundabouts.

Reward

The objective of an RL algorithm is to optimize the expected value of the discounted future reward. Different reward functions will modify the resultant policy. The purpose of the reward function, in these use cases, is to perform the rapid and safe navigation of the ego vehicle through an intersection, 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 in the following components:

- Reward based on the velocity: $k_v * v_{ego}$;
- Reward for crossing the intersection: $+1$;
- Penalty for collisions: -2;
- Penalty relative to the episode duration: $-0.2t/t_{out}$.

where $k_v = 2 \times 10^{-3}$. Under this setup, the episode reward ranges between [-2, 1.1].

6.3.2. Experiments

Traffic flow is a crucial element in roundabout design. To accommodate complex traffic patterns, we have constructed a sizeable roundabout with a 50-meter radius. The ego vehicle consistently enters the intersection in the same lane. Adversarial vehicles are randomly generated every three to five seconds in the previous entrance to the roundabout. These vehicles can achieve speeds ranging from 5 to 15 m/s (18 to 54 km/h) and can leave the roundabout before or after the ego vehicle incorporation, as detailed in the state definition. A visual depiction of this scenario is illustrated in Figure 6.8. In this implementation the SUMO simulator is in charge of avoiding frontal collisions once the vehicle has entered the intersection.

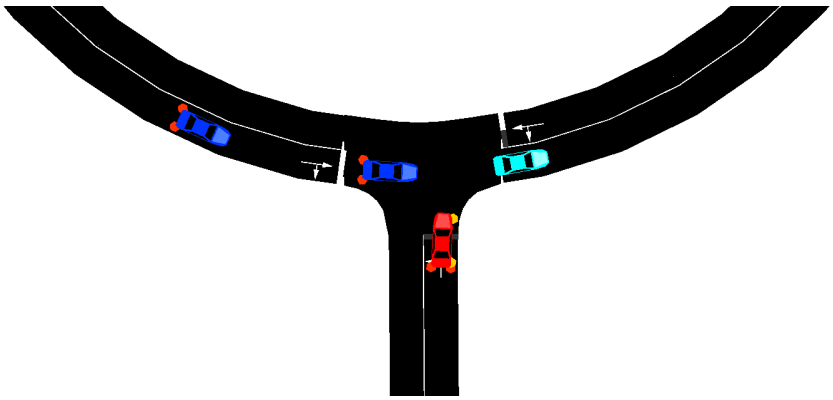


Figure 6.8: A bird-eye-view of the simulation framework in SUMO. The ego vehicle (red) is entering the intersection. The colour intensity of the adversarial vehicles (blue) is related to their velocities, being the darker the fastest.

In Figure 6.9, we present a comparative analysis of training rewards among the DRL agents. The analysis reveals that the TRPO agent shows superior performance, characterized by a higher mean reward. In contrast, the DQN agent is the least effective, showing the lowest mean reward among the evaluated agents. The performance of the A2C and PPO agents is similar, with the latter demonstrating a small better outcome. A notable observation across all agents is their ability to achieve convergence, typically around 200k

time-steps. Beyond this point, only minor enhancements in policy are observed until the end of the training process. Given that the maximum mean reward per episode slightly over 1, and the TRPO agent's performance approaches this value, it is reasonable to conclude that the TRPO agent successfully identifies an optimal policy for addressing the given scenario.

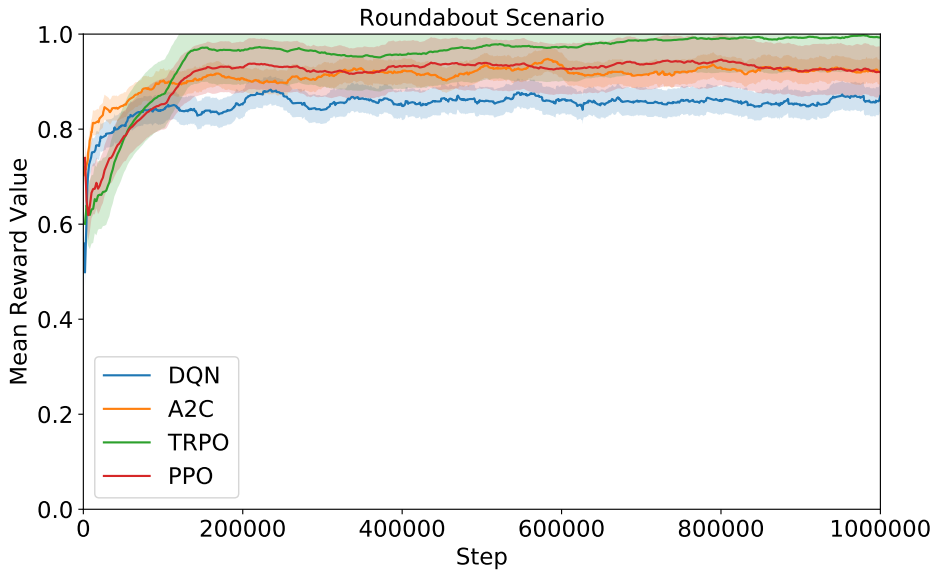


Figure 6.9: Evolution of the mean rewards during the training process for the DRL agents in the merge scenario: DQN(blue), A2C(orange), TRPO(green) and PPO(red).

After the training phase, the agents are evaluated over 1000 new episodes, with the results presented in Table 6.6. During the testing phase, the TRPO agent outperformed others, achieving the highest success rate at 91.3%. Conversely, the DQN exhibited the lowest success rate. The average duration for each episode was comparable across all four agents, with the PPO agent achieving the shortest average time while maintaining an impressive success rate. Given the priority of safety in AD, the TRPO agent is chosen for integration into the hybrid approach in roundabout scenarios, due to its balance between success rate and operational efficiency.

Table 6.6: A comparison of the four DRL agents in the roundabout scenario. The success rate [%] and the average time (sec) are presented.

Metric	DQN	A2C	TRPO	PPO
Success Rate [%]	81.50	89.90	**91.30**	90.10
Average Time (sec)	16.12	13.97	14.75	**12.45**

The sequence of events depicted in Figure 6.10 illustrates the behaviour of the agent in this use case. As the ego vehicle approaches the intersection, it stops due to the presence of an adversarial vehicle in its path (adversary 3). It remains stationary until another adversarial vehicle (adversary 4) begins to turn and exit the roundabout, creating a sufficient gap for the ego vehicle to merge the roundabout safely.

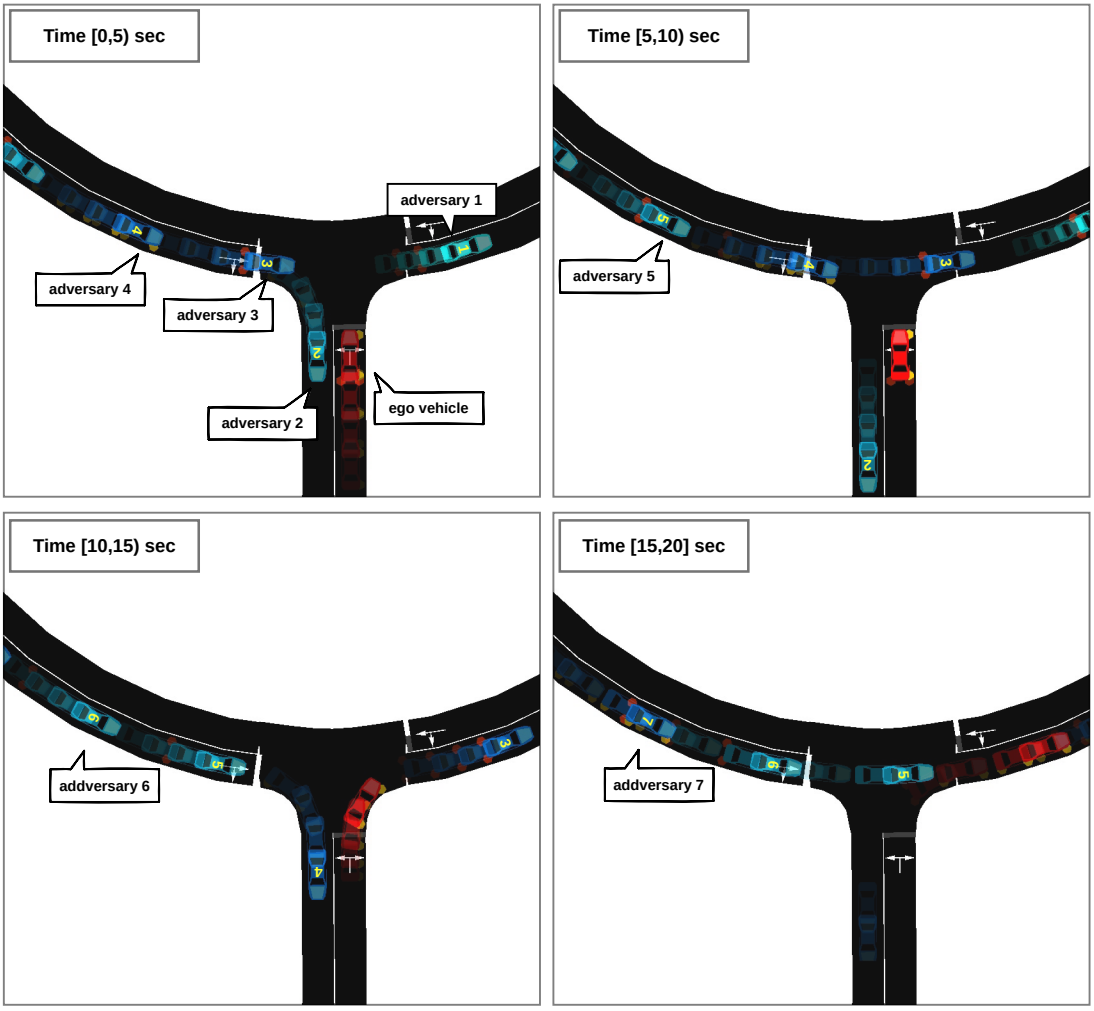


Figure 6.10: A bird's-eye view of a simulated episode in SUMO is presented, describing four scenes from the same episode at five-second intervals. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded. This temporal representation illustrates the behaviour of the TRPO agent within the roundabout scenario.

6.4. Merge Scenario

Our study also focuses on merge intersections. While these might seem simple at first place, their complexity lies in the high traffic density and the adversarial intentions. In these urban merge scenarios, the adversarial behaviour varies; some may be cooperative and yield, while others may not. Therefore, the agent's challenge is to accurately predict the actions of these vehicles to navigate the intersection effectively.

6.4.1. POMDP formulation

State

The state of a vehicle is described by three parameters: its distance to the intersection point, its longitudinal velocity, and its cooperation driving intention, represented as $s_i = (d_i, v_i, i_i)$. These parameters are normalized within the range of $[0, 1]$. There are two

categories of adversarial vehicles in this scenario depending on their cooperation level: the first type consistently yields and the second type always proceeds without yielding. These two behaviours are represented in Figure 6.11a, where the state is defined as:

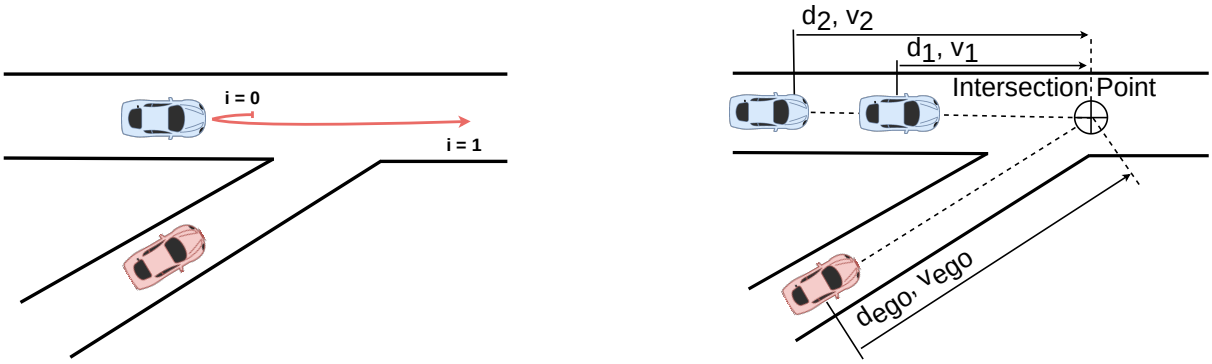
$$s = (s_e, s_1, s_2, ..., s_n) \tag{6.7}$$

Observation

As introduced in Section 6.3.1, the ego vehicle is capable of observing the longitudinal velocity and position of the surrounding vehicles. The observation vector is defined as:

$$\Omega = (d_e, v_e, d_1, v_1, d_2, v_2) \tag{6.8}$$

where it includes only the ego vehicle and the two nearest adversarial vehicles. Although both the roundabout and the merge scenario utilize the same observation vector, they differ significantly in terms of road structure and behaviour of adversarial vehicles. Therefore, we developed these two scenarios to reflect the variety of situations that can be found in an urban environment. A one lane merge scenario is presented in Figure 6.11b.



(a) (b)

Figure 6.11: The state and observations representations of the merge scenario. (**a**) The ego vehicle (red) is approaching the merge intersection. The adversary (blue) have two possible behaviours: yield (i=0) or continue (i=1). (**b**) A representation of the observation vector within the merge scenario. The observation vector is defined by the ego vehicle's (red) distance to the intersection and its velocity. The two closest adversaries' (blue) distances and velocities are also considered in the observation matrix.

Action

We also consider two high-level actions: one action is to stop and yield, and the other is to merge into traffic. The action space is thus defined as:

$$a = (stop, drive) \tag{6.9}$$

Reward

The reward function is defined in the same manner as detailed in Section 6.3.1, with the desired behaviour remaining similar across all intersection scenarios.

6.4.2. Experiments

In the merge scenario, traffic density is enhanced by increasing the frequency of traffic flow generation, with an adversarial vehicle appearing every one to three seconds. The ego vehicle always enters the intersection in the same lane. This scenario emphasizes the cooperation of adversarial vehicles, leading to the categorization of these vehicles into three types, each exhibiting a distinct level of cooperation. The speed of these adversarial vehicles is maintained at 5 to 15 m/s (18 to 54 km/h). A graphical representation of this scenario is shown in Figure 6.12.

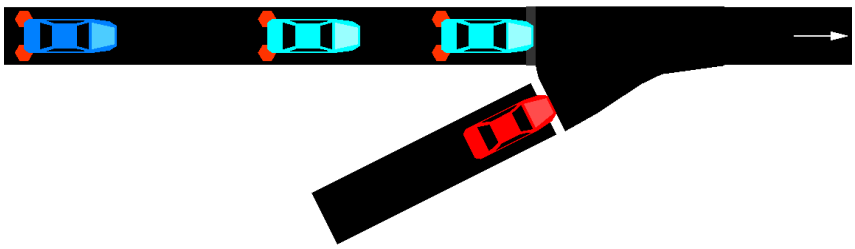


Figure 6.12: A bird-eye-view of the simulation framework in SUMO. The ego vehicle (red) is entering the intersection. The colour intensity of the adversarial vehicles (blue) is related to their intention. The clear yields and the dark does not yield.

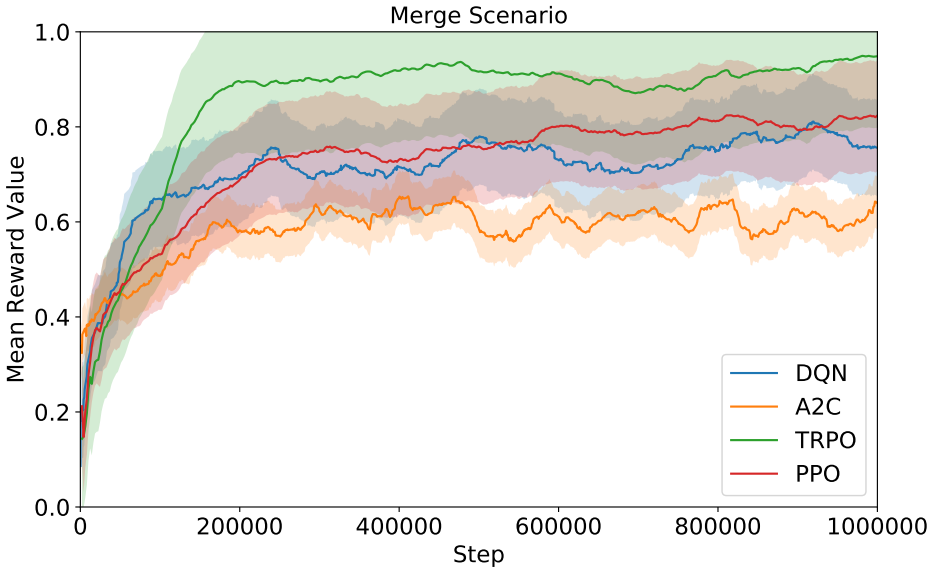


Figure 6.13: Evolution of the mean rewards during the training process for the DRL agents in the merge scenario: DQN(blue), A2C(orange), TRPO(green) and PPO(red).

In Figure 6.13, the progression of the training process is depicted. The TRPO agent emerges as the top performer, with the PPO and DQN agents demonstrating performances that are also competitive. In contrast, the A2C agent exhibits a notably poorer result.

The evaluation of the agents was conducted over 1000 new episodes with the NNs weights obtained in the training phase, with the outcomes detailed in Table 6.7. 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, which also marks it as the safest among the agents. Consequently, this agent has been selected for integration into the DM hybrid system.

Table 6.7: A comparison of the four DRL agents in the roundabout scenario. The success rate [%] and the average time (sec) are presented.

Metric	DQN	A2C	TRPO	PPO
Success Rate [%]	82.1	82.8	**92.90**	86.70
Average Time (sec)	**5.81**	5.61	8.84	7.18

The sequence depicted in Figure 6.14 demonstrates the agent's behaviour. As the ego vehicle nears the intersection, it stops because of an adversarial vehicle (adversary 2) ahead. It waits until this vehicle has passed and a sufficient gap appears before the next adversarial vehicle (adversary 3) arrives at the intersection. Then, the ego vehicle utilizes this gap to merge onto the road.

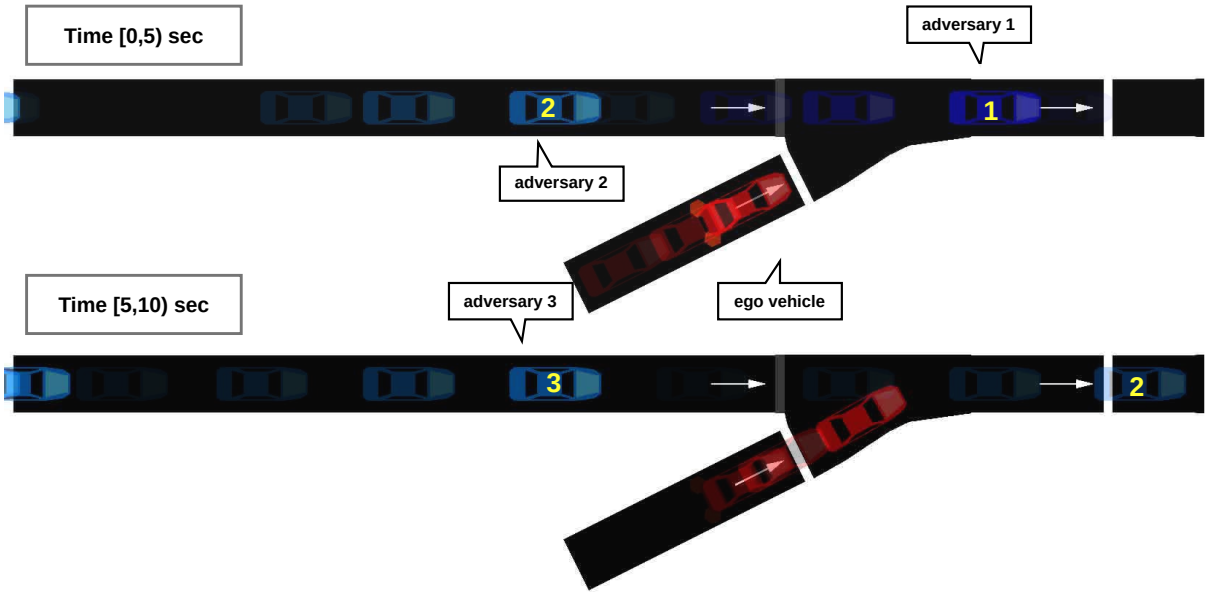


Figure 6.14: A bird's-eye view of a simulated episode in SUMO is presented, describing two scenes from the same episode at five-second intervals. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded. This temporal representation illustrates the behaviour of the TRPO agent within the merge scenario.

6.5. Crossroad Scenario

The last intersection scenario is the most challenging one, the crossroad intersection. In this scenario adversarial vehicles come from both right and left lanes when the ego vehicle approaches the intersection. We have defined this scenario as an unsignalized and uncontrolled intersection, in which no one vehicle has the right to cross. The ego vehicle must learn when to cross finding the gaps between the adversarial vehicles, which may continue forward or take any of the sides exits.

6.5.1. POMDP formulation

State

The state of a vehicle is defined by three parameters: its distance to the intersection point, its longitudinal velocity, and its driving intention, denoted as $s_i = \{d_i, v_i, i_i\}$. These physical values are normalized to fall within the range $[0, 1]$. The driving intention i of the adversarial vehicles, ranging from $[0, 2]$, is related to their intended route. Specifically, adversaries turning right are marked with $i = 0$, those continuing straight have $i = 1$, and those turning left are assigned with $i = 2$. These intentions are illustrated in Figure 6.15a. The overall state of the environment is represented as:

$$s = (s_e, s_1, s_2, \ldots, s_n) \tag{6.10}$$

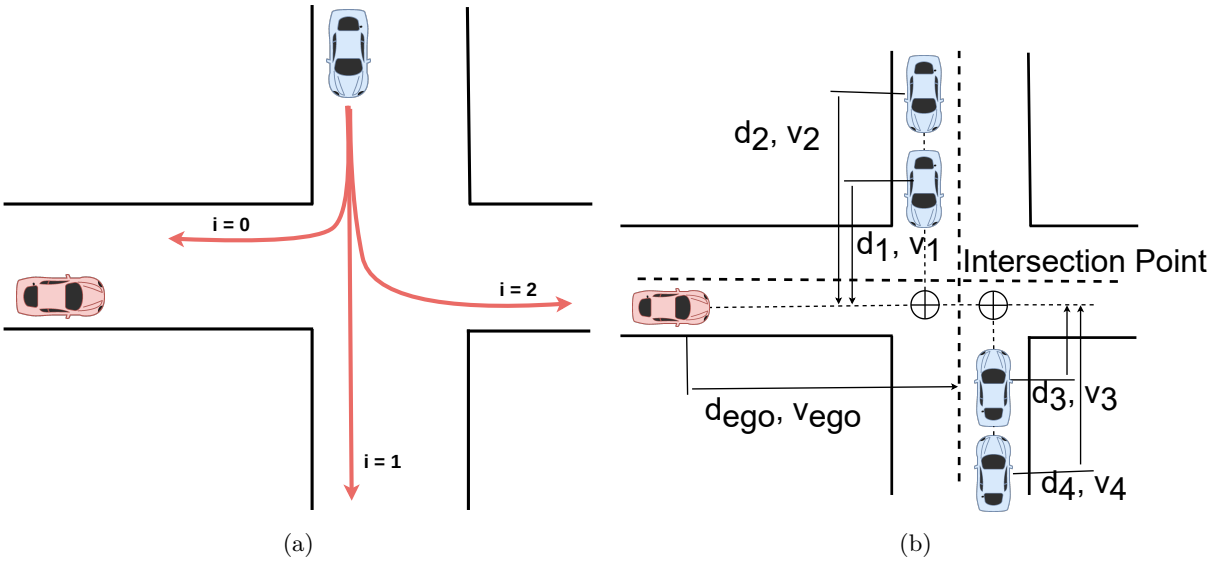


Figure 6.15: The state and observations representations of the merge scenario. (**a**) The ego vehicle (red) is approaching the crossroad intersection. The adversary (blue) have three possible behaviours: turn right (i=0), continue straight (i=1) or turn left (i=2). (**b**) A representation of the observation vector within the crossroad scenario. The observation vector is defined by the ego vehicle's (red) distance to the intersection and its velocity. The four closest adversaries' (blue) distances and velocities are also considered in the observation matrix.

Observation

According to Section 6.3.1, the ego vehicle is able to observe the longitudinal velocity and position of the nearby surrounding vehicles. In this particular scenario, where more than two vehicles are considered, the observation vector is defined as:

$$\Omega = (d_e, v_e, d_1, v_1, \ldots, d_4, v_4) \tag{6.11}$$

encompassing the observations of the ego vehicle and the two closest adversarial vehicles in each lane (right and left). This is visually represented in Figure 6.15b.

Action

Once more, we consider two high-level actions: one is to stop, and the other is to cross the intersection. Accordingly, the action space is defined as:

$$a = (stop, cross) \tag{6.12}$$

Reward

The reward function is defined in the same manner as detailed in Section 6.3.1, with the desired behaviour remaining similar across all intersection scenarios.

6.5.2. Experiments

We have developed a crossroad environment in SUMO comprising two intersecting roads, each with two lanes. A visual depiction of this scenario is provided in Figure 6.16.

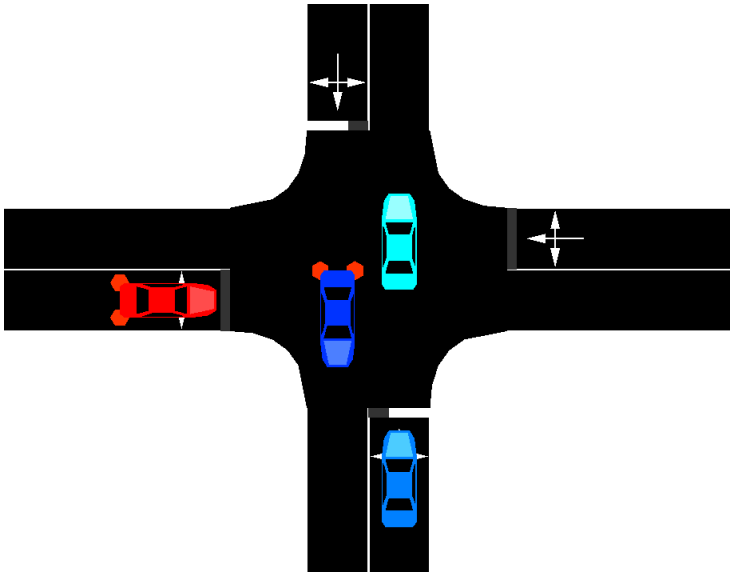


Figure 6.16: An overhead perspective of the SUMO simulation environment. The central vehicle (red) approaches the junction. The adversarial vehicles (in blue) vary in colour intensity according to their velocities, with darker shades indicating higher velocities.

The ego vehicle approaches the intersection, encountering vehicles from both sides. An adversarial vehicle is generated randomly every three to five seconds appearing in each lane. These vehicles are capable of reaching speeds ranging from 5 to 15 m/s (18 to 54 km/h) and follow predefined randomly generated routes, including turning to either side or continuing straight.

Figure 6.17 illustrates the training process progression for the agents. The TRPO agent stands out as the most effective performer. 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 compared to its counterparts.

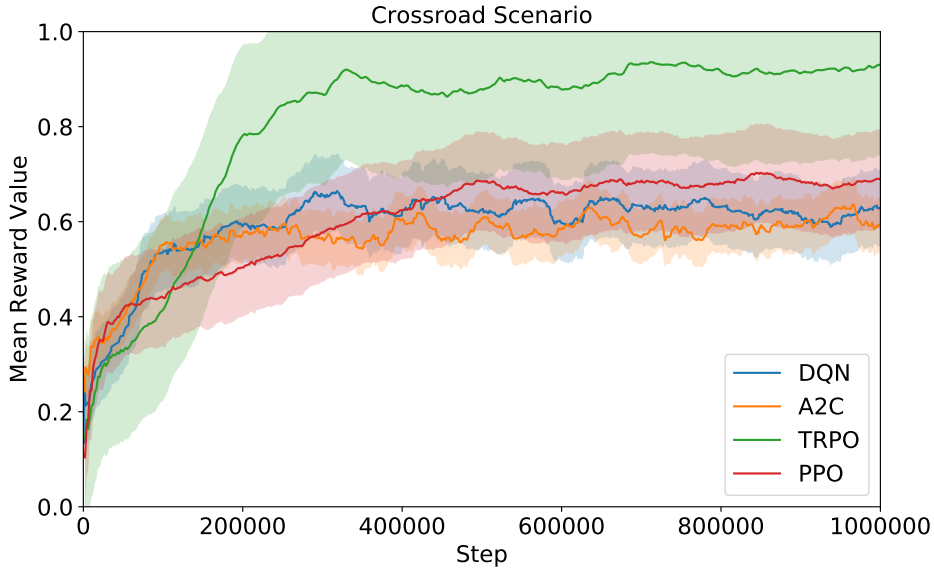


Figure 6.17: Evolution of the mean rewards during the training process for the DRL agents in the crossroad scenario: DQN(blue), A2C(orange), TRPO(green) and PPO(red).

After the training phase, the agents are evaluated over 1000 new episodes, with results summarized in Table 6.8. The DQN agent emerged as the fastest, closely followed by the A2C in terms of average completion time. However, these agents recorded lower success rates compared to the PPO and TRPO agents. Notably, the TRPO agent distinguished itself by achieving the highest success rate at 99.2% with a competitive average time, thereby establishing itself as the safest option among the evaluated agents. As a result, the TRPO agent is selected for integration into the DM hybrid system.

Table 6.8: A comparison of the four DRL agents in the roundabout scenario. The success rate [%] and the average time (sec) are presented.

Metric	DQN	A2C	TRPO	PPO
Success Rate [%]	88.5	94.60	**99.20**	97.30
Average Time (sec)	**11.13**	11.76	13.87	12.81

The sequence of events depicted in Figure 6.18 illustrates the behaviour of the agent. As the ego vehicle approaches the intersection, it stops due to the presence of an adversarial vehicles (adversaries 1 and 2) in its path. It remains stationary until it identifies that adversary 3 is turning, creating a sufficient gap for the ego vehicle to merge the crossroad safely.

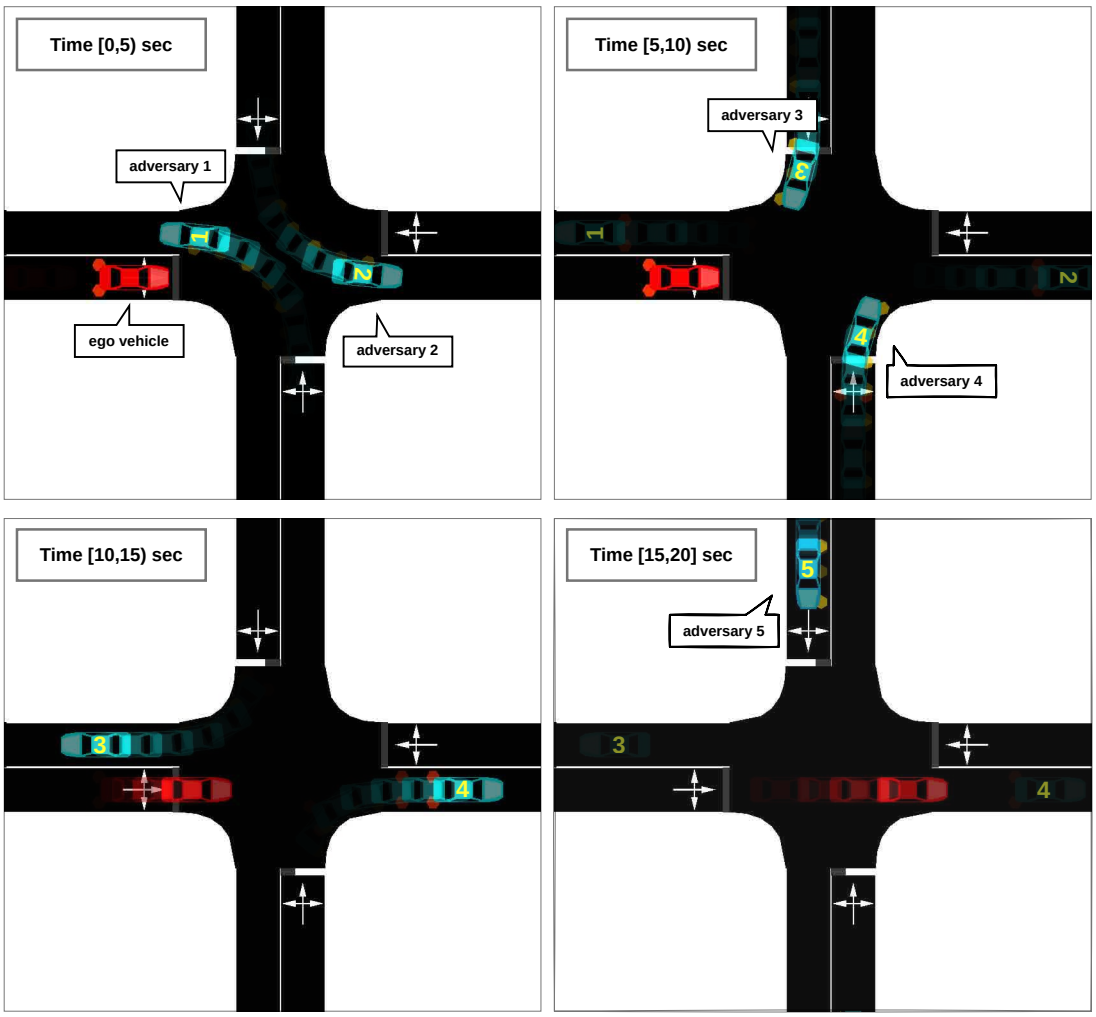


Figure 6.18: A bird's-eye view of a simulated episode in SUMO is presented, describing four scenes from the same episode at five-second intervals. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded. This temporal representation illustrates the behaviour of the TRPO agent within the crossroad scenario.

6.6. SOTA comparison

In previous sections is shown that the proposed POMDP formulation and the DRL agents are effective for solving urban scenarios. However, it's important to remark that these scenarios were specifically tailored for our approach. For a fair comparison of the performance of our proposal with the SOTA, we have conducted a study in SMARTS scenarios.

6.6.1. SMARTS Scenarios

The 'Driving SMARTS 2022' benchmark, launched in the NeurIPS 2022 Driving SMARTS competition [135], offers a variety of scenarios to evaluate DRL proposals for AD. For our purposes, we have selectively focused on four scenarios aligned with the ones we have developed, specifically targeting our urban environment needs. A visual representation of these scenarios is provided in Figure 6.19.

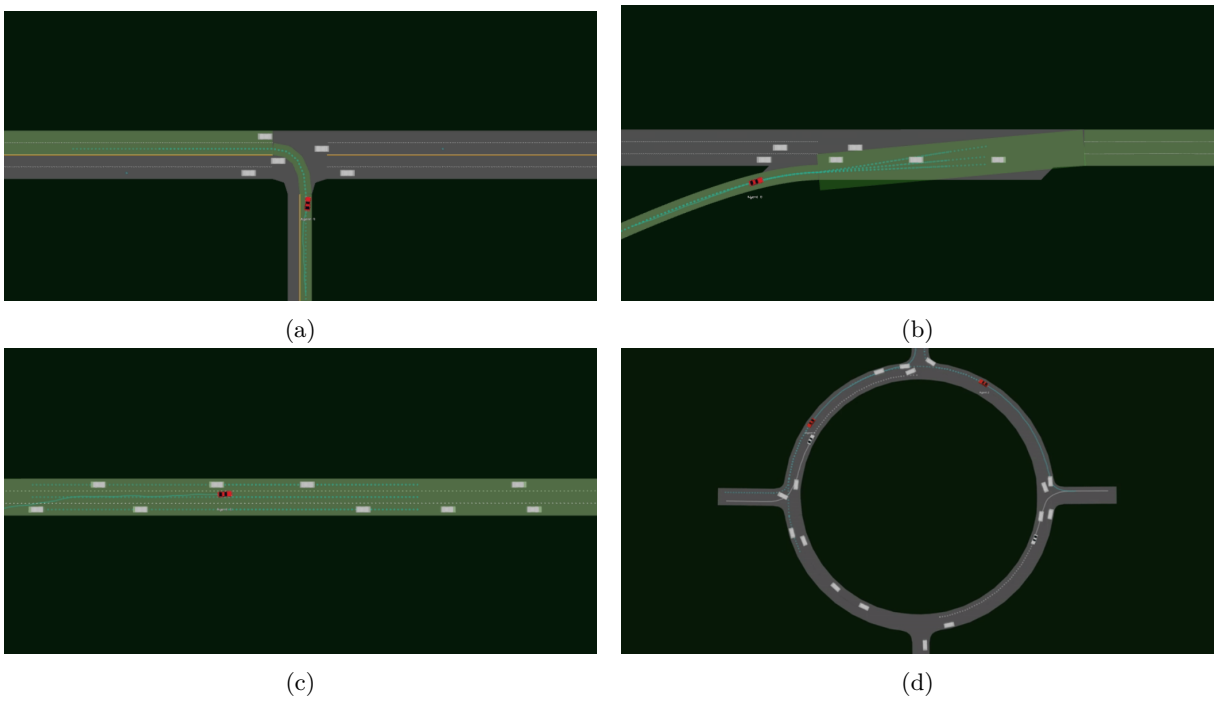


Figure 6.19: The designed scenarios for a comparison with state-of-the-art methods within the SMARTS framework. **(a)** Unprotected left turn. **(b)** Three lane merge. **(c)** Three lane road. **(d)** Roundabout.

In the four 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. The selected scenarios present the following characteristics:

- **Unprotected Left Turn:** The scenario is a T-intersection, where the road travelled by the ego vehicle comprises one lane in each direction, intersecting with a perpendicular road that boasts two lanes in each direction. As the ego vehicle approaches this unprotected intersection, it must execute a left turn, requiring it to merge into the traffic flow. We use our **crossroad** agent for this scenario.
- **Three Lane Merge:** The scenario is generated in a three-lane merge environment, characterized by a primary road that merges into a three lane road. The challenge for the ego vehicle lies in identifying the optimal moment and lane for merging. We evaluate our **merge** agent in this scenario.

- **Three Lane Road:** In this scenario, the ego vehicle is in a three-lane road, tasked with overtaking slower-moving adversarial vehicles. The environment is designed to simulate a dense traffic condition. We use our **lane change** agent for this scenario.

- **Roundabout:** The roundabout scenario features a circular intersection with multiple exits, requiring the ego vehicle to navigate through dynamically entering and exiting traffic. The ego vehicle must decide the optimal time to enter and exit the roundabout, considering the positions and velocities of other vehicles within the roundabout. We evaluate our **roundabout** agent in this scenario.

6.6.2. Experiments

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 SMARTS with two representative methods of the SOTA: [85] employs a Transformer-based scene representation alongside an actor-critic DRL approach (also described in Section 3.4.1). Moreover, [136] is based in three ingredients, namely expert demonstration (a human expert demonstrates their execution of the task), policy derivation (the imitative expert policy is derived using behavioral cloning and uncertainty estimation), and DRL (the imitative expert policy is utilized to guide the learning of the DRL agent). The comparative analysis presented in Table 6.9 underscores the superior performance of our framework against the other approaches . Our approach demonstrates exceptional efficiency, highlighting its robustness in handling various driving challenges using an uniform architecture along the scenarios.

Table 6.9: A comparison of our proposal with others frameworks found in the literature in SMARTS scenarios. The success rate [%] and the average time (sec) are presented.

Proposal	Metric	Left Turn	Merge	Road	Roundabout
Ours	Success Rate [%]	95.30	**98.40**	93.60	**91.70**
	Average Time (sec)	**12.22**	**21.9**	24.34	37.47
[85]	Success Rate [%]	94.00	96.00	-	76.00
	Average Time (sec)	12.50	28.60	-	56.60
[136]	Success Rate [%]	**96.00**	-	-	84.00
	Average Time (sec)	14.26	-	-	**36.62**

In the merge scenario, our framework outshines with a success rate of **98.40%**, surpassing the 96.00% achieved by [85] and excelling in a domain where [136] provides no comparative data. This significant margin underscores our framework's adeptness at navigating complex driving situations, ensuring safer and more reliable vehicle operations.

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 [85] and the 84.00% by [136], demonstrating our system's supe-

rior capability in managing the intricate dynamics and unpredictable nature inherent in roundabouts with a competitive ending time.

Efficiency, measured through the average completion time, further distinguishes our framework. Notably, in the merge scenario, it accomplishes the task in **21.9 seconds**, markedly quicker than the 28.60 seconds required by [85], showcasing our framework's proficiency in executing manoeuvrers swiftly without compromising on safety or reliability.

Although [136] achieves a marginally higher success rate in the left turn scenario, our framework maintains competitive success rates across all scenarios while consistently offering more efficient manoeuvrer execution. This balance of high success rates coupled with reduced task completion times positions our framework as a highly effective and efficient solution for AD in diverse and dynamic environments.

6.7. Summary

This chapter studies the implementation of RL for DM in urban driving scenarios, focusing on crossroads, merges, roundabouts, and lane changes. By formulating these scenarios as four different POMDPs, we divide the DM process into separate tasks, providing a structured approach to understanding and solving the complex dynamics of urban driving. Utilizing SUMO simulator, we implement a comprehensive framework that integrates RL algorithms to control the ego vehicle's high-level actions.

The core of our methodology lies in defining observation and action spaces tailored to each scenario, coupled with a reward function designed to encourage optimal driving behaviours. This setup allows the ego vehicle to learn policies for safely navigating urban environments, as demonstrated by the training process and performance metrics such as success rate and average episode duration.

Among the RL algorithms evaluated, TRPO consistently outperforms the other proposals (DQN, A2C, PPO) in terms of success rate across different scenarios, highlighting its efficacy in learning efficient driving policies. This is further evidenced by the comparative analysis in the SMARTS scenarios, where our approach not only meets but often exceeds the performance benchmarks set by existing frameworks in the literature.

Chapter 7

Hybrid Decision Making Architecture Experimental Results

Cuando salgas de la tormenta, no serás la misma persona que entraste. Ese es el objetivo de esta tormenta.

Haruki Murakami

7.1. Introduction

Throughout this dissertation, the author encountered difficulties in validating a full AD system. Various simulation platforms and AD challenges were discussed in Chapter 4. However, these benchmarks did not primarily concentrate on the real-world implementation of an AD stack based on RL. This chapter presents three main objectives: Firstly, to offer an open-source validation framework for RL based on CARLA simulator, incorporating vehicle dynamics to reduce the RG. The main advantage of this framework lies in its flexibility to define observation and action spaces, coupled with simple yet efficient coding and comprehensible documentation. Secondly, to evaluate our hybrid DM approach within the AD stack of the Robesafe group by presenting it as a baseline in this validation framework. This includes to use not only RL metrics but also comfort and safety ones. Finally, introducing a further step towards real-world application by presenting a DT of our university Campus and our real vehicle. We validate our architecture through a PI, synchronously executing both virtual and real implementations.

Four main sections are outlined in this chapter: The first section details the integration of the OpenAI Gym framework with the simulator. The second section underlines in the process of defining scenarios using the CARLA PythonAPI and introduces the metrics for evaluating RL-based AD architectures. In the third section we describe our AD stack, including an evaluation of our approach. The last section, introduces our DT with a PE implementation as a way to short the gap from simulation to real world.

7.2. OpenAI Gym Implementation

The initial step for realistically validating our approach involves creating scenarios within the CARLA simulator using OpenAI Gym [15] libraries. The essential steps for crafting a custom Gym environment are delineated in the official Gym documentation. While an environment is typically perceived as a black box that receives actions and provides observations and rewards, this section delves into the implementation of an environment and the methodology employed for its integration with our simulation framework.

The framework is implemented in Python, where a class is established encompassing various methods. For a custom Gym environment to be operational, three main methods are essential: **init**, **reset** and **step**. Additionally, we have defined several methods necessary for interaction with the simulator. The architecture of our custom Gym environment comprises the following methods.

- **init:** It is responsible for initializing the environment and performing all necessary initial setups. Various settings are available for customization: rendering can be toggled between true and false; different observation modes can be chosen; and the specific scenario for evaluation or training can be selected. These scenarios are described in detail in the subsequent section. The ego vehicle is created in this method.
- **reset:** It is called when the current episode finishes. Its primary function is to reset the position of the ego vehicle to its original state and to reinitialize most of the variables utilized in calculations.
- **step:** This method receives an action as an input argument. It moves the simulation through four key steps. Initially, it executes the selected action. Subsequently, it performs a simulation tick. The third step involves calculating the reward for the current action and determining if the episode has concluded. If the episode is finished, a *done* signal is set to true. Lastly, it computes the new observation vector. This method returns the observations, the calculated reward, the *done* signal, and an auxiliary variable called *info*, a common implementation in Gym environments, that contains additional information.
- **set action:** It serves as an interface module between the selected action and the ego vehicle. Three variables can be adjusted in this method: the ego vehicle's throttle, steer, and brake. The vehicle's throttle and brake are normalized between [0,1], while the steer is normalized between [-1,1].
- **tick:** It directly initiates a tick in the simulated world at a frequency of 0.01 seconds. Additionally, it is responsible for the creation and removal of adversarial vehicles.
- **get reward:** It checks the status of the ego vehicle and calculates a reward based on this information. In the basic implementation, a positive reward is assigned if

the vehicle reaches the end of the episode without a collision; conversely, a collision results in a negative reward.

- **get observations:** This method is responsible for extracting the observations from the simulation. As previously mentioned, various modes can be chosen during initialization. Specifically, three main modes have been developed in this work. The first mode is a low-dimensional observation vector containing the positions and velocities of the surrounding vehicles. The second mode provides a front camera image of the ego vehicle. The third mode is a bird-eye-view image of the simulation centred on the ego vehicle. This last mode has been implemented using *carla-birdeye-view* [137]. Additionally to these three options, a list of waypoints corresponding to the route to be followed is provided.

The Gym environment flowchart illustrates the process of an episode at a low level in Figure 7.1. Initially, the environment is set up by the **init** method. At the beginning of a new episode, the **reset** method reinitializes the variables. During each simulation step, the **step** method invokes the remaining methods that interact with the simulator using the PythonAPI, returning the pertinent information. Upon the conclusion of an episode, a new one can start.

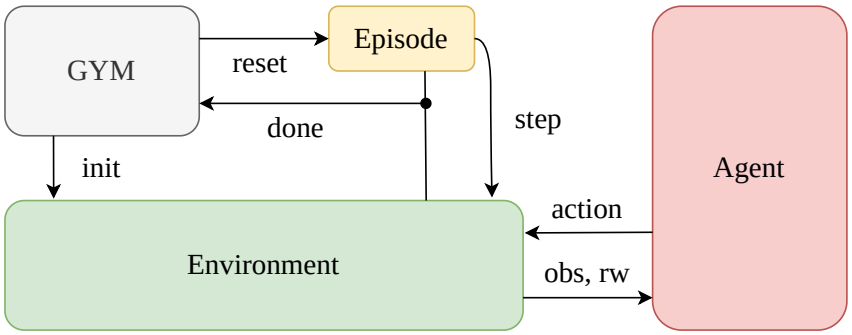


Figure 7.1: At the start of a new episode, the **reset** method reinitializes the variables. During each simulation step, the **step** method invokes the remaining methods that interact with the simulator using the PythonAPI, returning the pertinent information. Upon the conclusion of an episode, a new one can start.

7.3. Urban Scenarios for Reinforcement Learning in CARLA

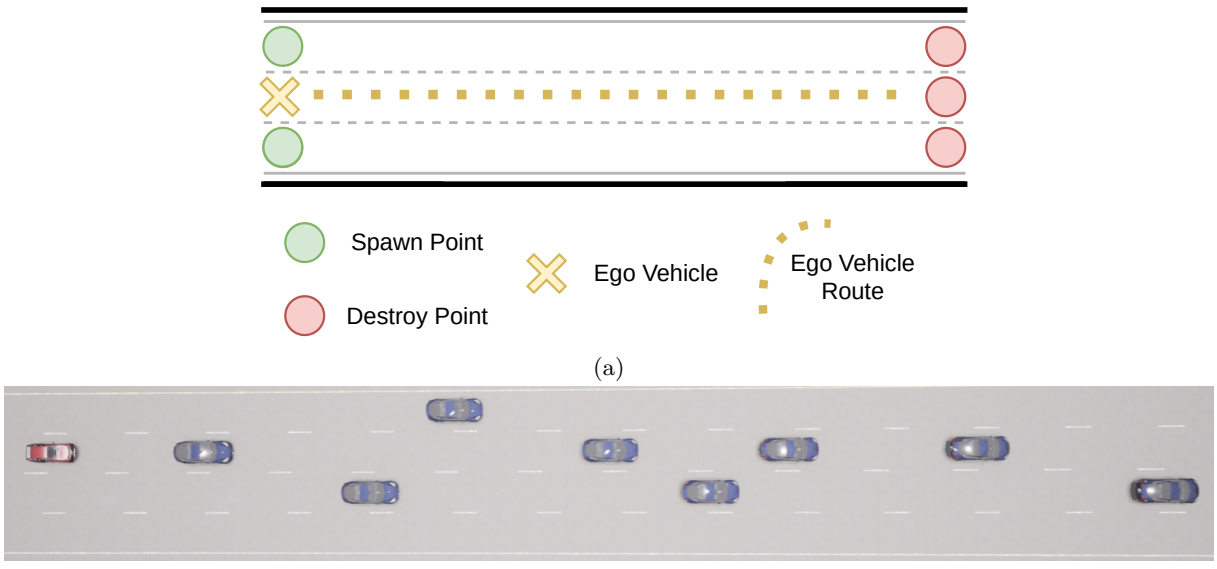
Following the CL strategy, four single use cases are defined according to the four scenarios trained in SUMO. Additionally, a concatenated use case scenario is established to facilitate the evaluation of a complete AD stack in a continuous driving scenario. In all cases, a predefined route is determined, with the objective being to reach the end of the scenario without any collisions.

7.3.1. Single use case scenarios

These scenarios are developed to provide realistic and uncontrolled urban driving situations. 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 in CARLA. Initially, an ideal spot is identified on the provided towns, 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, their velocities, and their levels of cooperation, in a similar way to SUMO. 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. A representation of how these scenarios are generated, and how the vehicles move within them, is detailed in the following sections for the single use cases.

Lane Change

We define a lane change scenario in the *Town04* map of CARLA, featuring a 400-meter-long road with four lanes. Vehicles are generated every three to five seconds at the beginning of the road and may change lanes or continue straight ahead. Upon reaching the road's end, these vehicles are destroyed. They travel at velocities ranging from 5 to 15 m/s (18 to 54 km/h). The traffic generation and the vehicles' behaviour in this scenario are illustrated in Figure 7.2.



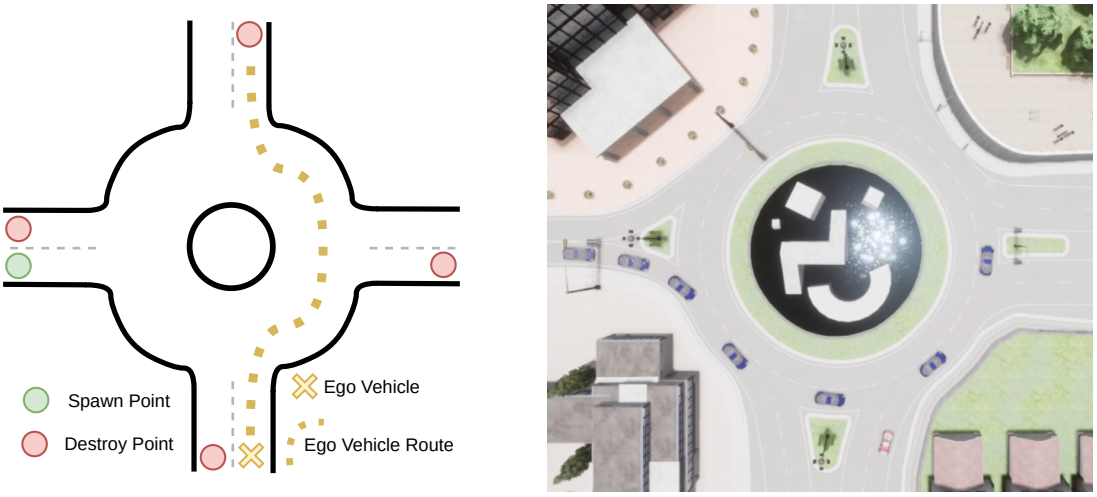
(a)

(b)

Figure 7.2: The lane change scenario in CARLA includes: (**a**) A visual representation of the traffic flow, featuring spawn points (green), destroy points (red), and the initial location of the ego vehicle (yellow). (**b**) A bird-eye-view of the scenario in CARLA.

Roundabout

The roundabout scenario is generated in the *Town03* map of CARLA, featuring a roundabout with a radius of 30 meters. Adversarial vehicles are generated every three to five seconds, sufficiently distanced from the intersection point, before they proceed to navigate into the roundabout. These vehicles may exit the roundabout either before or after the ego vehicle's entry point, travelling at velocities ranging from 5 to 15 m/s (18 to 54 km/h). Upon reaching an exit, they are destroyed. The ego vehicle, spawned outside the roundabout, must enter and exit following the predefined route without colliding. An overview of these traffic flows and the scenario is presented in Figure 7.3.

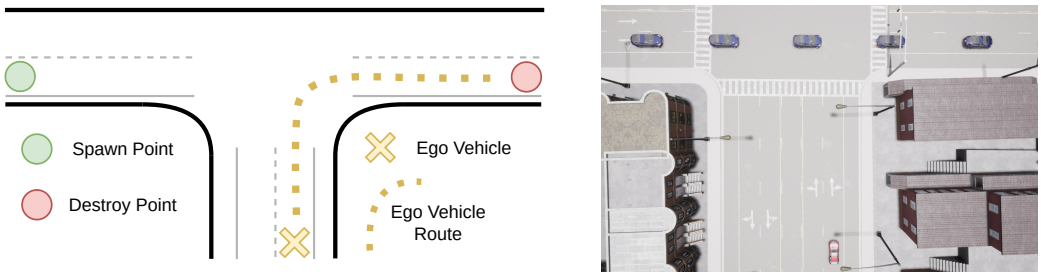


(a) (b)

Figure 7.3: The roundabout scenario in CARLA includes: (**a**) A representation of the traffic flow as described in Figure 7.2. (**b**) A bird-eye-view of the scenario in CARLA.

Merge

The merge scenario is also defined in the *Town03* map of CARLA. Adversarial vehicles are generated on a 60-meters lane perpendicular to the ego vehicle's lane every three to five seconds. These vehicles travel at velocities ranging from 5 to 15 m/s (18 to 54 km/h) and are subsequently destroyed when they reach the end of the scenario. The objective in this scenario is to navigate to the endpoint without any collisions. The scenario is depicted in Figure 7.4.

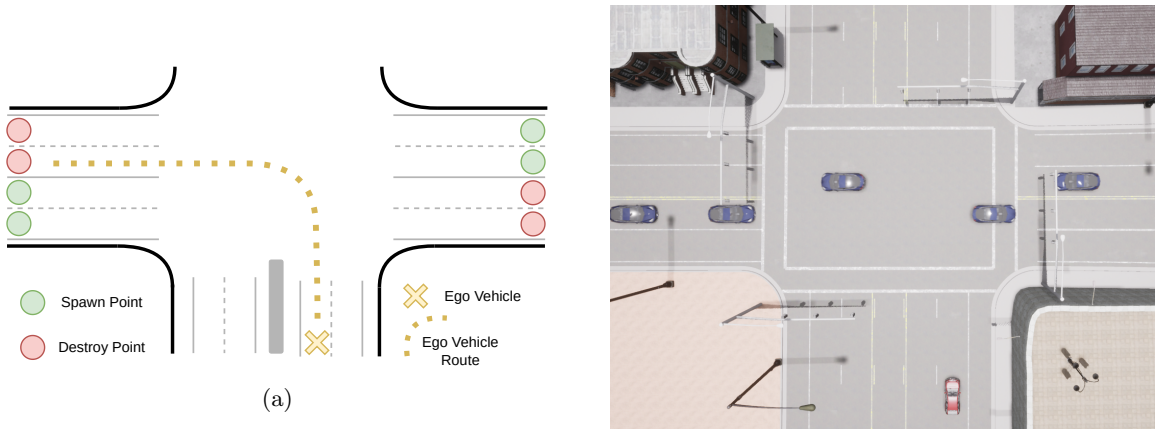


(a) (b)

Figure 7.4: The merge scenario in CARLA includes: (**a**) A representation of the traffic flow as described in Figure 7.2. (**b**) A bird-eye-view of the scenario in CARLA.

Crossroad

A crossroad intersection scenario is established in the *Town03* map of CARLA, with roads extending 50 meters in length. Adversarial vehicles are generated on both sides of the intersection every three to five seconds and may either proceed straight or turn at the intersection. These vehicles travel at velocities ranging from 5 to 15 m/s (18 to 54 km/h). The ego vehicle is required to follow the trajectory shown in Figure 7.5 and reach the end without colliding. A bird-eye-view of the scenario is also presented in this figure.



(a) (b)

Figure 7.5: The merge scenario in CARLA includes: (**a**) A representation of the traffic flow as described in Figure 7.2. (**b**) A bird-eye-view of the scenario in CARLA.

7.3.2. Concatenated use case scenario

We develope a scenario wherein the ego vehicle navigates through diverse urban environments under varying driving conditions. The scenario is set in the Town03 map, chosen for its variety of use cases. To ensure that the ego vehicle encounters complex situations, the methodology for spawning and destroying vehicles mirrors that used in the individual use cases. This scenario enable a deeper and more qualitative understanding of the AD stack's performance. The scenario combines the different use cases previously presented. 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. Traffic within each scenario segment is generated in a consistent manner with their individual descriptions, with the addition of randomly placed vehicles throughout the route to simulate a more realistic urban environment. The complete scenario is depicted in Figure 7.6.

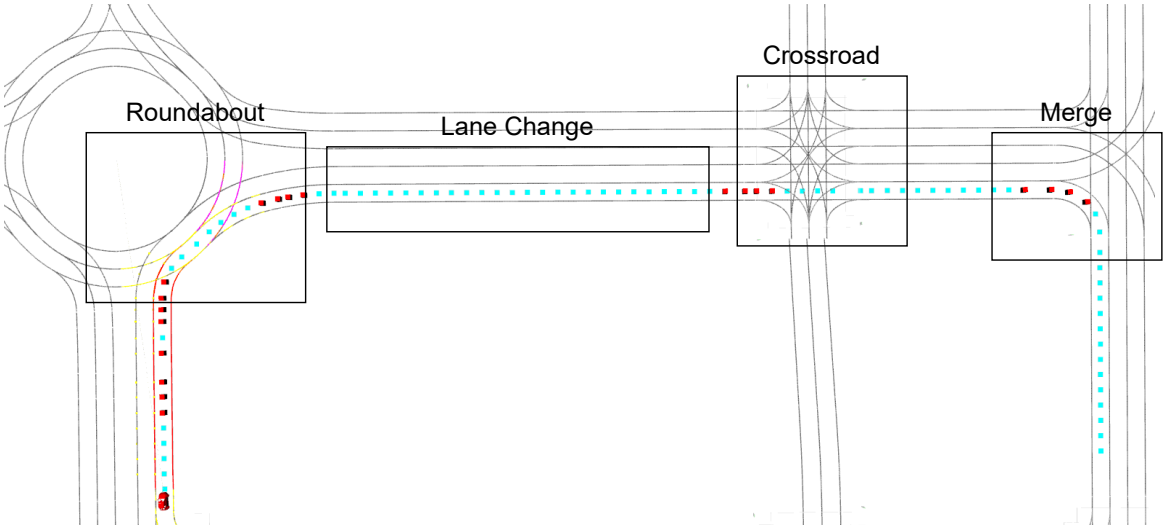


Figure 7.6: Concatenated Scenario: The vehicle initially navigates through a roundabout, subsequently approaching a two-lane road populated with slow-moving vehicles. This is followed by a crossroad and, ultimately, a merge intersection. The blue dots represent the path points to be followed, while the red points signify the tactical trajectory's use case indicators.

7.4. Evaluation of our Autonomous Driving Stack

In this section, we evaluate the performance of our AD stack, which includes our hybrid DM architecture, within the CARLA simulator, adhering to the hybrid methodology introduced in Chapter 4. Each component of the architecture operates independently but is integrated through a ROS interface, culminating in a full AD system. Information exchange between our framework and the simulator is managed through a ROS bridge, which converts the data acquired via the PythonAPI into ROS topics. Internal communication among the modules is done through ROS. An overview of our ROS-integrated AD architecture is illustrated in Figure 7.7.

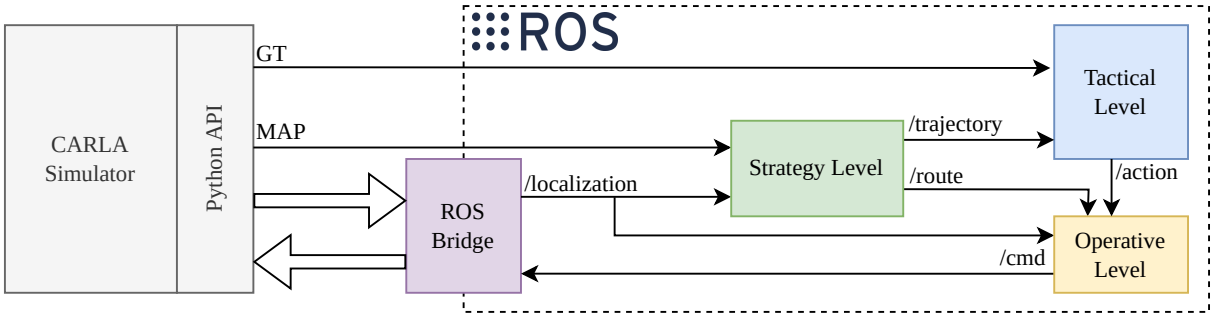


Figure 7.7: Overview of the integration of the AD stack within the ROS framework and CARLA. Vehicle and map information are directly extracted via PythonAPI, while ROS nodes handle internal communications. The strategy level (green) generates trajectories, the tactical level (blue) selects actions based on the observations, and the operative level (yellow) generate the reference signals. These signals are then translated into vehicle movements by the ROS bridge using PythonAPI.

In our setup, the perception module directly gets ground truth information from the simulator, bypassing the need for real-world sensor data processing. This way, adversarial vehicles and map data is directly extracted using the PythonAPI, while additional data is communicated via ROS. The architecture's modules operate as ROS nodes. Localization data is published by the ROS bridge and is subscribed by both the strategy and operative levels. The strategy level generates the trajectories using the Python module Networkx [138] and publishes them. The tactical level, utilizing the trajectory and ground truth data, determines an action, which is then published on a ROS topic. The operative level, receiving the route, localization, and action data, generates control commands that are published on another topic. These commands are captured by the ROS bridge, which translates them into vehicle movements through the PythonAPI.

7.4.1. 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 in each episode.
- **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 in navigating the environment.
- **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 [139]. This operates under the management of the 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.

7.4.2. Single use case scenarios

Table 7.1 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.

Table 7.1: Performance metrics for lane change, roundabout, merge and crossroad. Success rate percentage, jerk dynamics, acceleration, time to complete manoeuvres, and average speed are evaluated to assess the efficiency, smoothness, and safety of the CARLA Autopilot and our AD stack.

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

The jerk dynamics, a critical metric for assessing the smoothness and comfort of the driving experience, 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, particularly in the crossroad scenario where it reaches 14.6 m/s^3 for the 95th percentile and 22.8 m/s^3 for maximum jerk. 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 compared to the CARLA Autopilot. The exception is the merge scenario, where both systems show similar performance. Time to complete manoeuvres and average speed metrics highlight the efficiency and agility of our AD stack. Our system completes scenarios in significantly less time than the CARLA Autopilot, demonstrating faster response and higher DM capabilities. Additionally, our AD stack maintains higher average speeds across scenarios, ensuring swift and efficient navigation through various driving scenarios.

In conclusion, while the CARLA Autopilot shows impeccable success rates due to its inherent advantages, our AD stack outperforms in key aspects of driving dynamics, including jerk, acceleration, time efficiency, and speed.

Therefore, we present a qualitative analysis for each scenario. This includes a bird's-eye view representation of the ego vehicle navigating various scenarios, alongside an analysis of our architecture control references and the actual signals obtained by the simulated vehicle. The analysis is based on three parameters. First, we present the linear velocity signals, measured in meters per second (m/s). Next, the steering angle signals are depicted, where measurements are the normalized values of wheel steering, ranging between [-1,1]. In these cases, the red signal represents the generated signal (target), and the blue one illustrates the vehicle's response (actual). Lastly, we present comfort metrics, with acceleration (in m/s^2) shown in blue and jerk (in m/s^3) in purple.

Lane Change Qualitative Results

A visualization of the control signals is illustrated in Figure 7.8. In second 3 a left lane change action is required by the tactical level and the operational level generates the corresponding target signal, the manoeuvre ends in second 5, returning the steering value to zero. In second 7 this manoeuvre is performed again, but this time changing to the right lane. The signals for lane changing are smoothly executed, ensuring the vehicle's actual response closely aligns with the target signal. Notably, the jerk and acceleration signals exhibit minimal variation throughout this scenario, indicating the vehicle's proficiency in maintaining its velocity while adeptly navigating around other traffic participants.

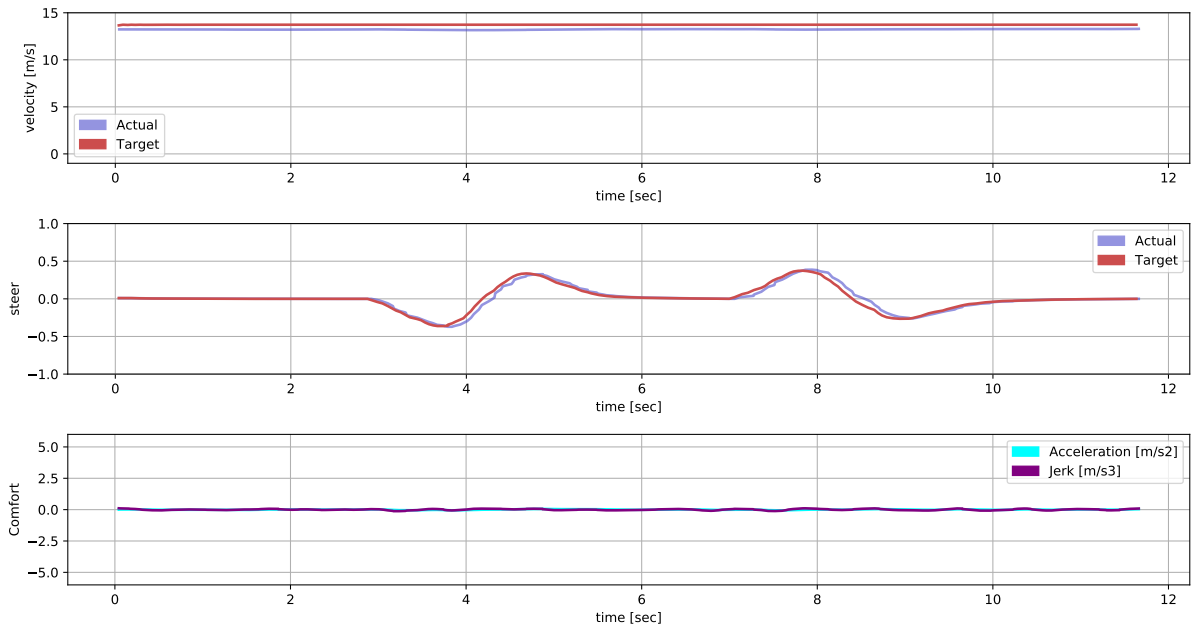


Figure 7.8: Control signals within the lane change scenario in CARLA. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.

The agent's behaviour is further illustrated in Figure 7.9, which demonstrates the vehicle's adept manoeuvring from the middle to the right lane to circumvent a slower ahead car, followed by a return to the middle lane upon identifying an opportune gap. In this illustration, the ego vehicle (red) navigates around adversarial vehicles (blue) in the following sequence: 1) It detects adversary 4 moving slowly in its lane, while adversary 3 occupies the adjacent right lane. 2) The ego vehicle initiates a lane change upon spotting a gap in the right lane. 3) With the right lane clear of traffic, the ego vehicle continues straight ahead. 4) Upon encountering adversary 6 ahead and noticing a viable gap in the left lane. 5) the ego vehicle successfully executes a left lane change, finally overtaking adversary 6.

Figure 7.9: A bird's-eye view of a simulated episode in CARLA is presented, describing various scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded.

Roundabout Qualitative Results

Figure 7.10 presents the control signals generated by the operative level within the roundabout scenario. Initially, a deceleration is generated when a vehicle is detected approaching (approximately at second 4), followed by the resuming of the nominal velocity (around second 5) to continue along the path. The vehicle successfully adheres to these commands. In this scenario, variations in the jerk and acceleration signals are more pronounced due to changes in velocity.

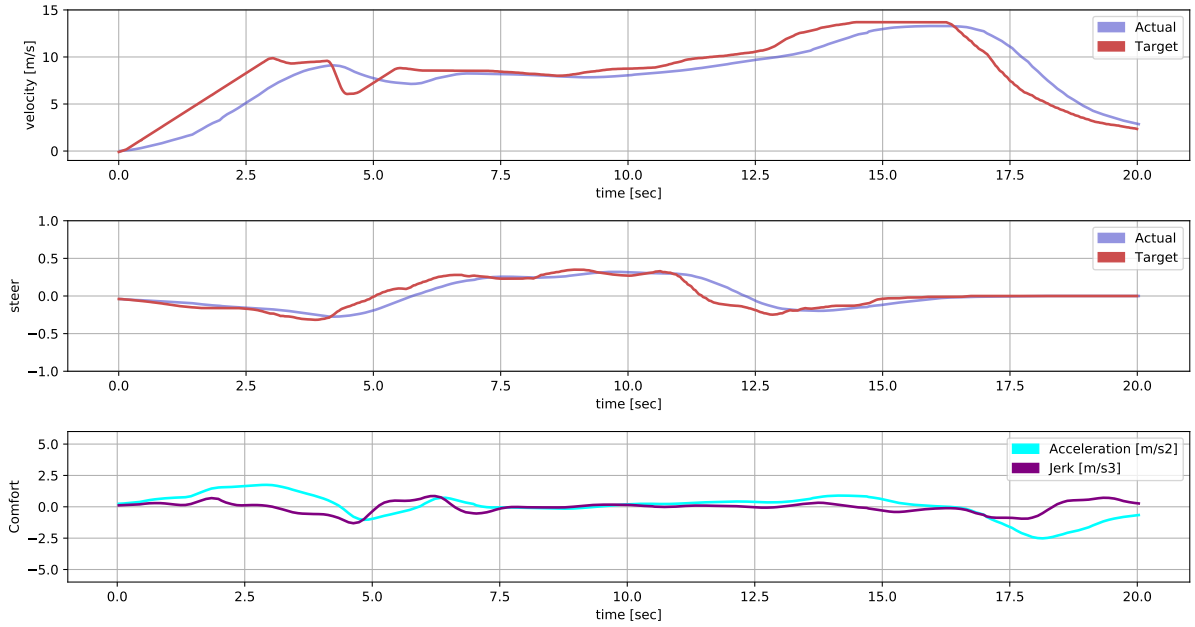


Figure 7.10: Control signals within the roundabout scenario in CARLA. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.

Figure 7.11 demonstrates the vehicle's manoeuvre as it merges into the roundabout upon identifying a suitable gap. The sequence of navigation around adversarial vehicles (depicted in blue) by the ego vehicle (in red) is as follows: 1) As the ego vehicle approaches the roundabout, the corresponding DRL agent is activated, utilizing tactical trajectory information. The maximum velocity is reduced, indicating entry into a roundabout. 2) The agent executes a stop signal in response to adversary 3's presence within the roundabout, causing the velocity to decrease. However, it quickly recognizes a gap due to the adversary 4 is exiting the roundabout and sends a drive signal. This action is observable both in the visual representation and the control graphics. 3) The ego vehicle enters the roundabout through the previously identified gap. 4) The ACC module adjusts the ego vehicle's velocity in relation to adversary 3, which is the leading vehicle. 5) As adversary 3 continues around the roundabout and the ego vehicle approaches its exit, the nominal velocity is resumed due to the absence of a leading vehicle (around second 13). 6) The ego vehicle successfully exits the roundabout, completing the scenario.

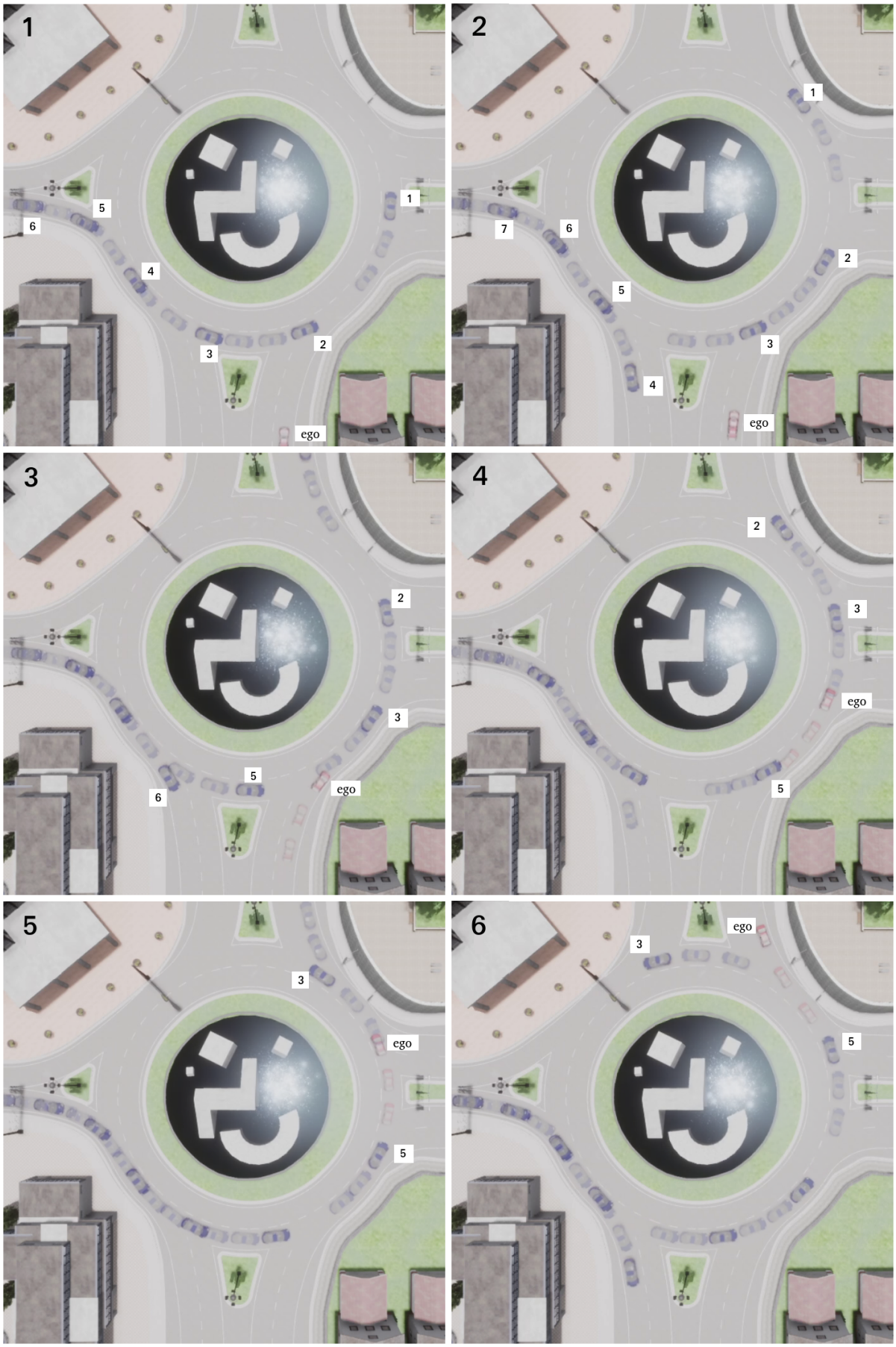


Figure 7.11: A bird's-eye view of a simulated episode in CARLA is presented, describing various scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded.

Merge Qualitative Results

Figure 7.12 illustrates the control signals during a merge scenario. A stop signal is initiated due to the presence of adversarial vehicles on the road, prompting the vehicle to halt. Subsequently, a command to resume nominal velocity is issued once a gap is detected, enabling the vehicle to accurately follow the target signals. The steering signal performs a turn right command for merging the lane. Comfort metrics in this scenario are comparable to those observed in previous scenarios, with acceleration slightly exceeding 2 m/s^2 and jerk also close to 2 m/s^3.

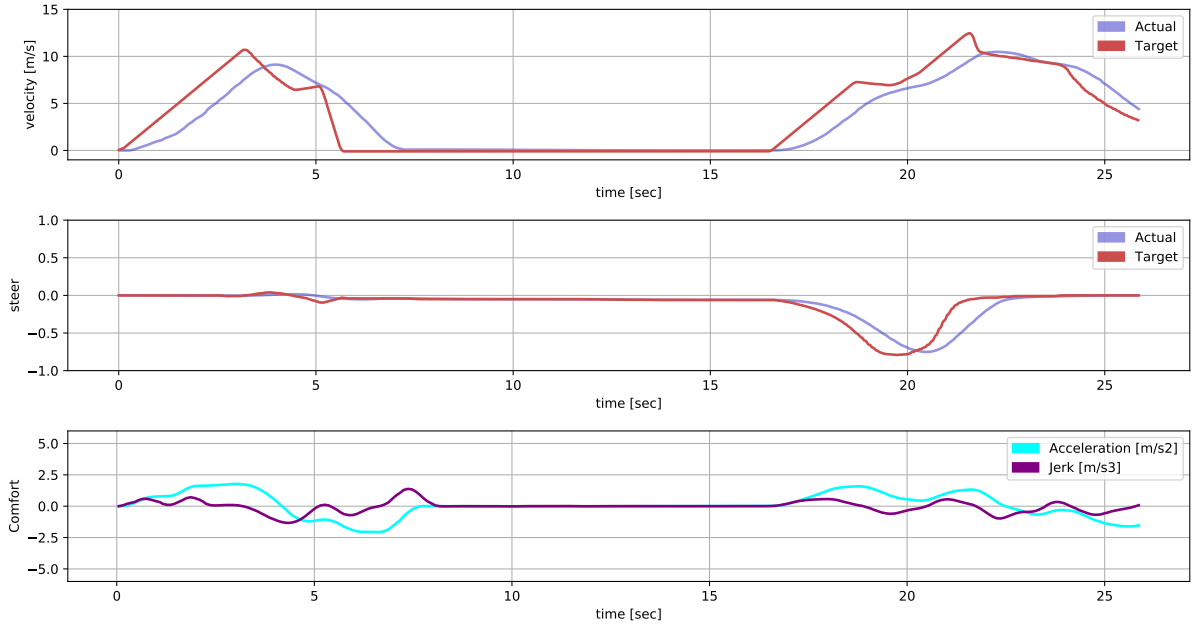


Figure 7.12: Control signals within the merge scenario in CARLA. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.

Figure 7.13 shows the vehicle's manoeuvring strategy during a merge scenario, identifying and utilizing suitable gaps for merging. In this sequence, with the ego vehicle depicted in red and adversarial vehicles in blue, the following steps are observed: 1) The ego vehicle approaches the scenario, activating the DRL agent designated for the merge intersection, resulting in a velocity decrease as shown in the control signals. 2) The proximity of adversaries 1, 2, and 3 makes the vehicle to stop. A stop signal, occurring around the 5th second, ensures the vehicle pauses before the merge. 3) and 4) The ego vehicle remains stationary, faced with a densely populated intersection. 5) A viable gap between adversaries 3 and 4 is identified. Around the 16th second, a drive signal is sent, resulting in the ego vehicle's movement. 6) The ego vehicle initiates movement while simultaneously executing a turn. 7) Having successfully merged into the intersection, the vehicle follows the leading vehicle at a velocity marginally below the nominal rate. 8) The ego vehicle successfully concludes the scenario, reaching the end of the merge scenario.

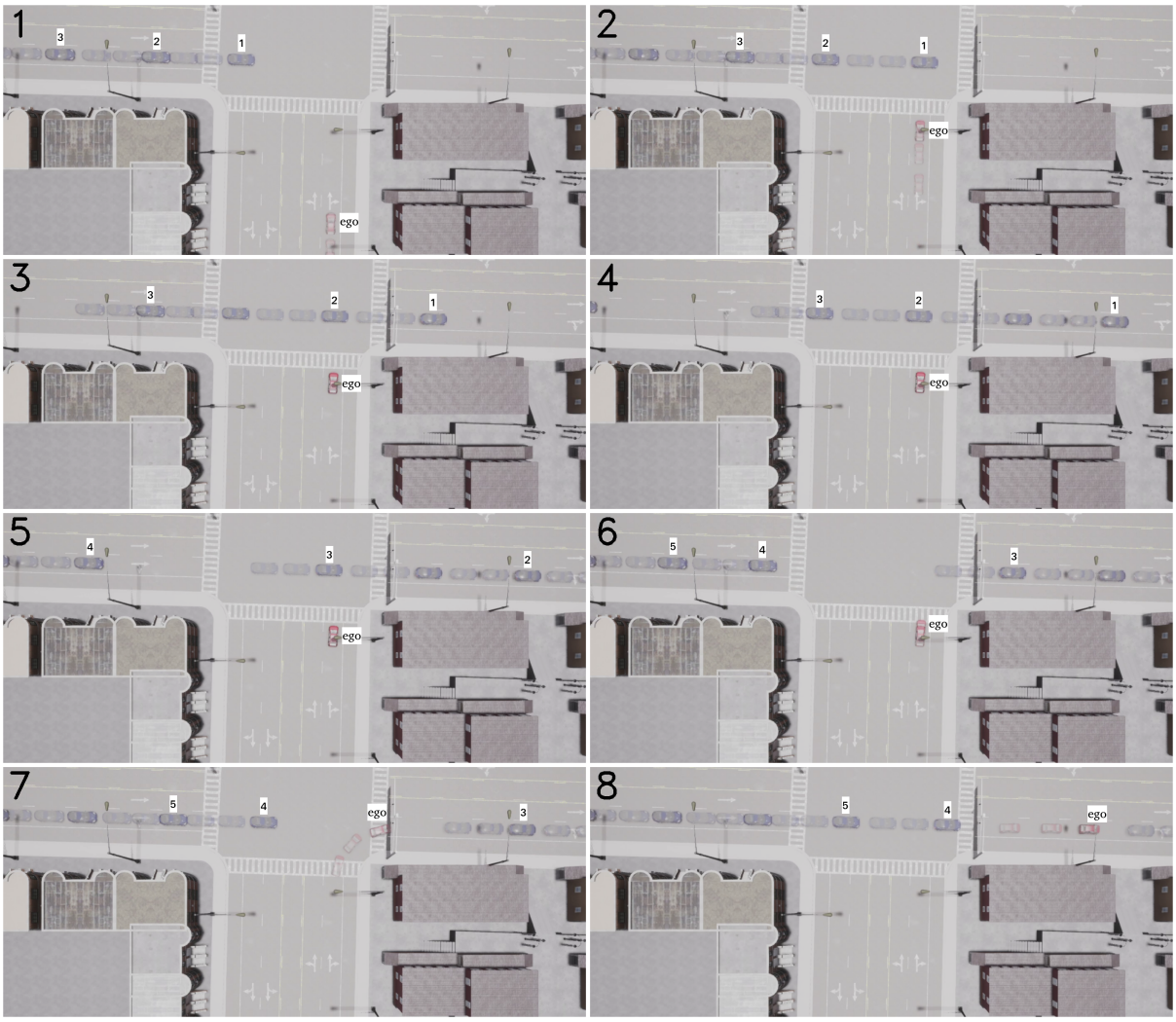


Figure 7.13: A bird's-eye view of a simulated episode in CARLA is presented, describing various scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded.

Crossroad Qualitative Results

Figure 7.14 showcases the control signals for navigating a crossroad scenario. When adversarial vehicles are detected on the road, a stop signal is generated, pausing the vehicle's progress until a clear gap is identified for safe continuation. Due to the vehicle's straight trajectory through the scenario, the steer command remains constant, showcasing no variance. However, this scenario presents slightly higher fluctuations in comfort signals compared to previous scenarios, due to the need to stop completely under the presence of adversarial vehicles.

The dynamic manoeuvres are further illustrated in Figure 7.15, with the sequence unfolding as follows: 1) As the ego vehicle approaches the intersection, it reduces speed, anticipating potential stops due to traffic. 2) The presence of adversaries (vehicles 1, 2, 3, and 4) results in a complete halt to avoid potential collisions. 3) Once adversaries 2 and 4 clear the intersection, creating a suitable gap, the ego vehicle resumes its path by

executing a drive command. 4) Successfully navigating through the crossroad, the ego vehicle reaches the scenario's end without incident.

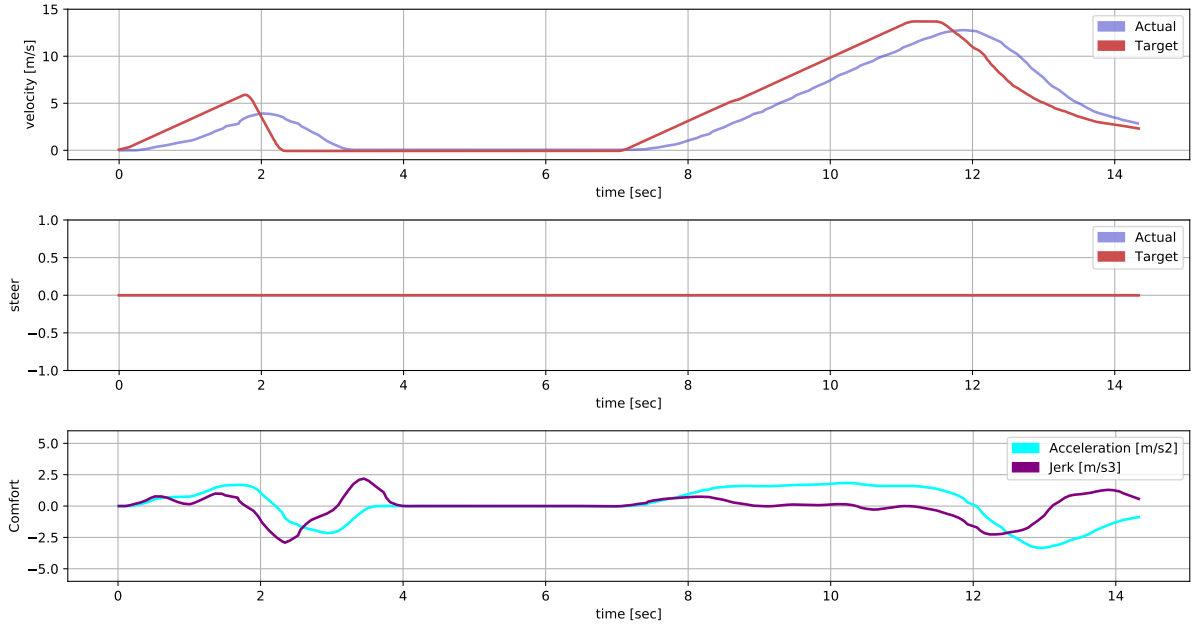


Figure 7.14: Control signals within the crossroad scenario in CARLA. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.

Figure 7.15: A bird's-eye view of a simulated episode in CARLA is presented, describing various scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded.

7.4.3. Concatenated use cases scenario

The evaluation of concatenated scenarios serves as a comprehensive test of our entire AD stack, reflecting its performance in realistic urban driving conditions. 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 7.2. 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 tackles 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 that the Autopilot. Specifically, our system recorded jerk metrics of 4.37, compared to the Autopilot's 8.63 m/s^3. 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 7.2: 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	95.76
	Autopilot	**100**
95th Percentile of Jerk (per episode, in m/s^3) ↓	Ours	**4.37**
	Autopilot	8.63
Maximum Jerk (per episode, in m/s^3) ↓	Ours	**6.20**
	Autopilot	11.94
95th Percentile of Acceleration (per episode, in m/s^2) ↓	Ours	3.33
	Autopilot	**2.98**
Time (sec) ↓	Ours	**76.85**
	Autopilot	140.23
Speed (in m/s) ↑	Ours	**7.60**
	Autopilot	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 7.16 showcasing our AD stack, and Figure 7.17 depicting the CARLA Autopilot's performance.

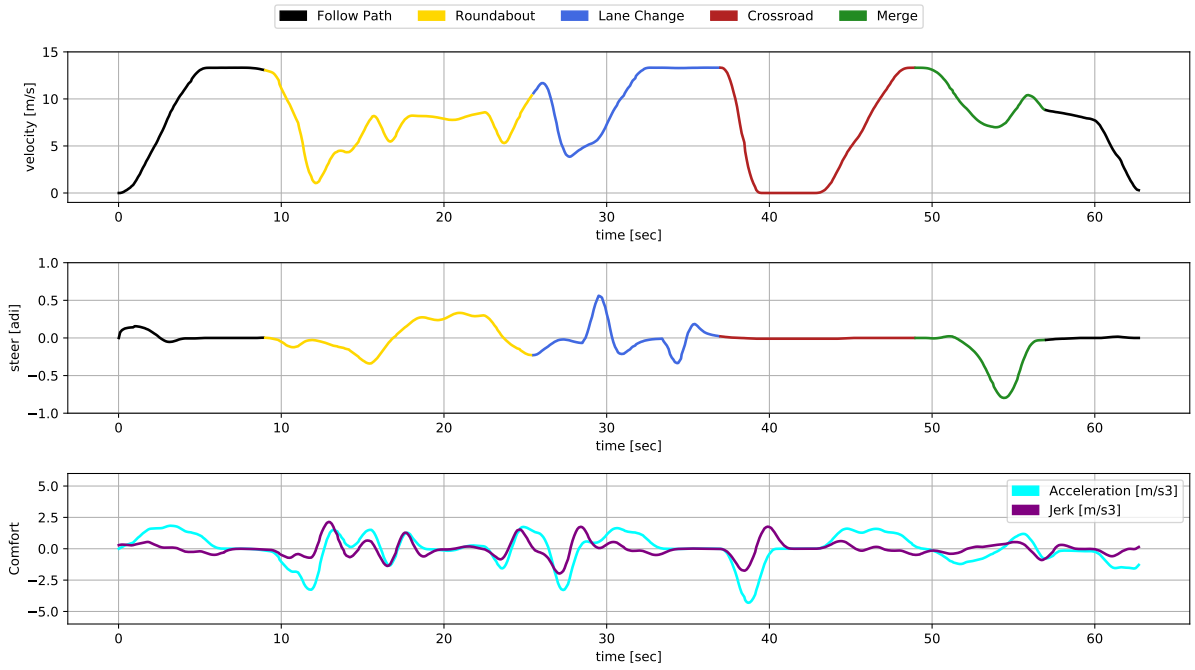


Figure 7.16: Our AD stack temporal response within the concatenated scenario in CARLA. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.

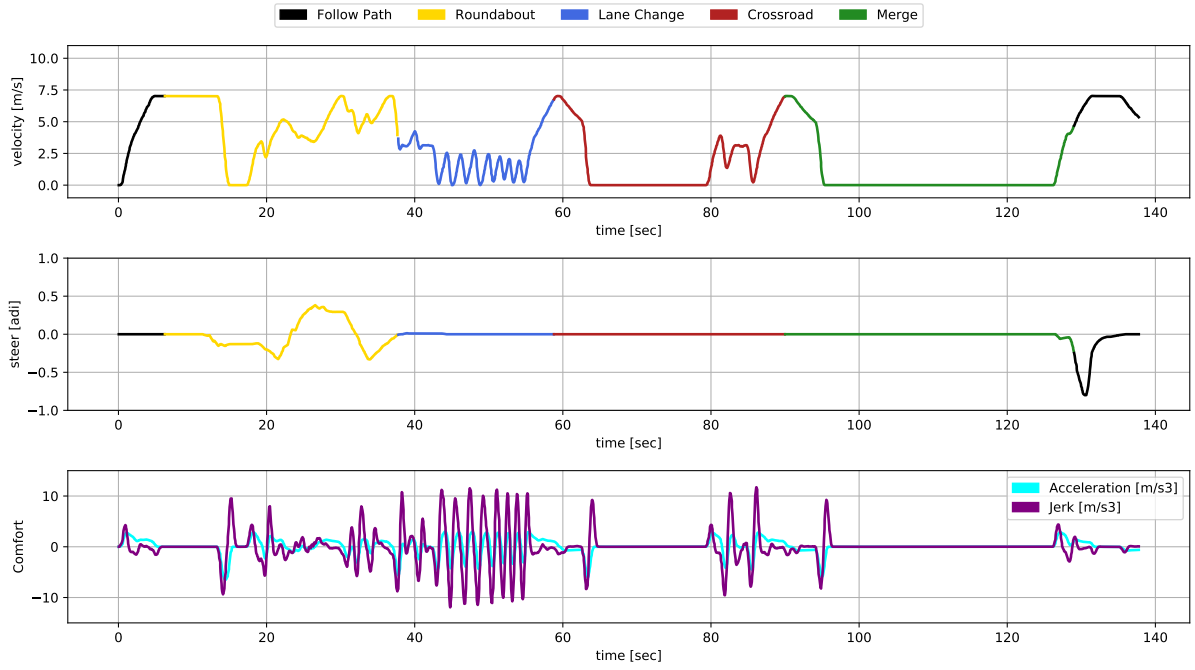


Figure 7.17: CARLA Autopilot temporal response within the concatenated scenario in CARLA. The linear velocity is depicted in the top chart, steering data is presented in the middle chart, and comfort metrics, specifically acceleration and jerk, are illustrated in the bottom chart.

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.

7.5. Shortening the Gap from Simulation to Real-World Implementations

The RL training process can be costly and unsafe when applied directly to real vehicles. Therefore, safety considerations must extend not only to the algorithmic level but also encompass the expenses associated with sensors and potential vehicle damage during training. To address this challenge, many approaches have shifted their focus towards an initial experimental phase that relies on a high-fidelity simulation. Such simulations can encompass critical scenarios, allowing for the detection of risky actions in the training of the DM system before transitioning to real-world testing. In the context of AD, a notable disparity often exists between simulation and reality, a phenomenon commonly referred to as the RG. As introduced in Section 4.3.3, various solutions have been proposed to bridge this RG, primarily categorized into three groups: 1) Sim2real, involving knowledge transfer from simulation to the real world. The core idea is to train AD systems in simulation and then fine-tuning them in real-world vehicles. 2) Digital Twins, utilizing virtual representations of the real environment allow real vehicles to learn knowledge of their DT by synchronizing data from both the real and simulated worlds in an offline way. 3) Parallel Intelligence 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.

In this thesis, we introduce a novel methodology for the practical implementation of a DM module for our AD stack, specifically focusing on merge intersection scenarios and following a CL strategy. Our approach leverages DRL and is structured in three key stages. Firstly, the DRL agent is trained in a lightweight simulator, such as SUMO, to establish a prior behaviour model. Secondly, this model is transferred to a hyperrealistic simulation environment in CARLA for a second training stage using a DT of our real environment and

vehicle. This process includes capturing the dynamics of the vehicle and its sensors, and faithfully replicating the road layout of our university Campus, culminating in a virtual representation of both the vehicle and the Campus. This DT serves as a virtual testing setup for designed scenarios, allowing for the evaluation of our AD stack in simulation in a safe way. The third stage is the validation of our AD stack in a real scenario with virtual perception through a PE, where simulated and real-world experiments are conducted synchronously in real-time. Interaction with adversarial vehicles is simulated, 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. The comprehensive methodology is depicted in Figure 7.18.

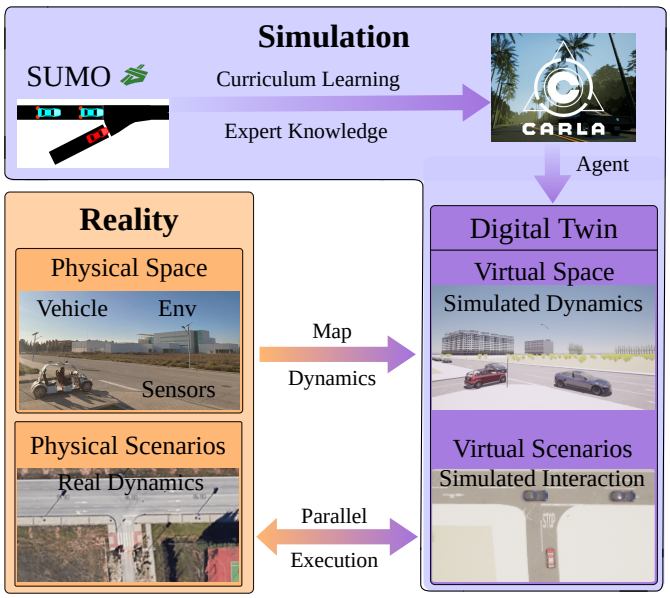


Figure 7.18: Methodology for DM design. 1) DRL agent training in SUMO. 2) Second training of the model in CARLA using a DT of our real environment and vehicle. 3) Validation of the AD stack in a real scenario with virtual perception through a parallel execution.

7.5.1. Digital Twins

In this phase, we integrate the DM module explained in Chapter 6, with the remaining modules of our AD stack, involving two critical steps. Initially, we get the characteristics of our real vehicle and we take measures of the scenario. With this information, we replicate a realistic configuration in CARLA obtaining our DT. This is followed by a secondary training phase using the DT, allowing the agent to adapt it to the new dynamics and control signals taken the SUMO model as a prior. Training from scratch in this hyperrealistic simulator could be time-intensive and has risk of non-convergence, as discussed in our previous work [140].

We define a merge scenario as DT of our Campus. To create this DT, the process begins with acquiring the campus map from OpenStreetMap [141]. This map is then refined using precise GNNs measurements of the area. The refined map is imported into

the RoadRunner tool [142] 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, which does not pose an issue since perception is beyond the scope of this thesis and adversaries' data is directly obtained from ground truth. 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. The objective in this scenario is to navigate to the endpoint without collisions. A depiction of the scenario is presented in Figure 7.19.

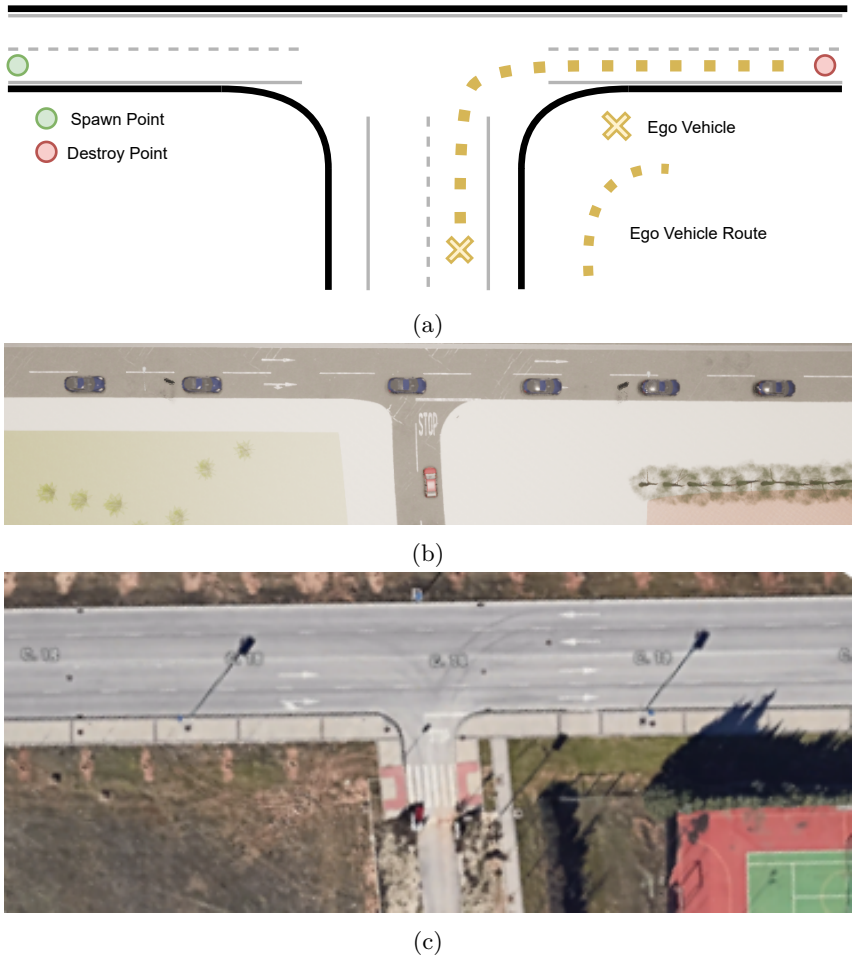


Figure 7.19: Merge intersection within the DT approach. (**a**) Visual representation of the traffic flow. (**b**) Bird-eye-view of the scenario in CARLA. (**c**) Bird-eye-view of the real world intersection within our university Campus.

In CARLA, vehicles are defined using a bicycle dynamic model as explained in Section 4.4.2. To build our ego vehicle DT we calculate the parameters of the model in order the simulated vehicle has the sa,e response as the real one for the same control commands. Table 7.3 outlines essential parameters defining our DT vehicle's model: **Mass (kg):** The vehicle's weight, affecting acceleration and handling. **Max RPM:** Maximum engine speed, indicating power output. **Max Torque ($\mathbf{N \cdot m}$):** Peak engine force, crucial for acceleration. **Moi ($\mathbf{kg \cdot m^2}$):** Vehicle's rotational resistance, influencing directional changes. **Drag Coefficient:** Air resistance measure, affecting aerodynamic efficiency.

Tire Friction: Grip level of tires on the road, essential for control. **Damping Rate:** Suspension's shock absorption capability, for ride smoothness. **Max Steer Angle:** Limits of wheel turning, affecting manoeuvrability. **Delay Response (s):** Time lag in vehicle's response to control inputs.

Table 7.3: The configuration parameters of the Digital Twin vehicle.

Parameter	Value
Mass (kg)	1030
Max RPM	5000
Max Torque ($N \cdot m$)	126
Moi ($kg \cdot m^2$)	0.05
Drag Coefficient	0.60
Tire Friction	0.85
Damping Rate	0.2
Max Steer Angle (deg)	40
Delay Response (s)	0.50

The replication of the vehicle's bodywork is purely aesthetic and falls outside the scope of this thesis.

7.5.2. Parallel Execution

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 7.20.

- **Real Agent:** This agent processes input from a GNSS system to create a localization pose within the Campus map. 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 at a frequency of 20 Hz. This module is responsible for translating these 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. The modules comprising the real-world platform, which include the localization module and the DBW system, are thoroughly described in our prior work [143].
- **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.

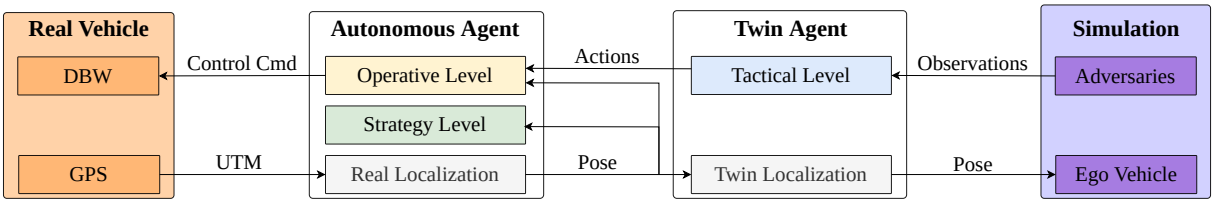


Figure 7.20: The Real Agent processes GPS input and generates control signals, the DBW module controls the real vehicle, while the Twin Agent synchronizes the simulation. Simultaneously, the DM coordinates actions with simulated observations for comprehensive control and coordination within the system.

7.5.3. Results

In this section, we present the outcomes of the experiments conducted for each stage, as detailed in the preceding section. Firstly, some experiments were conducted in the CARLA simulation to evaluate the performance and training efficiency of our DT. Three different use cases were executed: training exclusively in SUMO, fine-tuning the prior SUMO model with CARLA data, and training the model from scratch in CARLA. The outcomes of the experiments are depicted in Table 7.4.

Table 7.4: A comparison in terms of performance and training time of the proposed training approaches for our DT.

Metric	Prior model	Fine-Tuning	From Scratch
Success Rate [%]	75.60	91.80	-
Average Time (sec)	21.53	19.98	-
Episodes Convergence	1M	1M + 10K	1M
Training Time (h)	5	21.5	1650

As anticipated, the prior model in SUMO reduces its performance when tested in CARLA, primarily due to the dynamic environment present in this simulator. It is worth highlighting that the fine-tuning process proved to be highly effective, achieving similar success rates as those observed in previous sections, specifically a 91.80% success rate. Additionally, the average processing times for both the prior and fine-tuned models were similar. On the other hand, the CL strategy, employing both pre-training in SUMO and fine-tuning in CARLA, enables us to achieve model convergence 75 times faster compared to training the model from scratch. In this case, training time was estimated and given its long duration, the performance parameters were not calculated for this case.

For the PE experiments, we do not assess the approach in terms of success rate, as such evaluation would be overly time-consuming for each episode in a real-world implementation. 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 scenarios, each varying in vehicle density and behaviour: 1) Low Traffic Flow, characterized by sparse traffic, enabling the ego vehicle to merge swiftly without significant delays. 2) Mixed Traffic Flow, features a combination of dense and sparse traffic, forcing the ego vehicle to wait for an opportune gap before merging. 3) High Traffic Flow, which presents a congested traffic environment where adversarial vehicles do not yield, making it difficult or impossible for the ego vehicle to merge.

Low Traffic Flow

The control signals are illustrated in Figure 7.21. Initially, the ego vehicle begins its movement with a smoothly increasing target until it reaches the optimal velocity for executing the curve, as it is pre-determined by the velocity profiler of the low-level controller.

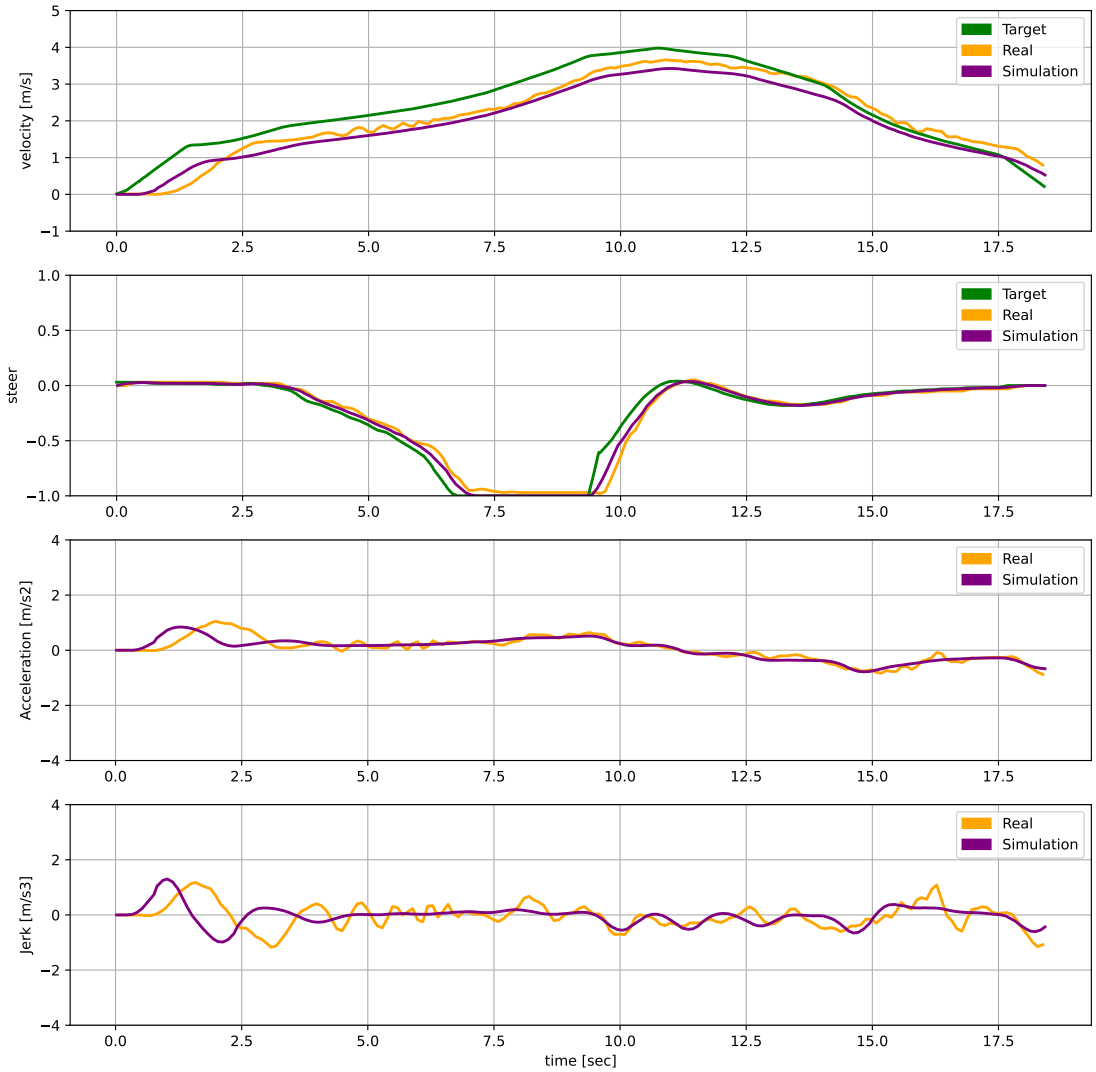


Figure 7.21: Control signals during a PE (virtual and real) within the merge scenario under low traffic flow. Identical control signals are provided to the real and simulated vehicles. Linear velocity is represented in the top graph, steer in the middle graph, acceleration in the third graph and jerk in the bottom graph. The target (green), real (orange), and simulated (purple) signals are illustrated.

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 between the simulated and real vehicle signals, for the same control targets, reveals high similarity, with the real signal appearing slightly sharper. This discrepancy results in marginally higher jerk and acceleration values, although within acceptable levels of comfort. Both signal sets demonstrate a delayed response, due to the dynamics of the real vehicle as well as modelled in the DT. Another difference between the real and simulated models lies in the accuracy of trajectory tracking, as depicted in Figure 7.22. The trajectory is followed more accurately by the DT, with a mean lateral error of **0.09** m and a maximum error of 0.16 m. Conversely, the real vehicle exhibits a larger mean lateral error of 0.23 m and a maximum error of 1.08 m.

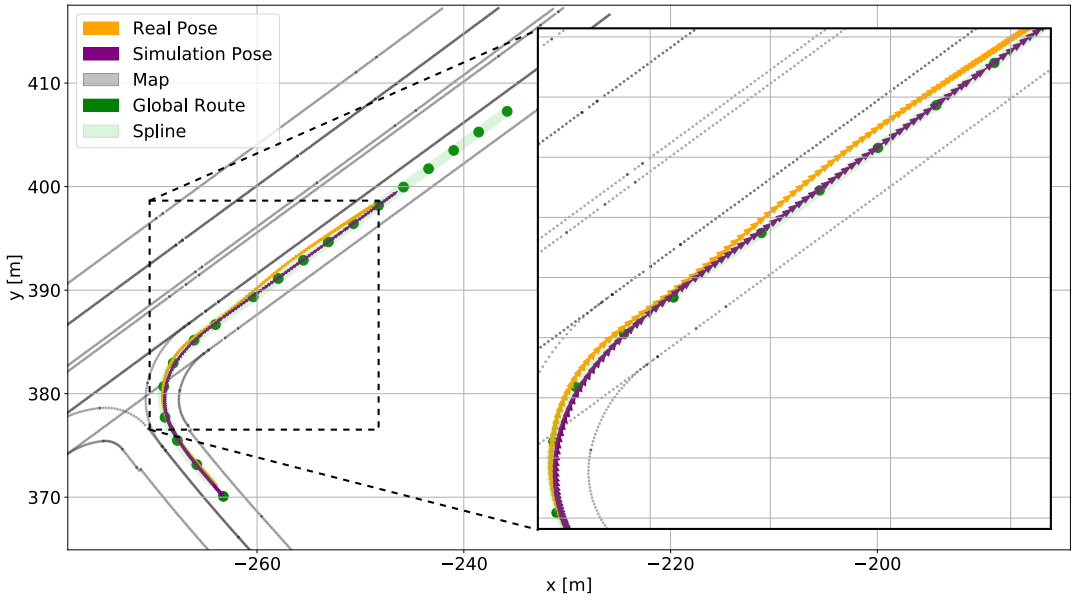


Figure 7.22: The position signals during a Parallel Execution (virtual and real) within the merge scenario under low traffic flow. Identical trajectories are provided to the real and simulated vehicles. The trajectory to be followed (green), real (orange), and simulated (purple) poses are illustrated.

A bird's-eye view representation of our approach is provided to illustrate this scenario, as shown in Figure 7.23.

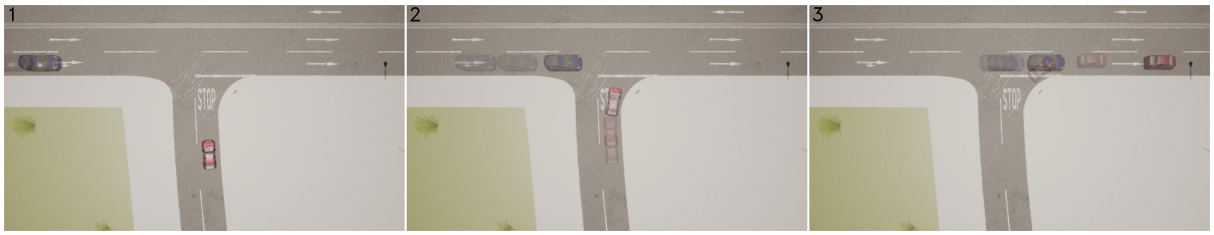


Figure 7.23: Bird's-eye view of a simulated episode under low traffic flow in CARLA describing three scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded.

The sequence is as follows: 1) The ego vehicle starts moving and encounters no approaching vehicles on the main road. 2) The vehicle merges by executing the right turn and adheres to the nominal trajectory, due to the DM action is 'drive'. 3) Subsequently, the vehicle arrives at the route's endpoint and stops, thus successfully completing the scenario.

Mixed Traffic Flow

The control signals are depicted in Figure 7.24 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 the DRL agent to select the stop action, leading to a reduction in velocity until the vehicle stops. Once a gap is identified, the

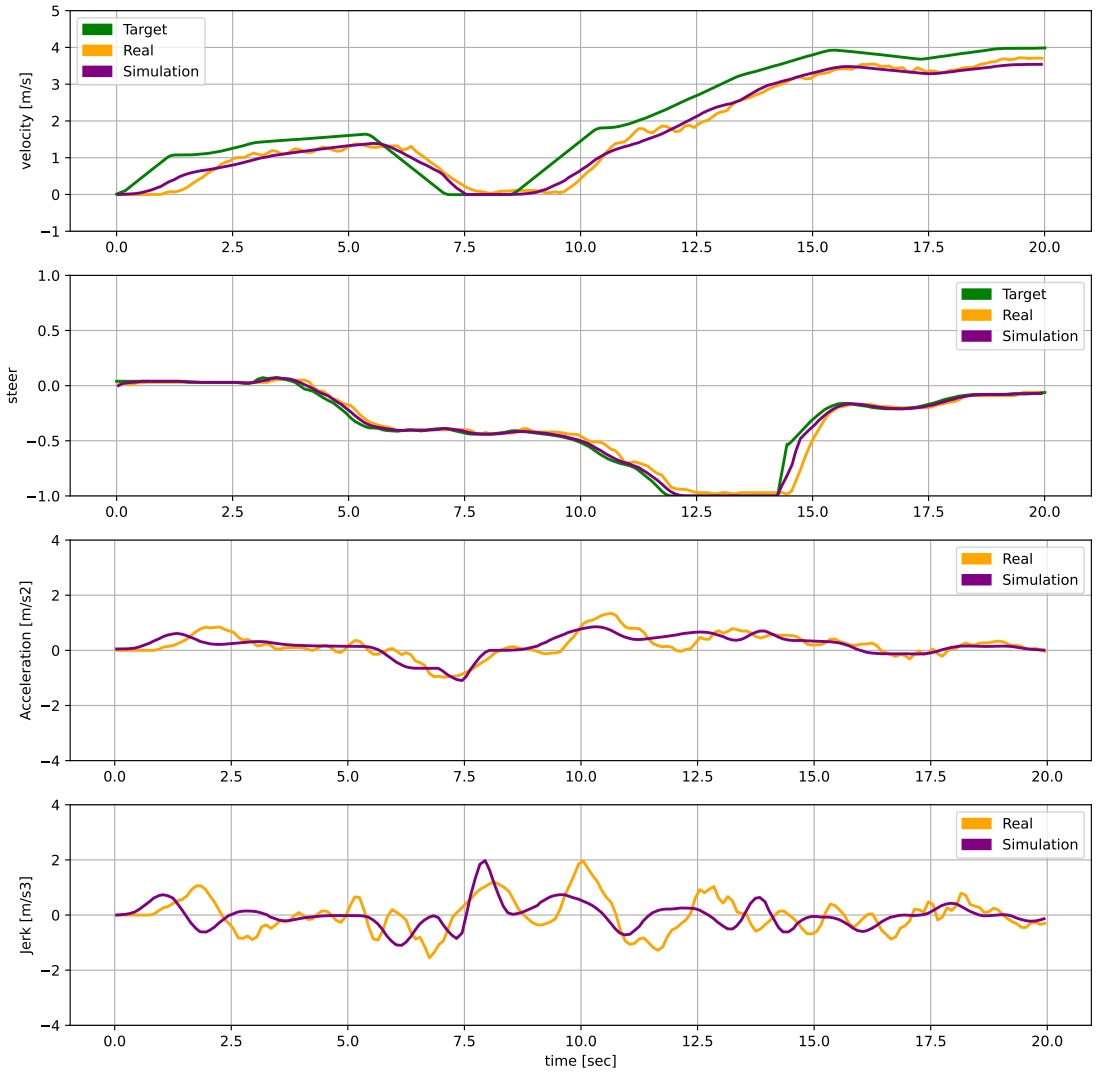


Figure 7.24: Control signals is demonstrated during a Parallel Execution (virtual and real) within the merge scenario under mixed traffic flow. Identical control signals are provided to the real and simulated vehicles. Linear velocity is represented in the top graph, steer in the second graph, acceleration in the third graph and jerk in the bottom graph. The target (green), real (orange), and simulated (purple) signals are illustrated.

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. The trajectory tracking performance is illustrated in Figure 7.25, showcasing a higher precision during this scenario, where the vehicle stops before merging. The tracking during the curve is notably accurate, yet the real-world implementation displays some lateral deviations upon merging the main road. The trajectory tracking mean lateral error for the DT is **0.03** m with a maximum deviation of 0.08 m. In contrast, the real-world execution reveals a mean lateral error of 0.17 m and a maximum deviation of 0.87 m.

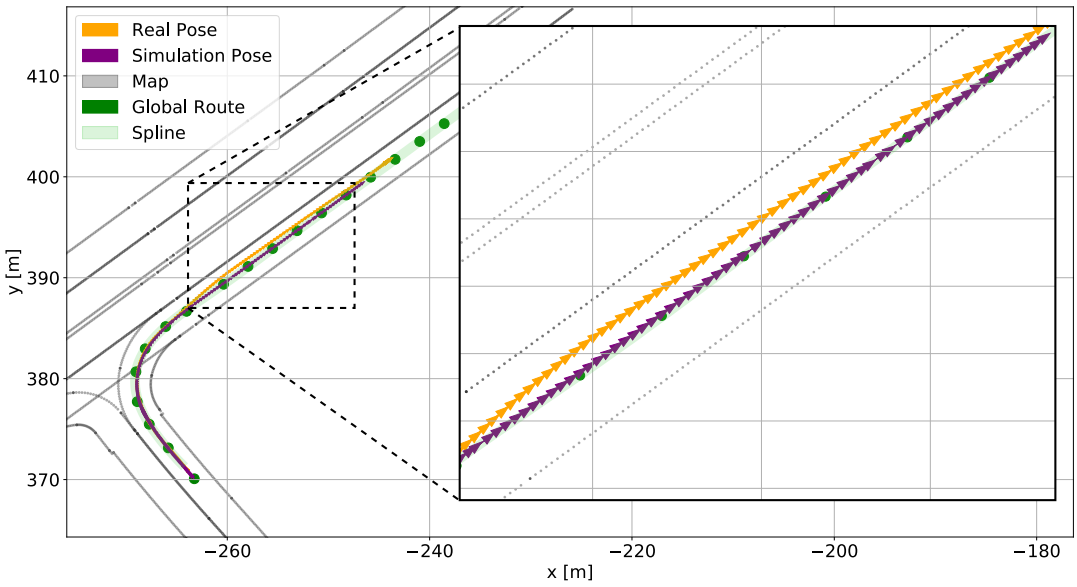


Figure 7.25: Position signals during a Parallel Execution (virtual and real) within the merge scenario under mixed traffic flow. Identical trajectories are provided to the real and simulated vehicles. The trajectory to be followed (green), real (orange), and simulated (purple) poses are illustrated.

A bird's-eye view representation for this use case is illustrated in Figure 7.26. The sequence unfolds as follows: 1) The ego vehicle initiates movement, detecting an approaching adversarial vehicle. 2) It remains stationary until a suitable gap for merging appears. 3) The vehicle then merges onto the main road, completing the scenario.

Figure 7.26: Bird's-eye view of a simulated episode under mixed traffic flow in CARLA describing three scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded.

High Traffic Flow

The control signals for this scenario are illustrated in Figure 7.27, where 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.

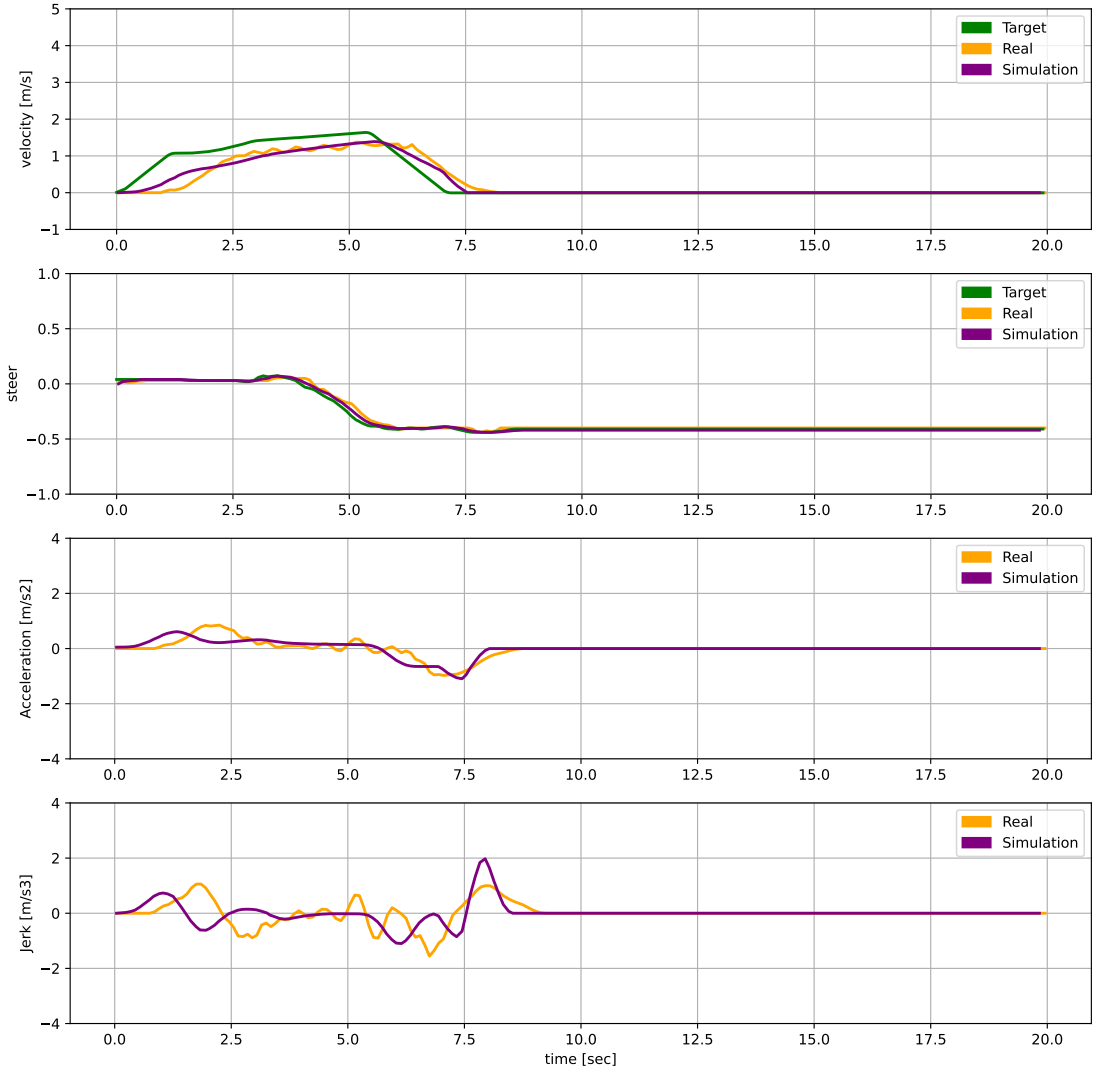


Figure 7.27: Control signals is demonstrated during a Parallel Execution (virtual and real) within the merge scenario under high traffic flow. Identical control signals are provided to the real and simulated vehicles. Linear velocity is represented in the top graph, steer in the second graph, acceleration in the third graph and jerk in the bottom graph. The target (green), real (orange), and simulated (purple) signals are illustrated.

The performance of trajectory tracking is showcased in Figure 7.28. In this specific scenario, as the vehicle stops prior to navigating the curve, both simulated and real-world approaches demonstrate minimal lateral errors. This outcome indicates a uniform

response from both models when dealing with straightforward trajectories, with the maximum deviation for each remaining below 0.10 meters.

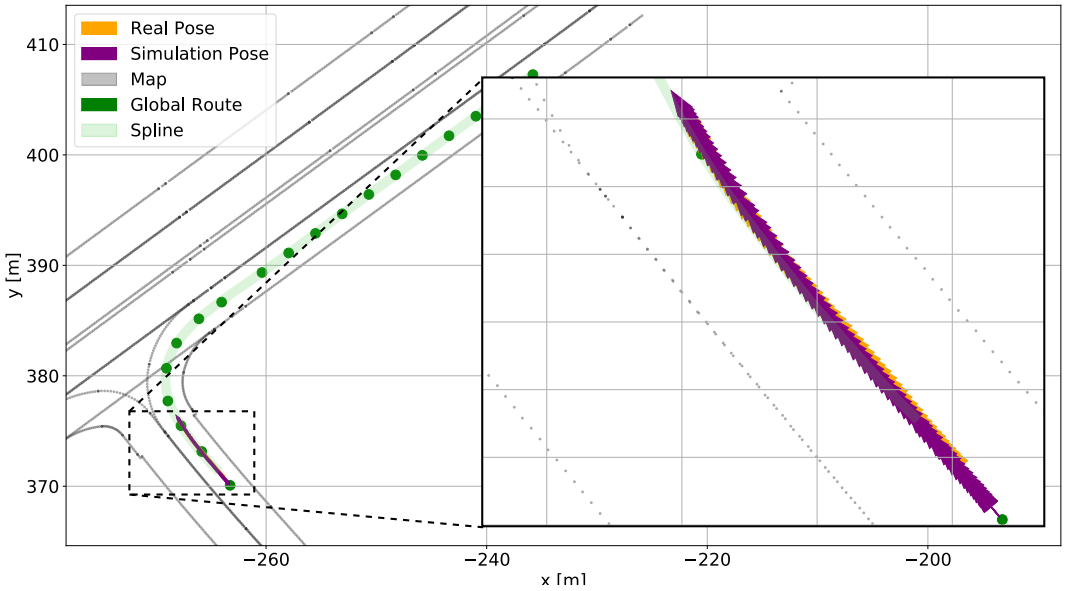


Figure 7.28: Position signals during a Parallel Execution (virtual and real) within the merge scenario under high traffic flow. Identical trajectories are provided to the real and simulated vehicles. The trajectory to be followed (green), real (orange), and simulated (purple) poses are illustrated.

A bird's-eye view representation of our approach is illustrated in Figure 7.29. The sequence unfolds as follows: 1) The ego vehicle initiates movement, detecting an approaching adversarial vehicle. 2) It remains stationary. 3) The episode concludes successfully without any collisions and without the ego vehicle managing to merge, demonstrating a prudent decision of the agent.

Figure 7.29: Bird's-eye view of a simulated episode under high traffic flow in CARLA describing three scenes from the same episode. The most prominent vehicles in each scene represent the latest positions, while the positions from earlier moments are gradually faded.

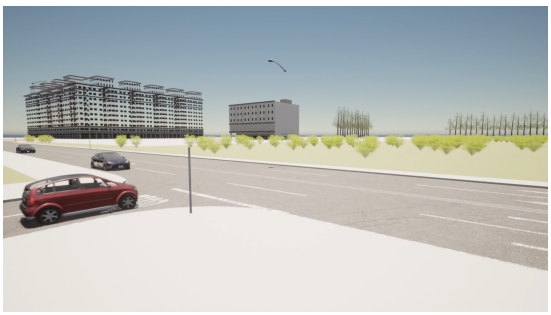
A sequence of images captured at the same time points during experiments for three different traffic scenarios (high, mixed, and low traffic flows) is illustrated in Figures 7.30, 7.31, and 7.32, visually representing the DT and the real vehicle. In these figures, the ego vehicle is shown in the left-side, while the DT is depicted in the right-side with the vehicle in red As we said,, the DT does not extend to the physical body of the vehicle itself. In contrast to the real-world environment, which lacks other traffic participants, the

simulated scenario in the DT presents a notable traffic flow with blue vehicles, highlighting the dynamic and complex nature of the simulated traffic conditions.

7.6. Summary

This chapter has presented an exploration into the practical validation of our AD stack, which includes our hybrid DM proposal, with the focus on RL integration within a standard simulation environment like CARLA, complemented by real-world application through a DT and PE approach. Through the detailed creation of urban scenarios in CARLA, we aimed to simulate realistic driving conditions, focusing on safety, comfort, and efficiency metrics to evaluate our approach. The evaluation of our AD stack highlighted its effectiveness in navigating these scenarios, demonstrating significant achievements in success rates, jerk dynamics, and overall efficiency when compared to the CARLA Autopilot. In transitioning towards real-world implementation, the concept of DT was introduced, providing a safe and cost-effective bridge from simulation to real application. This stage involved replicating the dynamics of our real vehicle and our Campus environment within CARLA, followed by a PE methodology that synchronized real-world driving with virtual simulation. The results from both simulated and real-world applications demonstrated the viability of our approach, offering insights into the potential for RL-based systems in AD.

(a)



(b)

(c)

(d)

(e)

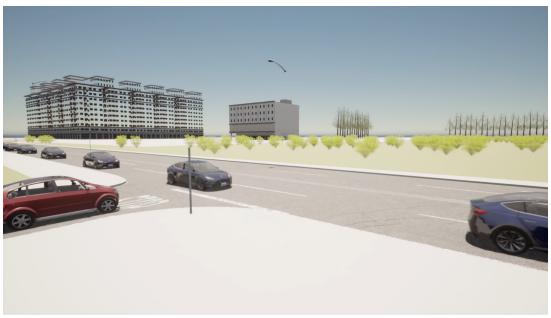
(f)

(g)

(h)

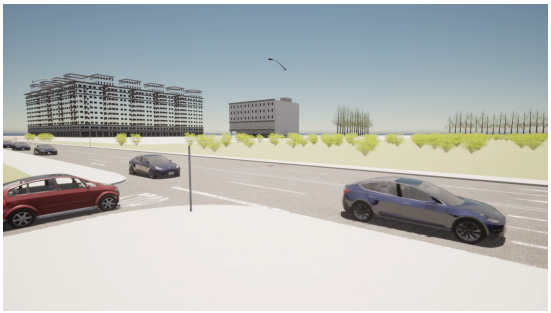
Figure 7.30: Digital Twins. Low traffic flow scenario.**a)** An adversarial vehicle is approaching but is far enough. **b)** The vehicle continues its course without stopping, merging into the main lane. **c)** The vehicle now occupies the main road. **d)** The vehicle arrives at the end of the scenario.

(a)



(b)

(c)



(d)

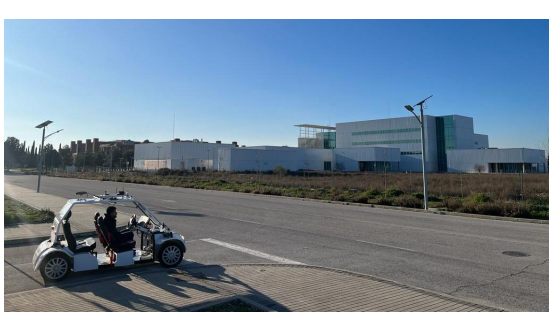
(e)

(f)

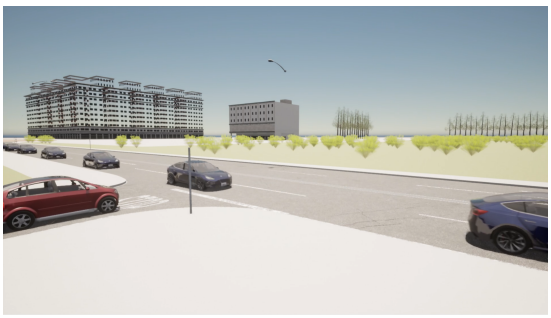
(g)

(h)

Figure 7.31: Digital Twins. Mixed traffic flow scenario. **a)** Adversarial vehicles are approaching, prompting the vehicle to stop. **b)** The vehicle remains stationary, awaiting an opportunity to merge. **c)** A gap is identified, allowing the vehicle to initiate merging into the main lane. **d)** Successfully merged, the vehicle continues along the main road.



(a)

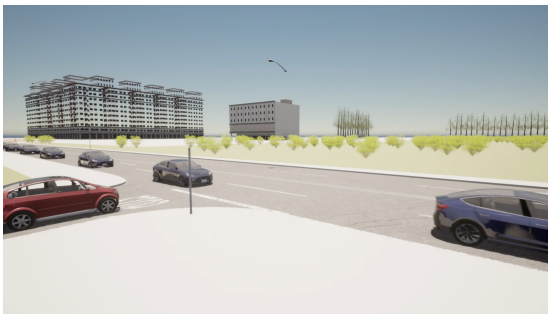


(b)

(c)

(d)

(e)



(f)

(g)

(h)

Figure 7.32: Digital Twins. High traffic flow scenario. **a)** The ego vehicle reaches the intersection. **b)** Despite ongoing observation, no feasible gap emerges, leading to a stop. **c)** The vehicle remains stationary, continuously assessing the traffic flow for opportunities. **d)** With no gaps detected, the vehicle concludes the scenario without merging.

Chapter 8

Conclusions and Future Works

Nada es permanente a excepción del cambio.

Heráclito

8.1. Conclusions

In this doctoral thesis, we have developed an innovative hybrid Decision Making (DM) architecture for Autonomous Driving (AD) systems, integrating classical control techniques with Deep Reinforcement Learning (DRL). This hybrid approach represents a significant advancement towards applying these systems in real-world settings, combining the reliability of traditional control methods with the adaptive capabilities of DRL.

In our examination of AD, we explore a broad spectrum of methods, placing particular emphasis on DM. Through our analysis of both End-to-End and modular AD architectures, we underscore the vital role of DM in ensuring both safe and efficient navigation. Additionally, we have conducted a detailed assessment of the leading DM research trends, evaluating their strengths and weaknesses. Some of these trends focus on developing promising DRL modules, while others focus on their practical deployment. Our review identifies a significant gap in the existing State of the Art (SOTA), suggesting that our proposed architecture has the potential to offer practical solutions.

The proposed hybrid DM architecture is currently integrated within an AD stack; however, its design is sufficiently flexible to be adapted to other architectures. Structured across three distinct levels: strategic, tactical, and operational, our hybrid architecture employs a modular approach. This modularity allows the replacement or interchange of these modules as needed, and supports their development in parallel. Such a framework proves immensely valuable for scientific research, offering versatility and adaptability in application.

The operative level, which is pivotal to our architecture, employs Linear–quadratic regulator (LQR) and Model Predictive Control (MPC) techniques for precise vehicle manoeuvring. It is meticulously designed to ensure the translation of high-level decisions into smooth and safe vehicle movements. For trajectory tracking, the LQR controller is employed in conjunction with a delay compensation module, and for manoeuvres, two MPC controllers (one for linear and another for lateral control) are used, featuring models that do not demand high computational resources. This hybrid control architecture enables us to execute trajectories in real time, effectively addressing the potential issues associated with relying solely on MPC techniques.

The development of the tactical level involves delving into DM algorithms for taking high-level actions, where the TRPO algorithm stands out for its exceptional performance. The DRL models trained in SUMO without vehicle dynamics are integrated in our AD architecture for a more comprehensive evaluation with dynamics. Testing these models in the CARLA simulator is the first step towards a real implementation. Throughout the training and validation process, four scenarios are utilized, culminating in the conclusion that the system can effectively navigate these scenarios when they are concatenated. Moreover, the proposed methodology enables the expansion of these scenarios to encompass a broader range, including additional lanes, various morphologies, or diverse situations. Consequently, our proposal is characterized as scalable in terms of both complexity and the number of scenarios.

The real-world implementation of any system developed in simulation is invariably challenging. In this thesis, we aim to develop an architecture that can be applied in a real-world environment, and to this end, we have adopted a Curriculum Learning (CL) methodology. This involves a three-phase development process: initial training in a lightweight simulator (SUMO), employing a Digital Twins (DT) method in a highly realistic simulator (CARLA) to fine-tuning the model, and finally testing the complete AD stack in real-world scenarios with virtual observations through Parallel Execution (PE). 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 PE and DT we can implement our architecture in a real environment proving the practical viability of our proposal.

In conclusion, this dissertation presents a comprehensive development and application of a hybrid DM architecture, including its integration into an AD stack, showcasing its theoretical foundation, practical implementation, adaptability and robustness. Our work significantly contributes to the field of AD, paving the way for future research and development.

8.2. Future Works

Despite the advancements presented in this dissertation, the field of AD and DM remains ripe for further exploration. In this final section, we identify three potential lines of research for future work. The initial line emphasizes DRL techniques, the second one is focused on simulation strategies, and the final line explores the application through real-world experiments.

The focus of this thesis proposal lies in a classical implementation of DRL, as the aim is to integrate it into a complete AD architecture. However, as previously mentioned, there are several lines of research within Reinforcement Learning (RL). Delving deeper into these lines in the future, particularly in the following ones, would be intriguing:

- **Transfer Learning** has emerged to address the diverse challenges encountered by RL [144], by leveraging knowledge from external sources to enhance the efficiency and effectiveness of the learning process. Specifically, representation transfer can be valuable for developing generalizable models that are trained in one scenario and then transferred to another. Moreover, knowledge transfer from pre-trained models can be employed to achieve scalability in these models.

- In **Multi Agent Reinforcement Learning**, multiple agents simultaneously learn within a shared environment, enhancing model performance through the dynamics of competition and cooperation [145]. Integrating CL with this approach allows for a structured progression in learning. By systematically sampling the behaviour of entities from previous stages of the curriculum at each step, agents can learn from each other's experiences and strategies.

- Although end-to-end approaches are challenging to apply in real-world experiments, recent advancements are moving in this direction, particularly **Transformer-based Reinforcement Learning** has helped to alleviate the high computational demands of these end-to-end systems [146]. This development has been shown to outperform previous methods, such as the use of LSTM [147], in terms of efficiency and effectiveness. Furthermore, these techniques can be used to enhance scene representation, thereby achieving better generalization across changes in the environment and scenarios.

- **Inverse Reinforcement Learning (IRL)** addresses the challenge of inferring the hidden preferences of an agent based on its observed behaviour [148]. This methodology allows for demonstrations to replace the manual specification of rewards. Such demonstrations can be provided by a human expert or a previously developed system. A compelling direction for research is to create an end-to-end RL architecture that learns from our hybrid DM architecture until it achieves equivalent performance.

This model could subsequently act as a baseline for training in more intricate environments, leveraging the initial insights and strategies acquired from the simpler scenarios.

Regarding simulation, countless possibilities exist, and thanks to today's computational capacity, even more opportunities arise. In this line, we identify several interesting pieces of research, specifically:

- Scaling our proposal by incorporating more diverse scenarios with different shapes, traffic patterns, vehicles, etc. This includes tackling more complex cases while maintaining the proposed architecture.
- Creating a framework for the **Automatic Generation of Scenarios**, from critical to non-critical, minimizing the execution of redundant or non-essential scenarios to preserve computing resources. By integrating adaptive elements, search algorithms, and RL methodologies, this framework can generate critical scenarios while maintaining naturalness and relevance [149]. Such a system would equip training agents with a diverse set of scenarios, significantly improving their generalization abilities and preparing them for a broad spectrum of real-world conditions.
- **Sensor Simulation:** As a preliminary step before real-world implementation, utilizing realistic simulated sensors allows for testing the system against perception module failures or inaccuracies. This includes studying the system's performance in the presence of noise.

Finally, concerning real-world implementation, it would be beneficial to explore the following lines of research:

- Implementing an **Auto-tuning** system for controllers [150], as parameter adjustment in real vehicles can be time-consuming. This would involve developing algorithms that can automatically adjust controller parameters based on real-time feedback, reducing the need for manual tuning.
- Testing with real vehicles as adversaries, incorporating **vehicle-to-vehicle** [151] 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.

In conclusion, this dissertation represents a significant step forward in the quest for reliable and efficient autonomous vehicle navigation. The proposed hybrid DM framework, validated through simulation and preliminary real-world testing, lays the groundwork for future advancements in the field. By continuing to explore the synergies between DRL, classical control techniques, and advanced simulation methodologies, researchers can further enhance the safety, efficiency, and reliability of AVs, ultimately realizing the full potential of this transformative technology.

List of Publications

Journal Papers

- R. Gutiérrez, E. López-Guillén, L. M. Bergasa, R. Barea, Ó. Pérez, C. Gómez-Huélamo, F. Arango, J. del Egido, and J. López-Fernández, "A Waypoint Tracking Controller for Autonomous Road Vehicles Using ROS Framework", *Sensors*, vol. 20, no. 14, 2020, ISSN: 1424-8220. DOI: 10.3390/s20144062. [Online]. Available: https://www.mdpi.com/1424-8220/20/14/4062

- 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, ISSN: 1424-8220. DOI: 10.3390/s22218373. [Online]. Available: https://www.mdpi.com/1424-8220/22/21/8373

Conference Contributions

- R. Gutiérrez, J. F. Arango, C. Gomez-Huélamo, L. M. Bergasa, R. Barea, and J. Araluce, "Validation Method of a Self-Driving Architecture for Unexpected Pedestrian Scenario in CARLA Simulator", in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 1144–1149. DOI: 10.1109/IV48863.2021.9575884

- R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, N. Abdeselam, and L. M. Bergasa, "Hybrid Decision Making for Autonomous Driving in Complex Urban Scenarios", in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–7. DOI: 10.1109/IV55152.2023.10186666

- R. Gutiérrez-Moreno, C. Gómez-Huelamo, R. Barea, E. López-Guillén, F. Arango, and L. M. Bergasa, "Augmented Reinforcement Learning with Efficient Social-Based Motion Prediction for Autonomous Decision-Making", in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 2023, pp. 4167–4172. DOI: 10.1109/ITSC57777.2023.10422277

- R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, 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) IN SUBMISSION*, 2024

Bibliography

[1] J. Rosenzweig and M. Bartl, "A review and analysis of literature on autonomous driving", *E-Journal Making-of Innovation*, pp. 1–57, 2015.

[2] J. Deichmann, *Autonomous Driving's Future: Convenient and Connected*. McKinsey, 2023.

[3] M. Wansley, "The End of Accidents", *UC Davis L. Rev.*, vol. 55, p. 269, 2021.

[4] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles", in *2017 IEEE intelligent vehicles symposium (IV)*, IEEE, 2017, pp. 1671–1678.

[5] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey", *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[6] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based Reinforcement Learning: A Survey", *CoRR*, vol. abs/2006.16712, 2020. arXiv: 2006.16712. [Online]. Available: https://arxiv.org/abs/2006.16712.

[7] J. Dinneweth, A. Boubezoul, R. Mandiau, and S. Espié, "Multi-agent reinforcement learning for autonomous vehicles: A survey", *Autonomous Intelligent Systems*, vol. 2, no. 1, p. 27, 2022.

[8] Q. Sun, L. Zhang, H. Yu, W. Zhang, Y. Mei, and H. Xiong, "Hierarchical reinforcement learning for dynamic autonomous vehicle navigation at intelligent intersections", in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 4852–4861.

[9] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, N. Abdeselam, and L. M. Bergasa, "Hybrid Decision Making for Autonomous Driving in Complex Urban Scenarios", in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023.

[10] R. Gutiérrez, E. López-Guillén, L. M. Bergasa, R. Barea, Ó. Pérez, C. Gómez-Huélamo, F. Arango, J. del Egido, and J. López-Fernández, "A Waypoint Tracking Controller for Autonomous Road Vehicles Using ROS Framework", *Sensors*, vol. 20, no. 14, 2020, ISSN: 1424-8220. DOI: 10.3390/s20144062. [Online]. Available: https://www.mdpi.com/1424-8220/20/14/4062.

[11] 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*, 2011, pp. 63–68.

[12] 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*, Nov. 2020. [Online]. Available: https://arxiv.org/abs/2010.09776.

[13] R. Gutiérrez-Moreno, C. Gómez-Huelamo, R. Barea, E. López-Guillén, F. Arango, and L. M. Bergasa, "Augmented Reinforcement Learning with Efficient Social-Based Motion Prediction for Autonomous Decision-Making", in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, 2023, pp. 4167–4172. DOI: 10.1109/ITSC57777.2023.10422277.

[14] 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., ser. Proceedings of Machine Learning Research, vol. 78, PMLR, 13–15 Nov 2017, pp. 1–16. [Online]. Available: http://proceedings.mlr.press/v78/dosovitskiy17a.html.

[15] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *OpenAI Gym*, 2016. eprint: arXiv:1606.01540.

[16] 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, ISSN: 1424-8220. DOI: 10.3390/s22218373. [Online]. Available: https://www.mdpi.com/1424-8220/22/21/8373.

[17] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, F. Arango, 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) IN SUBMISSION*, 2024.

[18] M. O'Kelly, A. Sinha, H. Namkoong, J. C. Duchi, and R. Tedrake, "Scalable End-to-End Autonomous Vehicle Testing via Rare-event Simulation", *CoRR*, vol. abs/1811.00145, 2018. arXiv: 1811.00145. [Online]. Available: http://arxiv.org/abs/1811.00145.

[19] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: a survey", *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020. DOI: 10.1109/JAS.2020.1003021.

[20] T. Kessler, J. Bernhard, M. Buechel, K. Esterle, P. Hart, D. Malovetz, M. Truong Le, F. Diehl, T. Brunner, and A. Knoll, "Bridging the Gap between Open Source Software and Vehicle Hardware for Autonomous Driving", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1612–1619. DOI: 10.1109/IVS.2019.8813784.

[21] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies", *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020. DOI: 10.1109/ACCESS.2020.2983149.

[22] C. Gómez-Huélamo, A. Diaz-Diaz, J. Araluce, M. E. Ortiz, R. Gutiérrez, F. Arango, Á. Llamazares, and L. M. Bergasa, "How to build and validate a safe and reliable Autonomous Driving stack? A ROS based software modular architecture baseline", in *2022 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2022, pp. 1282–1289.

[23] Q. Liu, X. Li, S. Yuan, and Z. Li, *Decision-Making Technology for Autonomous Vehicles Learning-Based Methods, Applications and Future Outlook*, 2021. arXiv: 2107.01110 [cs.RO].

[24] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers", *Autonomous Agents and Multi-Agent Systems*, vol. 27, pp. 1–51, 2013.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning", 2013, cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. [Online]. Available: http://arxiv.org/abs/1312.5602.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning", *CoRR*, vol. abs/1602.01783, 2016. arXiv: 1602.01783. [Online]. Available: http://arxiv.org/abs/1602.01783.

[28] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization", in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, Jul. 2015, pp. 1889–1897. [Online]. Available: https://proceedings.mlr.press/v37/schulman15.html.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms", *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347.

[30] P. Wang, S. Gao, L. Li, S. Cheng, and H.-x. Zhao, "Research on driving behavior decision making system of autonomous driving vehicle based on benefit evaluation model", *Archives of Transport*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221245832.

[31] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams, "A Perception-Driven Autonomous Urban Vehicle", in *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, M. Buehler, K. Iagnemma, and S. Singh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 163–230, ISBN: 978-3-642-03991-1. DOI: 10.1007/978-3-642-03991-1_5. [Online]. Available: https://doi.org/10.1007/978-3-642-03991-1_5.

[32] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford Entry in the Urban Challenge", in *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, M. Buehler, K. Iagnemma, and S. Singh, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 91–123, ISBN: 978-3-642-03991-1. DOI: 10.1007/978-3-642-03991-1_3. [Online]. Available: https://doi.org/10.1007/978-3-642-03991-1_3.

[33] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knöppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making Bertha Drive—An Autonomous Journey on a Historic Route", *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014. DOI: 10.1109/MITS.2014.2306552.

[34] A. Artuñedo, J. Godoy, and J. Villagra, "A decision-making architecture for automated driving without detailed prior maps", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1645–1652. DOI: 10.1109/IVS.2019.8814070.

[35] C.-W. Chang, C. Lv, H. Wang, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, "Multi-point turn decision making framework for human-like automated driving", in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317831.

[36] P. F. Orzechowski, C. Burger, and M. Lauer, "Decision-making for automated vehicles using a hierarchical behavior-based arbitration scheme", *CoRR*, vol. abs/2003.01149, 2020. arXiv: 2003.01149. [Online]. Available: https://arxiv.org/abs/2003.01149.

[37] J. López, P. Sánchez-Vilariño, R. Sanz, and E. Paz, "Implementing autonomous driving behaviors using a message driven petri net framework", *Sensors*, vol. 20, no. 2, p. 449, 2020.

[38] N. Wu, F. Chu, S. Mammar, and M. Zhou, "Petri net modeling of the cooperation behavior of a driver and a copilot in an advanced driving assistance system", *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 977–989, 2011.

[39] K. M. Ng, M. B. I. Reaz, and M. A. M. Ali, "A review on the applications of Petri nets in modeling, analysis, and control of urban traffic", *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 858–870, 2013.

[40] H. Bey, F. Dierkes, S. Bayerl, A. Lange, D. Faßender, and J. Thielecke, "Optimization-based Tactical Behavior Planning for Autonomous Freeway Driving in Favor of the Traffic Flow", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1033–1040. DOI: 10.1109/IVS.2019.8813787.

[41] D. Yang, K. Redmill, and U. Ozguner, *A Multi-State Social Force Based Framework for Vehicle-Pedestrian Interaction in Uncontrolled Pedestrian Crossing Scenarios*, 2020. arXiv: 2005.07769 [cs.RO].

[42] L. Sun, W. Zhan, C.-Y. Chan, and M. Tomizuka, *Behavior Planning of Autonomous Cars with Social Perception*, 2019. arXiv: 1905.00988 [cs.RO].

[43] N. Li, D. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. Girard, *Game-Theoretic Modeling of Driver and Vehicle Interactions for Verification and Validation of Autonomous Vehicle Control Systems*, 2016. arXiv: 1608.08589 [cs.AI].

[44] Z. Kokkinogenis, M. Teixeira, P. M. d'Orey, and R. J. F. Rossetti, "Tactical Level Decision-Making for Platoons of Autonomous Vehicles Using Auction Mechanisms", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1632–1638. DOI: 10.1109/IVS.2019.8814122.

[45] D. Isele, *Interactive Decision Making for Autonomous Vehicles in Dense Traffic*, 2019. arXiv: 1909.12914 [cs.AI].

[46] J. Chen, P. Zhao, H. Liang, and T. Mei, "A Multiple Attribute-based Decision Making model for autonomous vehicle in urban environment", in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014, pp. 480–485. DOI: 10.1109/IVS.2014.6856470.

[47] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1584–1589. DOI: 10.1109/IVS.2017.7995935.

[48] D. Iberraken, L. Adouane, and D. Denis, "Safe Autonomous Overtaking Maneuver based on Inter-Vehicular Distance Prediction and Multi-Level Bayesian Decision-Making", in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3259–3265. DOI: 10.1109/ITSC.2018.8569401.

[49] ——, "Reliable Risk Management for Autonomous Vehicles based on Sequential Bayesian Decision Networks and Dynamic Inter-Vehicular Assessment", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2344–2351. DOI: 10.1109/IVS.2019.8813800.

[50] 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)*, 2017, pp. 1590–1595. DOI: 10.1109/IVS.2017.7995936.

[51] Y. Liu, X. Wang, L. Li, S. Cheng, and Z. Chen, "A Novel Lane Change Decision-Making Model of Autonomous Vehicle Based on Support Vector Machine", *IEEE Access*, vol. 7, pp. 26 543–26 550, 2019. DOI: 10.1109/ACCESS.2019.2900416.

[52] R. Tami, B. Soualmi, A. Doufene, J. Ibanez, and J. Dauwels, "Machine learning method to ensure robust decision-making of AVs", in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1217–1222. DOI: 10.1109/ITSC.2019.8917085.

[53] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving", in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2722–2730.

[54] M. Bojarski *et al.*, "End to end learning for self-driving cars", *arXiv preprint arXiv:1604.07316*, 2016.

[55] C. Hubschneider, A. Bauer, M. Weber, and J. M. Zöllner, "Adding navigation to the equation: Turning decisions for end-to-end vehicle control", in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–8.

[56] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-End Driving Via Conditional Imitation Learning", in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4693–4700.

[57] K. Mori, H. Fukui, T. Murase, T. Hirakawa, T. Yamashita, and H. Fujiyoshi, "Visual Explanation by Attention Branch Network for End-to-end Learning-based Self-driving", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 1577–1582.

[58] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "LIDAR-based driving path generation using fully convolutional neural networks", in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[59] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning", *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[60] C. Muench and D. M. Gavrila, "Composable Q-Functions for Pedestrian Car Interactions", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 905–912.

[61] N. De Moura, R. Chatila, K. Evans, S. Chauvier, and E. Dogan, "Ethical decision making for autonomous vehicles", in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 2006–2013. DOI: 10.1109/IV47402.2020.9304618.

[62] X. Lin, J. Zhang, J. Shang, Y. Wang, H. Yu, and X. Zhang, "Decision Making through Occluded Intersections for Autonomous Driving", in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2449–2455.

[63] C. Hubmann, N. Quetschlich, J. Schulz, J. Bernhard, D. Althoff, and C. Stiller, "A POMDP Maneuver Planner For Occlusions in Urban Scenarios", in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2172–2179.

[64] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning", *Nature*, vol. 518, pp. 529–533, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:205242740.

[65] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search", *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[66] 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)*, 2018, pp. 2156–2162. DOI: 10.1109/ITSC.2018.8569448.

[67] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. V. Gool, *End-to-End Urban Driving by Imitating a Reinforcement Learning Coach*, 2021. arXiv: 2108.08265 [cs.CV].

[68] I. Kostrikov, D. Yarats, and R. Fergus, *Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels*, 2021. arXiv: 2004.13649 [cs.LG].

[69] M. Moghadam and G. H. Elkaim, *A Hierarchical Architecture for Sequential Decision-Making in Autonomous Driving using Deep Reinforcement Learning*, 2019. arXiv: 1906.08464 [cs.RO].

[70] C.-J. Hoel, K. Wolff, and L. Laine, "Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning", *CoRR*, vol. abs/1803.10056, 2018. arXiv: 1803.10056. [Online]. Available: http://arxiv.org/abs/1803.10056.

[71] 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. arXiv: 1810.10469. [Online]. Available: http://arxiv.org/abs/1810.10469.

[72] G. Li, Y. Qiu, Y. Yang, Z. Li, S. Li, W. Chu, P. Green, and S. E. Li, "Lane Change Strategies for Autonomous Vehicles: A Deep Reinforcement Learning Approach Based on Transformer", *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2197–2211, 2023. DOI: 10.1109/TIV.2022.3227921.

[73] R. Valiente, M. Razzaghpour, B. Toghi, G. Shah, and Y. P. Fallah, *Prediction-aware and Reinforcement Learning based Altruistic Cooperative Driving*, 2022. arXiv: 2211.10585 [cs.RO].

[74] 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. arXiv: 2004.04450 [cs.AI].

[75] 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)*, 2019, pp. 3263–3268. DOI: 10.1109/ITSC.2019.8916922.

[76] 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. arXiv: 1909.11538. [Online]. Available: http://arxiv.org/abs/1909.11538.

[77] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement Learning with Iterative Reasoning for Merging in Dense Traffic", *CoRR*, vol. abs/2005.11895, 2020. arXiv: 2005.11895. [Online]. Available: https://arxiv.org/abs/2005.11895.

[78] M. Moghadam, A. Alizadeh, E. Tekin, and G. H. Elkaim, "An End-to-end Deep Reinforcement Learning Approach for the Long-term Short-term Planning on the Frenet Space", *CoRR*, vol. abs/2011.13098, 2020. arXiv: 2011.13098. [Online]. Available: https://arxiv.org/abs/2011.13098.

[79] Z. Qiao, K. Muelling, J. M. Dolan, P. Palanisamy, and P. W. Mudalige, "POMDP and Hierarchical Options MDP with Continuous Actions for Autonomous Driving at Intersections", *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2377–2382, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:54460595.

[80] W. Song, G.-m. Xiong, and H. Chen, "Intention-Aware Autonomous Driving Decision-Making in an Uncontrolled Intersection", *Mathematical Problems in Engineering*, vol. 2016, pp. 1–15, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:55511877.

[81] K. Shu, H. Yu, X. Chen, L. Chen, Q. Wang, L. Li, and D. Cao, "Autonomous Driving at Intersections: A Critical-Turning-Point Approach for Left Turns", *CoRR*, vol. abs/2003.02409, 2020. arXiv: 2003.02409. [Online]. Available: https://arxiv.org/abs/2003.02409.

[82] P. Agarwal, A. A. Rahman, P.-L. St-Charles, S. J. Prince, and S. E. Kahou, "Transformers in reinforcement learning: a survey", *arXiv preprint arXiv:2307.05979*, 2023.

[83] Z. Huang, H. Liu, J. Wu, W. Huang, and C. Lv, "Learning interaction-aware motion prediction model for decision-making in autonomous driving", in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2023, pp. 4820–4826.

[84] J. Fu, Y. Shen, Z. Jian, S. Chen, J. Xin, and N. Zheng, "InteractionNet: Joint Planning and Prediction for Autonomous Driving with Transformers", in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 9332–9339.

[85] 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. arXiv: 2208.12263 [cs.LG].

[86] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[87] D. Cao, J. Zhao, W. Hu, F. Ding, Q. Huang, and Z. Chen, "Attention enabled multi-agent DRL for decentralized volt-VAR control of active distribution system using PV inverters and SVCs", *IEEE transactions on sustainable energy*, vol. 12, no. 3, pp. 1582–1592, 2021.

[88] 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)*, 2021, pp. 559–566. DOI: 10.1109/ITSC48978.2021.9564720.

[89] H. Avilés, M. Negrete, A. Reyes, R. Machucho, K. Rivera, G. de-la-Garza, and A. Petrilli, "Autonomous Behavior Selection For Self-driving Cars Using Probabilistic Logic Factored Markov Decision Processes", *Applied Artificial Intelligence*, vol. 38, no. 1, p. 2 304 942, 2024.

[90] J. Lu, G. Alcan, and V. Kyrki, "Integrating Expert Guidance for Efficient Learning of Safe Overtaking in Autonomous Driving Using Deep Reinforcement Learning", *arXiv preprint arXiv:2308.09456*, 2023.

[91] A. Aksjonov and V. Kyrki, *A Safety-Critical Decision Making and Control Framework Combining Machine Learning and Rule-based Algorithms*, 2022. arXiv: 2201.12819 [cs.AI].

[92] N. S Kassem, S. F Saad, and Y. I Elshaaer, "Behavior Planning for Autonomous Driving: Methodologies, Applications, and Future Orientation", 2023.

[93] X. Gong, B. Wang, and S. Liang, "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.

[94] M. Sefati, J. Chandiramani, K. Kreiskoether, A. Kampker, and S. Baldi, "Towards tactical behaviour planning under uncertainties for automated vehicles in urban scenarios", in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–7. DOI: 10.1109/ITSC.2017.8317819.

[95] 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.

[96] R. Benekohal and J. Treiterer, "CARSIM. CAR-following model for SIMulation of traffic in normal and stop-and-go conditions", *Transportation Research Record*, pp. 99–111, Jan. 1988.

[97] M. Tideman and M. van Noort, "A simulation tool suite for developing connected vehicle systems", in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 713–718. DOI: 10.1109/IVS.2013.6629551.

[98] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator", in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[99] A. P. Capasso, G. Bacchiani, and D. Molinari, "Intelligent Roundabout Insertion using Deep Reinforcement Learning", *CoRR*, vol. abs/2001.00786, 2020. arXiv: 2001.00786. [Online]. Available: http://arxiv.org/abs/2001.00786.

[100] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-Aware Reinforcement Learning for Merging in Dense Traffic", *CoRR*, vol. abs/1906.11021, 2019. arXiv: 1906.11021. [Online]. Available: http://arxiv.org/abs/1906.11021.

[101] C.-J. Hoel, K. Wolff, and L. Laine, "Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning", *CoRR*, vol. abs/1803.10056, 2018. arXiv: 1803.10056. [Online]. Available: http://arxiv.org/abs/1803.10056.

[102] E. Leurent, *An Environment for Autonomous Driving Decision-Making*, https://github.com/eleurent/highway-env, 2018.

[103] *CARLA Challenge*, https://carlachallenge.org/, Accessed: 2023.

[104] F. Codevilla, E. Santana, A. Lopez, and A. Gaidon, "Exploring the Limitations of Behavior Cloning for Autonomous Driving", in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9328–9337. DOI: 10.1109/ICCV.2019.00942.

[105] 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", *IEEE Transactions on Intelligent Vehicles*, pp. 1–20, 2023. DOI: 10.1109/TIV.2023.3312777.

[106] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics", 1998.

[107] NVIDIA, *PhysX Vehicle SDK*, https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/Vehicles.html, Accessed: 2024, 2020.

[108] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System", in *ICRA workshop on open source software*, Kobe, Japan, 2009, p. 5.

[109] S. Yu, M. Hirche, Y. Huang, H. Chen, and F. Allgöwer, "Model predictive control for autonomous ground vehicles: a review", *Autonomous Intelligent Systems*, vol. 1, pp. 1–17, 2021.

[110] M. Brezak and I. Petrović, "Real-time approximation of clothoids with bounded error for path planning applications", *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 507–515, 2013.

[111] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice", in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4889–4895.

[112] J.-W. Lee and B. Litkouhi, "A unified framework of the automated lane centering/changing control for motion smoothness adaptation", in *2012 15th international IEEE conference on intelligent transportation systems*, IEEE, 2012, pp. 282–287.

[113] L. Han, H. Yashiro, H. T. N. Nejad, Q. H. Do, and S. Mita, "Bezier curve based path planning for autonomous vehicle in urban environment", in *2010 IEEE intelligent vehicles symposium*, IEEE, 2010, pp. 1036–1042.

[114] Z. Liang, G. Zheng, and J. Li, "Automatic parking path optimization based on bezier curve fitting", in *2012 IEEE International Conference on Automation and Logistics*, IEEE, 2012, pp. 583–587.

[115] H. Kano and H. Fujioka, "B-spline trajectory planning with curvature constraint", in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 1963–1968.

[116] E. Magid, R. Lavrenov, and A. Khasianov, "Modified Spline-based Path Planning for Autonomous Ground Vehicle.", in *ICINCO (2)*, 2017, pp. 132–141.

[117] C. Götte, M. Keller, T. Nattermann, C. Haß, K.-H. Glander, and T. Bertram, "Spline-based motion planning for automated driving", *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9114–9119, 2017.

[118] M. Elbanhawi, M. Simic, and R. Jazar, "Solutions for path planning using spline parameterization", *Nonlinear Approaches in Engineering Applications: Advanced Analysis of Vehicle Related Technologies*, pp. 369–399, 2016.

[119] M. Samuel, M. Hussein, and M. B. Mohamad, "A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle", *International Journal of Computer Applications*, vol. 135, no. 1, pp. 35–38, 2016.

[120] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing", in *2007 American control conference*, IEEE, 2007, pp. 2296–2301.

[121] V. Girbés, L. Armesto, J. Tornero, and J. E. Solanes, "Smooth kinematic controller vs. Pure-pursuit for non-holonomic vehicles", in *Towards Autonomous Robotic Systems: 12th Annual Conference, TAROS 2011, Sheffield, UK, August 31–September 2, 2011. Proceedings 12*, Springer, 2011, pp. 277–288.

[122] M. Kissai, B. Monsuez, and A. Tapus, "Review of integrated vehicle dynamics control architectures", in *2017 European Conference on Mobile Robots (ECMR)*, IEEE, 2017, pp. 1–8.

[123] J. Ni and J. Hu, "Dynamics control of autonomous vehicle at driving limits and experiment on an autonomous formula racing car", *Mechanical Systems and Signal Processing*, vol. 90, pp. 154–174, 2017.

[124] E. Mousavinejad, Q.-L. Han, F. Yang, Y. Zhu, and L. Vlacic, "Integrated control of ground vehicles dynamics via advanced terminal sliding mode control", *Vehicle System Dynamics*, vol. 55, no. 2, pp. 268–294, 2017.

[125] Y. Liu and D. Cui, "Application of optimal control method to path tracking problem of vehicle", *SAE International Journal of Vehicle Dynamics, Stability, and NVH*, vol. 3, no. 10-03-03-0014, pp. 209–219, 2019.

[126] K. Makantasis and M. Papageorgiou, "Motorway path planning for automated road vehicles based on optimal control methods", *Transportation Research Record*, vol. 2672, no. 19, pp. 112–123, 2018.

[127] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: a toolkit for real domain data visualization", *Telecommunication Systems*, vol. 60, pp. 337–345, 2015.

[128] M. Samuel, M. Hussein, and M. Mohamad, "A Review of some Pure-Pursuit based Path Tracking Techniques for Control of Autonomous Vehicle", *International Journal of Computer Applications*, vol. 135, pp. 35–38, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:225532.

[129] J. López, C. Otero, R. Sanz, E. Paz, E. Molinos, and R. Barea, "A new approach to local navigation for autonomous driving vehicles based on the curvature velocity method", in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1751–1757. DOI: 10.1109/ICRA.2019.8794380.

[130] 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)*, 2023, pp. 1–6.

[131] J. Villagra, V. Milanés, J. Pérez, and J. Godoy, "Smooth path and speed planning for an automated public transport vehicle", *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 252–265, 2012.

[132] Y. Jiang, Z. Liu, D. Qian, H. Zuo, W. He, and J. Wang, "Robust Online Path Planning for Autonomous Vehicle Using Sequential Quadratic Programming", in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 175–182. DOI: 10.1109/IV51971.2022.9827017.

[133] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors", in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[134] Y. Zhang, H. Sun, J. Zhou, J. Pan, J. Hu, and J. Miao, "Optimal Vehicle Path Planning Using Quadratic Optimization for Baidu Apollo Open Platform", in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Oct. 2020. DOI: 10.1109/iv47402.2020.9304787. [Online]. Available: http://dx.doi.org/10.1109/IV47402.2020.9304787.

[135] 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. arXiv: 2211.07545 [cs.RO].

[136] 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. DOI: 10.1109/TNNLS.2022.3142822.

[137] deepsense-ai, *carla-birdeye-view: A Python library for rendering bird-eye view of CARLA simulator*, https://github.com/deepsense-ai/carla-birdeye-view, Last Accessed: 2023, 2020.

[138] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using NetworkX", Jan. 2008. [Online]. Available: https://www.osti.gov/biblio/960616.

[139] CARLA, *Traffic Manager. Autopilot.* https://carla.readthedocs.io/en/latest/adv_traffic_manager/, Accessed: 2024, 2024.

[140] 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, ISSN: 1424-8220. DOI: 10.3390/s22218373. [Online]. Available: https://www.mdpi.com/1424-8220/22/21/8373.

[141] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.

[142] MathWorks, *RoadRunner*, Software for Autonomous Vehicle Simulation, 2021. [Online]. Available: https://www.mathworks.com/products/roadrunner.html.

[143] 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, ISSN: 1424-8220. DOI: 10.3390/s20216121. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6121.

[144] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer Learning in Deep Reinforcement Learning: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 13 344–13 362, 2023. DOI: 10.1109/TPAMI.2023.3292075.

[145] P. Yadav, A. Mishra, and S. Kim, "A Comprehensive Survey on Multi-Agent Reinforcement Learning for Connected and Automated Vehicles", *Sensors*, vol. 23, no. 10, 2023, ISSN: 1424-8220. DOI: 10.3390/s23104710. [Online]. Available: https://www.mdpi.com/1424-8220/23/10/4710.

[146] G. Li, Y. Qiu, Y. Yang, Z. Li, S. Li, W. Chu, P. Green, and S. E. Li, "Lane change strategies for autonomous vehicles: a deep reinforcement learning approach based on transformer", *IEEE Transactions on Intelligent Vehicles*, 2022.

[147] E. Parisotto, F. Song, J. Rae, R. Pascanu, C. Gulcehre, S. Jayakumar, M. Jaderberg, R. L. Kaufman, A. Clark, S. Noury, *et al.*, "Stabilizing transformers for reinforcement learning", in *International conference on machine learning*, PMLR, 2020, pp. 7487–7498.

[148] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress", *Artificial Intelligence*, vol. 297, p. 103 500, 2021.

[149] J. Cai, W. Deng, H. Guang, Y. Wang, J. Li, and J. Ding, "A survey on data-driven scenario generation for automated vehicle testing", *Machines*, vol. 10, no. 11, p. 1101, 2022.

[150] H. Zhang, W. Assawinchaichote, and Y. Shi, "New PID parameter autotuning for nonlinear systems based on a modified monkey–multiagent DRL algorithm", *IEEE Access*, vol. 9, pp. 78 799–78 811, 2021.

[151] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications", *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.