Neuro-symbolic deep reinforcement learning for safe urban driving using low-cost sensors.

Mohamad Albilani

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAS008

Thèse de doctorat

Neuro-symbolic Deep Reinforcement Learning For Safe Urban Driving Using Low-Cost Sensors

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Telecom SudParis

École doctorale n°626 École doctorale de l'institut polytechnique de Paris (EDIPP)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 22/04/2024, par

MOHAMAD ALBILANI

Composition du Jury :

Maryline Laurent Professeur, Télécom SudParis	Présidente
Lounis Adouane Professeur, Université de Technologie de Compiègne	Rapporteur
Fawzi Nashashibi Directeur de recherche, Inria Paris	Rapporteur
Sascha Hornauer Maître de conférences, Mines Paris PSL	Examinateur
Philippe Xu Professeur, ENSTA Paris	Examinateur
Amel Bouzeghoub Professeur, Télécom SudParis	Directrice de thèse
Jean-Francois Beraud Directeur d'innovation, Avisto	Invité
Pierre Pacchioni Directeur technique, Avisto	Invité

Acknowledgements

I express my deep appreciation for my thesis supervisor, Mme. Amel Bouzeghoub, Professor at Telecom Sudparis, for her invaluable guidance and unwavering support throughout my thesis. I have gained immense knowledge from her expertise, and she has played a crucial role in shaping me into the researcher I am today. Her expertise and unwavering commitment to my academic success have motivated me to pursue excellence in all aspects of my research. She has taught me the importance of diligence, critical thinking, and attention to detail—skills that I will carry forward as I strive for new heights in my academic and professional journey.

I also wish to extend my gratitude to Mr. Pierre Pacchioni, Technical Director, and Mr. Jean-Francois Beraud, Doctor and Director of R&D at Avisto, for their unwavering support during this journey. Their pivotal roles in facilitating my progress at Avisto cannot be overstated. They provided invaluable guidance, resources, and encouragement, ensuring that I navigated through challenges effectively. Thanks to their mentorship, I maintained a strong link between the academic world and the industrial sector, allowing me to apply theoretical knowledge to practical settings and vice versa.

Furthermore, I want to express my heartfelt gratitude to the esteemed members of the jury who dedicated their time and expertise to examining and evaluating my work. Special thanks are due to Mr. Fawzi Nashashibi, Research Director at Inria, and Mr. Lounis Adouane, Professor at UTC, for their generous contributions, insightful ideas, and valuable comments. Their constructive feedback and guidance have enriched the quality of my research and its impact.

I extend my gratitude to the members of the jury, including Mme. Maryline Laurent, Professor at Telecom Sudparis, Mr. Philippe Xu, Professor at ENSTA Paris, and Mr. Sascha Hornauer, Associate Professor at PSL, for their integral role in evaluating my work. Their expertise and dedication to the assessment process are deeply appreciated.

To my wife, Inass, you are my source of inspiration, and I am endlessly grateful for your unwavering support and sacrifices. Thank you for being by my side throughout this journey, your encouragement has been my strength.

Résumé

La recherche effectuée dans cette thèse concerne le domaine de la conduite urbaine sûre, en utilisant des méthodes de fusion de capteurs et d'apprentissage par renforcement pour la perception et le contrôle des véhicules autonomes. L'évolution et l'intégration généralisée des technologies d'apprentissage automatique ont principalement propulsé la prolifération des véhicules autonomes ces dernières années. Cependant, des progrès substantiels sont nécessaires avant d'atteindre une adoption généralisée par le grand public. Pour accomplir son automatisation, les véhicules autonomes nécessitent l'intégration d'une série de capteurs coûteux, comprenant des caméras, des radars, des LiDARs et des capteurs ultrasoniques. En plus de leur fardeau financier, ces capteurs présentent une sensibilité aux variations telles que la météo, une limitation non partagée par les conducteurs humains qui peuvent naviguer dans des conditions diverses en se fiant à une vision frontale simple. Par ailleurs, l'avènement des algorithmes neuronaux de prise de décision constitue l'intelligence fondamentale des véhicules autonomes. Les solutions d'apprentissage profond par renforcement, facilitant l'apprentissage de la politique du conducteur de bout en bout, ont trouvé application dans des scénarios de conduite élémentaires, englobant des tâches telles que le maintien dans la voie, le contrôle de la direction et la gestion de l'accélération. Cependant, il s'avère que ces algorithmes sont coûteux en temps d'exécution et nécessitent de large ensembles de données pour un entraînement efficace. De plus, la sécurité doit être prise en compte tout au long des phases de développement et de déploiement des véhicules autonomes. Par ailleurs, l'avènement des algorithmes neuronaux de prise de décision constitue l'intelligence fondamentale des véhicules autonomes. Les solutions d'apprentissage profond par renforcement, facilitant l'apprentissage de la politique du conducteur de bout en bout, ont trouvé application dans des scénarios de conduite élémentaires, englobant des tâches telles que le maintien dans la voie, le contrôle de la direction et la gestion de l'accélération. Cependant, il s'avère que ces algorithmes sont coûteux en temps d'exécution et nécessitent de large ensembles de données pour un entraînement efficace. De plus, la sécurité doit être prise en compte tout au long des phases de développement et de déploiement des véhicules autonomes.

Cette thèse se propose de relever les trois défis suivants qui concernent les phases de perception et de prise de décision : (i) Comment peut-on obtenir une localisation absolue et de haute précision d'un VA en utilisant des capteurs à faible coûts tels que GPS et IMU ? (ii) Dans le contexte du stationnement automatique, comment éviter efficacement les obstacles statiques et dynamiques de la scène, en garantissant une opération de stationnement sûre ? En particulier, comment préserver le respect des caractéristiques spécifiques de la norme ISO-16787:2017 [Standardization, 2017a], concernant l'angle d'inclinaison et la déviation, pour s'aligner sur les réglementations industrielles et les protocoles de sécurité ? (iii) Comment accélérer la conver-

gence de l'algorithme d'apprentissage tout en en étant réactif face à des situations imprévues et hautement dynamiques, en particulier dans des scénarios de conduite urbaine ?

La thèse répond à ces questions à travers les contributions suivantes : La première contribution améliore la localisation des véhicules en fusionnant les mesures des capteurs GPS et IMU à faible coût avec une adaptation d'un filtre de Kalman, ES-EKF, et une réduction du bruit des mesures IMU. Cette méthode se révèle efficace dans les environnements urbains caractérisés par des interruptions de signal et des niveaux de bruit élevés, en atténuant l'impact négatif du bruit dans les mesures des capteurs IMU, ce qui permet de maintenir la précision et la robustesse de la localisation. L'algorithme est déployé et testé en utilisant des données de vérité terrain sur un microcontrôleur embarqué, le STM32 Nucleo, et a atteint un niveau de précision de 92 % sur une route en conditions réelles avec un temps d'exécution d'environ 20 μs. Ce pourcentage représente la proportion de points de données où la position estimée (X, Y) par l'ES-EKF se trouve dans un intervalle de confiance de 0,5 mètre par rapport à la position réelle correspondante.

La deuxième contribution propose l'algorithme DPPO-IL (Dynamic Proximal Policy Optimization with Imitation Learning), conçu pour faciliter le stationnement automatisé de bout en bout en accordant une attention toute particulière à la sécurité. Cet algorithme apprend à exécuter des manœuvres de stationnement optimales tout en naviguant entre des d'obstacles statiques et dynamiques grâce à un entraînement complet intégrant des données simulées et réelles. Il atteint un taux de réussite de 90 %, même dans les configurations les plus complexes, ce qui témoigne de sa capacité d'adaptation. La troisième contribution est un framework de conduite urbaine de bout en bout appelé Guided Hierarchical Reinforcement Learning (GHRL). Il intègre des données de vision et de localisation ainsi que des démonstrations d'experts exprimées avec des règles ASP (Answer Set Programming) pour guider la politique d'exploration de l'apprentissage par renforcement hiérarchique et accélérer la convergence de l'algorithme. Lorsqu'une situation critique se produit, le système s'appuie également sur des règles liées à la sécurité pour faire des choix judicieux dans des conditions imprévisibles ou dangereuses. GHRL est évalué sur le jeu de données NoCrash du simulateur Carla et les résultats montrent qu'en incorporant des règles logiques, GHRL obtient de meilleures performances que les algorithmes de l'état de l'art.

Mots clés : Fusion de Données, Apprentissage par Renforcement Neuro-Symbolique, Apprentissage par Renforcement Hiérarchique, Véhicule Autonome, Conduite Urbaine Sûre, Règles ASP

Abstract

The research conducted in this thesis is centered on the domain of safe urban driving, employing sensor fusion and reinforcement learning methodologies for the perception and control of autonomous vehicles (AV). The evolution and widespread integration of machine learning technologies have primarily propelled the proliferation of autonomous vehicles in recent years. However, substantial progress is requisite before achieving widespread adoption by the general populace. To accomplish its automation, autonomous vehicles necessitate the integration of an array of costly sensors, including cameras, radars, LiDARs, and ultrasonic sensors. In addition to their financial burden, these sensors exhibit susceptibility to environmental variables such as weather, a limitation not shared by human drivers who can navigate diverse conditions with a reliance on simple frontal vision. Moreover, the advent of decision-making neural network algorithms constitutes the core intelligence of autonomous vehicles. Deep Reinforcement Learning solutions, facilitating end-to-end driver policy learning, have found application in elementary driving scenarios, encompassing tasks like lane-keeping, steering control, and acceleration management. However, these algorithms demand substantial time and extensive datasets for effective training. In addition, safety must be considered throughout the development and deployment phases of autonomous vehicles.

This thesis addresses the following challenges: (i) How can absolute, high-precision AV localization be achieved using cost-effective sensors such as GPS and IMU? (ii) In the context of Automatic Parking Mode, how can static and dynamic obstacles within the scene be effectively avoided, ensuring a safe parking operation? In particular, how to maintain the adherence to specific aspects of the ISO-16787:2017 standard, including the inclination angle and the deviation, to align with industry regulations and safety protocols. (iii) How can we speed up the convergence of the learning algorithm while ensuring responsiveness to unforeseen and highly dynamic situations, particularly in urban driving scenarios?

This thesis seeks to answer these questions through the following contributions: The first contribution of this thesis improves vehicle localization by fusing data from GPS and IMU sensors with an adaptation of a Kalman filter, ES-EKF, and a reduction of noise in IMU measurements. This method excels in urban environments marked by signal obstructions and elevated noise levels, effectively mitigating the adverse impact of noise in IMU sensor measurements, thereby maintaining localization accuracy and robustness. The algorithm is deployed and tested employing ground truth data on an embedded microcontroller, the STM32 Nucleo, and has achieved an accuracy level of 92% on a real-world road and a swift execution time of around 20 μs. This percentage represents the proportion of data points where the estimated position (X, Y) from the ES-EKF falls within a confidence level of a 0.5m distance threshold of the

corresponding ground truth position.

The second contribution introduces the DPPO-IL (Dynamic Proximal Policy Optimization with Imitation Learning) algorithm, designed to facilitate end-to-end automated parking while maintaining a steadfast focus on safety. This algorithm acquires proficiency in executing optimal parking maneuvers while navigating static and dynamic obstacles through exhaustive training incorporating simulated and real-world data. It attains a commendable 90% success rate even in the most challenging settings, underscoring its versatility and adaptability. The third contribution is an end-to-end urban driving framework called GHRL. It incorporates vision and localization data and expert demonstrations expressed in the Answer Set Programming (ASP) rules to guide the hierarchical reinforcement learning (HRL) exploration policy and speed up the learning algorithm's convergence. When a critical situation occurs, the system relies on safety rules, which empower it to make prudent choices amidst unpredictable or hazardous conditions. GHRL is evaluated on the Carla NoCrash benchmark, and the results show that by incorporating logical rules, GHRL achieved better performance over state-of-the-art algorithms.

Keywords : Data Fusion, Neuro-Symbolic Reinforcement Learning, Hierarchical Reinforcement Learning, Self Driving Car, Safe Urban Driving, ASP Rules

Table des matières

Liste des figures

Liste des tableaux

Acronymes et anglicismes

ASP *Answer Set Programming*
DDPG *Deep Deterministic Policy Gradient*
DRL *Deep Reinforcement Learning*
ECU *Electronic Control Unit*
EIC *Environmental Interface Card*
GPS *Global Positioning System*
IL *Imitation Learning*
IMU *Inertial Measurement Unit*
LiDAR *Light Detection and Ranging*
POCA *MultiAgent POsthumous Credit Assignment*
PPO *Proximal Policy Optimization*
RADAR *Radio Detection and Ranging*
SAC *Soft Actor-Critic*

Chapitre 1

Introduction

Contents

1 Introduction

Autonomous driving has come a long way since its inception in the 1980s, with significant progress made in recent years thanks to the support of major companies like Waymo [1] and Tesla [2]. While the technology is not yet fully mature, vehicles can now drive hundreds of kilometers in certain conditions without human intervention. Autonomous systems possess a crucial advantage in terms of safety, as they exhibit faster responsiveness than human drivers, leading to a substantial reduction in the number of accidents [Wang, 2020a]. Additionally, by reducing congestion, these vehicles can improve air quality and save drivers time on the road [Seuwou, 2020]. However, we are not yet at the stage of fully autonomous vehicles. The ultimate goal of autonomous driving is complete automation. Nonetheless, several intermediate steps must be taken, as defined by the National Highway Traffic Safety Administration (NHTSA) (Figure 1.1). While levels 0 to 2 refer to advanced driver assistance, level 3 represents the delegation of driving tasks to the autonomous system, requiring the human driver to be able to regain control at any time. Levels 4 and 5 correspond to fully autonomous driving, with level 4 allowing for autonomous driving in specific conditions, while level 5 allows for autonomous driving in any situation. However, realizing these levels is still accompanied by legal and ethical challenges, including the issue of liability in the event of an accident involving an autonomous vehicle. Nevertheless, level 5 autonomy is becoming increasingly achievable with every passing day.

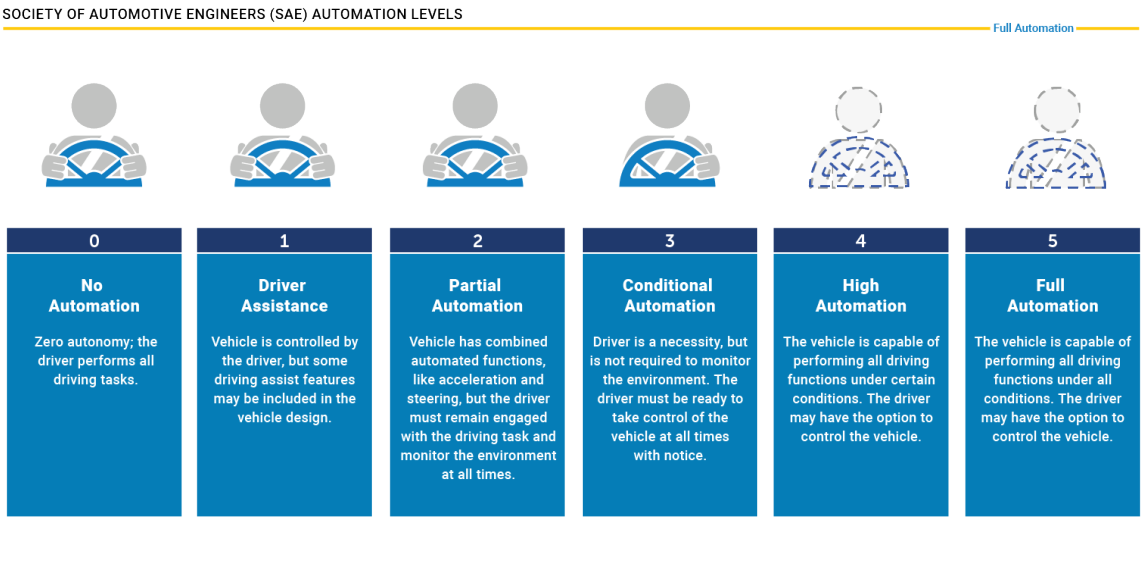


FIGURE 1.1 – Levels of automation for autonomous vehicles defined by the NHTSA. https://www.nhtsa.gov/document/levels-automation

1. https://waymo.com/
2. https://www.tesla.com/en_eu

1.1 Problem Statement

Safety is a concern throughout the development and deployment of self-driving cars, underlining the need for a robust framework that prioritizes functionality and safety. As autonomous vehicles can potentially reduce accidents and save lives significantly, it is crucial to address the alarming statistic that 94% of severe accidents result from human error [Wang, 2020a]. By mitigating the risk of human error, self-driving cars promise to reduce roadway accidents. However, achieving this promise requires navigating unique safety challenges that arise during the journey to fully autonomous vehicles. The work accomplished by [Xu, 2022] on Safebench has identified several safety challenges. These challenges include ensuring the reliability of software, anticipating and managing unforeseen interactions with other road users, and accounting for the influence of environmental factors.

The architecture underpinning self-driving cars plays a pivotal role in addressing these safety challenges and fostering a secure and trustworthy autonomous driving experience [Cina, 2023].

The architecture involves a multifaceted approach to address safety challenges and ensure a secure and reliable driving experience. Passive safety measures like airbags have provided some relief, but their effectiveness has reached technological limits. Consequently, the focus has shifted to active safety alternatives. This active safety architectural framework integrates and orchestrates many components and systems, guided by principles like modularity, redundancy, fault tolerance, perception, and decision-making. These principles collectively aim to establish the highest levels of safety in autonomous operations.

This architectural design strongly emphasizes harmonious module integration, allowing for individual testing and validation, thus ensuring unwavering reliability while minimizing vulnerabilities [Hataba, 2022]. Moreover, it enhances system resilience by implementing component redundancy and deploying fail-safe mechanisms to mitigate potential faults or failures [Hataba, 2022]. The architecture also elevates the perceptual capabilities of these vehicles by intelligently fusing data from diverse sensors [Alsuwian, 2022].

Data fusion and decision-making algorithms constitute an indispensable facet of the architecture. This symbiotic relationship is instrumental in providing self-driving vehicles with the necessary capabilities to navigate the roadways securely and responsibly. As emphasized in [Giacalone, 2019], data fusion plays a paramount role in ensuring safety within autonomous driving. Sensors are the crucial components that perceive the external environment for automated driving systems. However, relying solely on sensor fusion does not ensure the safety of AD cars in complex traffic environments. It orchestrates the synthesis of multifaceted information streams, affording self-driving vehicles an unparalleled and holistic perception of their immediate environment as asserted in the survey [Zhang, 2023]. This multifaceted perception is indispensable for minimizing the vulnerabilities associated with incomplete or erroneous data, which, in turn, significantly contributes to ensuring the safety of autonomous operations [Farrell, 2017]. Data fusion is the nexus through which diverse sensor modalities, such as LiDAR, cameras, radars, Global Positioning System (GPS), and ultrasonic sensors, converge. This fusion transcends the limitations of individual sensor inputs, facilitating comprehensive environmental awareness. Data fusion equips self-driving vehicles with an all-encompassing understanding of their surroundings

by processing and harmonizing these heterogeneous data sources [Nweke, 2019]. This expansive comprehension plays a pivotal role in analyzing spatiotemporal factors and mitigating potential risks, as the vehicle can detect objects and obstacles from myriad angles, thus substantially reducing the likelihood of accidents [Xu, 2023]. Integrating data from an array of sensors effectively eradicates blind spots, a critical concern in road safety [Shao, 2023]. The minimization of blind spots empowers autonomous vehicles to detect objects and potential hazards from all perspectives [Yeong, 2021]. This integration fosters a heightened level of safety, particularly during critical maneuvers such as lane changes and highway merging, where obscured obstacles could pose a significant threat [Xu, 2023]. Beyond providing a static environmental snapshot, data fusion endows self-driving vehicles with dynamic adaptability [Kovacova, 2022]. It enables the vehicle to execute a well-defined decision-making algorithm to adapt its behavior in response to real-time environmental changes. For instance, when confronted with heavy traffic, adverse weather conditions, or unexpected road closures [Liu, 2021], the adaptability facilitated by data fusion empowers the vehicle to make informed decisions and execute maneuvers that prioritize safety.

In the domain of data fusion methods for autonomous vehicle localization, the Kalman Filter is a widely adopted approach [Kalman, 1960]. It relies on statistical and control theory, employing linear-quadratic estimations to calculate the state of a process while minimizing mean square error. The Kalman Filter computes optimal positions using noisy sensor data, incorporating vehicle dynamics and kinematics equations [Gelb, 1974]. However, it has drawbacks, including its linearity assumption, sensitivity to Gaussian probability distributions, and sensitivity to sensor noise. To address nonlinearity, alternatives like the Extended Kalman Filter (EKF) [Hoshiya, 1984] and Unscented Kalman Filter (UKF) [Wan, 2001] are preferred. EKF linearizes state transitions, while UKF approximates posterior probability densities. However, they also have limitations, including a trade-off between accuracy and computational efficiency. Sensor fusion plays a pivotal role in enhancing the perception capabilities of autonomous vehicles, enabling them to make informed decisions while prioritizing safety. Particle Filters (PF) offer another option for multi-sensor fusion [Wan, 2001]. However, they can be computationally demanding, particularly with many particles, and require careful tuning to balance accuracy. The right balance between computational efficiency and precision remains a challenge in PF-based fusion. In parallel, recent advancements have introduced neural networks (NN) as practical tools for data fusion, but they come with limitations. NN-based methods often necessitate substantial labeled training data, which can be challenging and costly to acquire, especially for new environments or maps. Moreover, these methods frequently rely on high-cost sensors, such as LiDAR or cameras, which can hinder their practicality for low-cost autonomous systems. Additionally, deep neural networks can be computationally intensive, potentially limiting their use in real-time applications, especially on resource-constrained hardware.

Shifting our focus to decision-making, we must recognize that decision-making algorithms lie at the core of autonomous vehicles' intelligence [Li, 2022a]. These algorithms guide their actions and responses in real-time scenarios, with safety as a concern. Whether route planning, obstacle avoidance, or dynamic emergency responses, every decision is made with an unwa-

vering commitment to prioritize safety [Jamgochian, 2022]. Numerous promising methods for decision-making already exist, encompassing various techniques and approaches suited for specific scenarios and challenges [Li, 2022b]. While combining different methods for various driving situations is conceivable, adopting such an iterative approach poses the risk of creating an exceedingly complex system. In the context of autonomous driving, machine learning offers two key approaches for decision-making : supervised learning and deep reinforcement learning (DRL) [Li, 2022b]. Supervised learning involves training models directly from human driver demonstrations, but challenges arise due to potential distributional shifts between training and real-world test data [Burton, 2017]. In contrast, DRL aligns naturally with the decision-making task for autonomous driving. It deals with sequential decision-making problems aimed at maximizing utility or rewards, empowering autonomous vehicles to learn and adapt their decisions based on real-time feedback from the environment [Kiran, 2021].

However, DRL solutions for end-to-end driver policy learning have been applied to simple driving scenarios, like lane keeping, steering control, and managing acceleration [Kendall, 2019]. A primary challenge in DRL is guaranteeing the safety of autonomous driving (AD) systems, especially during the exploration phase [Gu, 2022]. When autonomous Vehicles (AV) engage in exploratory behavior in urban driving scenarios, they risk taking actions that could result in catastrophic consequences, potentially jeopardizing passengers' lives. Moreover, DRL typically demands substantial training data [Sutton, 2018]. Also, acquiring such extensive datasets for AD can be exceedingly challenging [Kiran, 2021]. These combined difficulties confine the training of AVs primarily to simulation environments and make their transition to real-world driving situations practically unfeasible.

Several distinct approaches exist to achieve safety in DRL [Garcıa, 2015]. One approach entails restricting the expected cost [Achiam, 2017]. Alternatively, using the loss function, another method maximizes the safety constraints [Xu, 2018]. Moreover, penalties can be introduced to discourage the agent from transgressing safety boundaries [Pham, 2018][Memarian, 2021]. An alternative is the creation of a more complex reward structure through temporal logic [De Giacomo, 2019][Jiang, 2021][Hasanbeig, 2019][Den Hengst, 2022]. These approaches consolidate safety concerns into a complex loss function, making the optimization more challenging. In contrast, the shielding technique [Alshiekh, 2018] deploys a shield to directly forestall the agent from taking actions that might potentially breach safety regulations during the exploration phase of DRL [Yang, 2023]. However, the shield's strictness may sometimes hinder the learning agent's ability to effectively explore the environment and discover its optimal policy [Könighofer, 2022].

In summary, while various sensor fusion and decision-making methods exist for autonomous driving, each with its strengths and limitations, the overarching challenge is to develop comprehensive and robust systems. These systems must navigate the complexities of the real world, guarantee safety, and ensure efficient use of affordable sensors while making critical decisions. This necessitates ongoing research and development efforts to bridge the gap between existing methods and the demanding requirements of autonomous driving in diverse and dynamic environments.

1.2 Avisto

This work is a collaboration between Telecom SudParis and AViSTO, an R&D and software engineering company founded in 1999 in Sophia Antipolis, a region in the French Riviera. In 2004, it became a subsidiary of the ADVANS Group. Over the years, AViSTO has expanded its portfolio and honed its expertise in various software technologies and services. The company's core focus lies in developing software solutions, specializing in "Object" technologies, including Java/JEE, C++, C#, .Net, Core, Python, and PHP. Additionally, AViSTO excels in web development, employing technologies such as Angular, ReactJS, and VueJS, as well as mobile application development for Android, iOS, and cross-platform solutions. AViSTO maintains a substantial workforce comprising between 200 and 249 employees, underscoring its significance in the software engineering industry. In 2019, the company reported a turnover of €17,659,300.00, further highlighting its robust presence. One distinguishing characteristic of AViSTO is its capacity to oversee the entire software development life cycle, encompassing requirements gathering and preparations for future developments.

Furthermore, the company has diversified its service offerings, incorporating expertise in DevOps, quality assurance, data science, and cybersecurity domains. Beyond its core software development competencies, AViSTO can undertake projects across various domains, including information systems, web development, telecommunications, industrial software, embedded applications, and networks.

1.2.1 KarLAB

KARlab is a collaborative multi-disciplinary research and development (R&D) project that commenced in 2018, involving the active participation of ELSYS Design, Mécagine, and AVISTO. The primary objective of this endeavor is to create an intelligent karting system designed to familiarize drivers with the anticipated interactions with future vehicles. The project serves to advance the technological expertise within the ADVANS group, particularly in the domains of electronic systems (embedded systems) and software engineering, encompassing immersive technologies, artificial intelligence (AI), big data, and more, all applied to the automotive industry.

In addition to its R&D focus, the project offers users a unique experience in augmented and virtual reality karting races. Equipped with a mixed reality headset, specifically the Microsoft Hololens [3], the kart driver engages with virtual elements seamlessly integrated into their real-world environment. In a manner reminiscent of the popular video game "Mario Kart," these virtual interactions have tangible consequences within the real-world racing context, impacting the performance of the player's kart and that of their competitors.

The evolution of the KARlab project, depicted in Figure 1.2, spans from its initial Proof Of Concept (PoC) phase to its current status, where the work is being actively implemented on an actual kart, representing a significant advancement in the project's development.

This project involves the interaction of three distinct entities : the kart, the game server, and the Hololens. The communication between these entities is facilitated through a WiFi net-

3. https ://www.microsoft.com/en-us/hololens

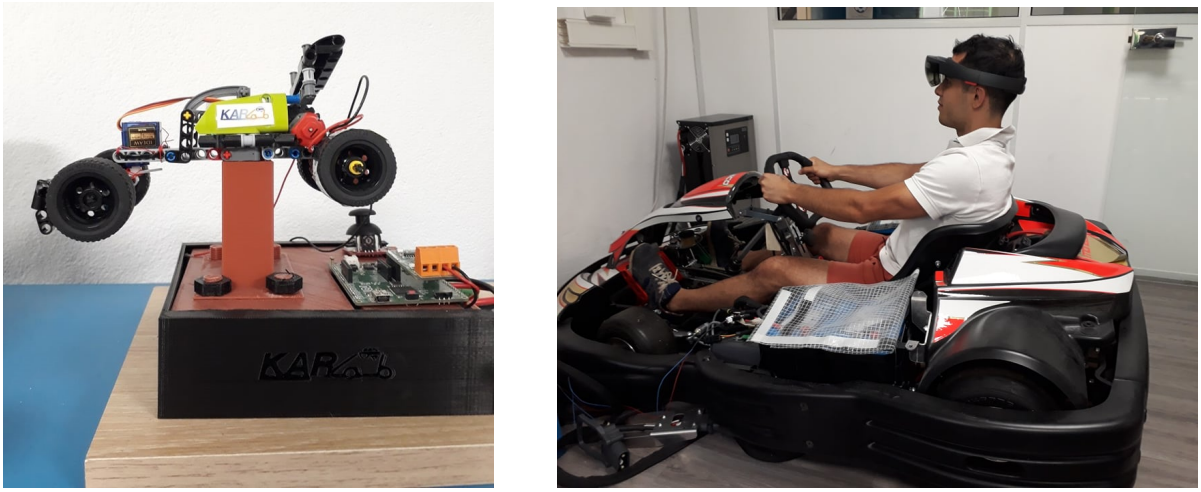


FIGURE 1.2 – Projet KARlab : Initial and actual states

work, with data transmission conducted using the TCP protocol. The communication involved components can transmit messages in the "*FlatBuffers*" format. A comprehensive overview of the interactions between these entities is illustrated in Figures 1.3 and 1.4.

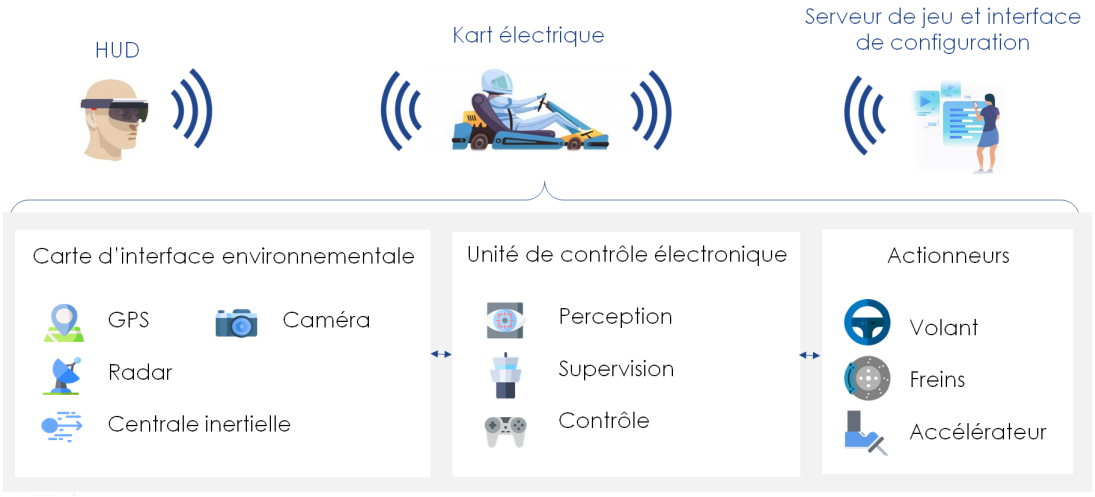


FIGURE 1.3 – Diagram summarizing the communications between the entities of the KARlab project, as well as the modules present on board the kartt

The onboard systems integrated within the kart are responsible for task allocation. This distribution of responsibilities encompasses three main components. The first component revolves around the Environmental Interface Card (EIC), enabling the kart to gather environmental data, such as its position and speed, through various sensors, including RADAR, GPS, and the inertial measurement unit (IMU). The EIC engages in data exchange with the Electronic Control Unit (ECU). The ECU assumes the tasks of data synthesis, information analysis, and the oversight of Kart control processes. The final component involves mechanical elements, with actuators responsible for manipulating the kart's wheels' angles, acceleration, and braking.

In the context of communication between the kart and the driver, the Hololens serves as the

conduit for real-time information delivery to the driver, enabling them to perceive notifications in their field of vision, such as detecting panels or real-time speed updates. In a masterful capacity, the driver exchanges information with all the karts participating in the race on the track. Furthermore, as depicted in Figure 1.4, the karts are interconnected, allowing them to engage in direct data exchange. Ultimately, a central server orchestrates and coordinates the overall game dynamics.

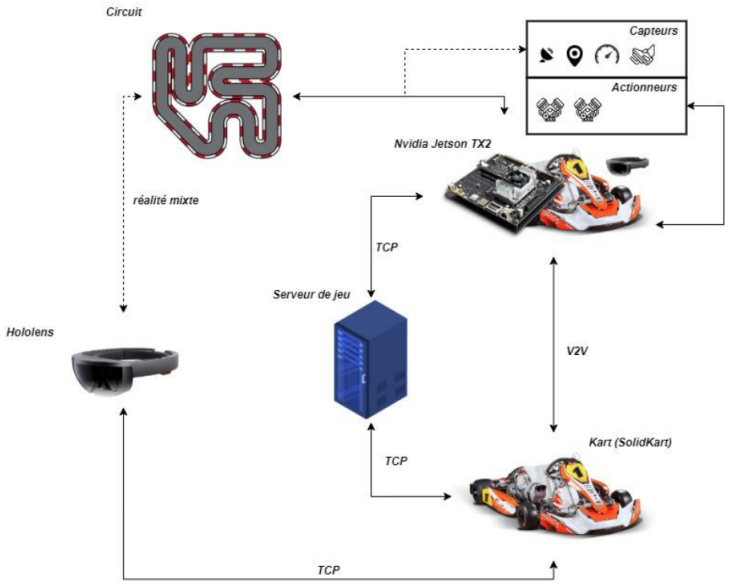
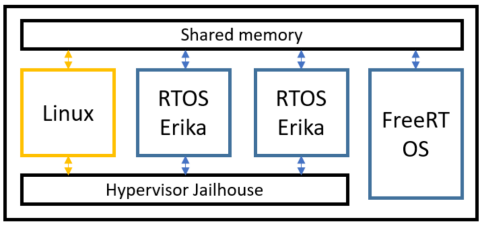


FIGURE 1.4 – Diagram of communications between KARlab entities

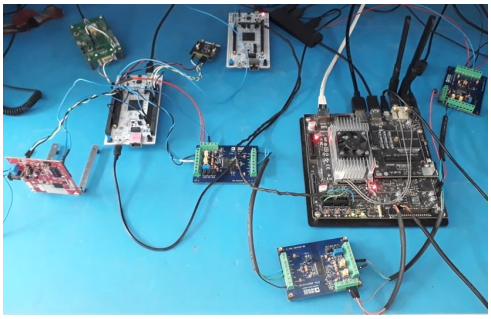
The technologies employed within the kart's onboard systems primarily function and communicate using the C and C++ programming languages. Before integration into the kart, the components responsible for supervision and artificial intelligence undergo initial development and testing phases using Python or C#, carried out within simulation environments. In addition, the servers and technologies associated with the Human Machine Interface (HMI) are constructed using React and JavaScript programming languages.

1.2.2 Onboard systems on the kart

For this project, a conventional electric kart called the "Sodikart," is the foundational platform. It is augmented by the integration of an onboard electronic card known as the "Nvidia Jetson TX2," effectively functioning as the Electronic Control Unit (ECU) mentioned previously. The Nvidia Jetson TX2 card assumes the role of a central processing unit, processing incoming information and managing critical events within the system. During the project's development phase, two distinct operating systems are employed concurrently : a Real-Time Operating System (RTOS), referred to as "Erika," and a conventional Rich Operating System (Linux). The RTOS is responsible for overseeing critical decision-making processes upon which the kart's integrity and driver's safety heavily rely, such as emergency braking procedures. It transmits directives to the mechanical actuators, as indicated in Figure 1.3. The RTOS is designed to swiftly make critical decisions and operate as close to real-time as possible. In contrast, the Li-



(a) Diagram of the ECU architecture



(b) Components on board the kart

nux system is responsible for handling non-critical data, particularly those related to the gaming aspects. It establishes connections with the game server, retrieves relevant data, and stores it in the kernel's memory space. The RTOS leverages this data to make informed decisions. The FreeRTOS bridges the Central Processing Units (CPUs) to the Controller Area Network (CAN) bus.

A hypervisor is pivotal between the operating systems and the hardware components. Its primary function is facilitating the concurrent operation of multiple operating systems on a single physical machine. The hypervisor assigns specific hardware resources to each operating system ; for instance, it allocates access to a WiFi device to the RTOS component. Furthermore, the hypervisor efficiently allocates and manages the computing power on the card among the coexisting operating systems, optimizing resource utilization within the kart's onboard systems. The operating systems engage in intercommunication through a shared memory space.

The second electronic card embedded in the kart is the Environmental Interface Card (EIC). This card is a hub for all the sensors integrated within the kart, including radar, GPS, and the Inertial Measurement Unit (IMU). The EIC performs the critical functions of collecting, organizing, and transmitting environmental data to the ECU. The ECU undertakes the associated data processing tasks, encompassing the determination of the kart's position and speed, hazard detection, data fusion, and the implementation of an extended Kalman filter. Furthermore, the ECU assumes control over the mechanical actuators, ensuring their operation following the processed data.

1.2.3 Use cases

The KarLAB project is characterized by a cohesive network of interconnected use cases, each meticulously outlined and contributing to the holistic research framework. This research comprises a multidimensional exploration, with the individual components elucidated as follows :

1.2.3.1 Localization : In this operational mode, the kart is designed to identify potential safety risks effectively. The system requirement in this mode is achieving precise real-time localization of the kart using cost-effective sensors. This critical capability enables the kart to adjust its behavior, encompassing speed reduction and the initiation of emergency braking, in response to identified safety concerns.

1.2.3.2 Automatic Parking Mode : In the second use case, Automatic Parking Mode, where important attention is given to safety and the autonomous parking process, specific requirements have been established to fulfill the project's objectives.

The system should be capable of effectively avoiding both static and dynamic obstacles within the scene, ensuring a safe parking operation. Additionally, adherence to particular aspects of the ISO-16787 :2017 standard, including the inclination angle and the deviation, must be maintained to align with industry regulations and safety protocols.

ISO 16787 :2017 [4] is a standard that specifically focuses on the Assisted Parking System (APS) for light-duty vehicles, such as passenger cars, pick-up trucks, light vans, and sport utility vehicles (excluding motorcycles). It outlines the minimum functionality requirements that drivers can expect from such systems, including detecting suitable parking spaces, calculating trajectories, and controlling the lateral movement of the vehicle during parking. Additionally, it covers aspects like providing information about obstacles in the vehicle's path and setting requirements for failure indication and performance testing procedures.

The system should establish a well-defined action space for the agent responsible for guiding the vehicle's parking maneuvers, including options such as stop, accelerate, brake, and orientation. This breadth of available actions ensures adaptability and the ability to respond effectively to various parking scenarios.

Moreover, the agent should be able to dynamically plan and execute the parking process, irrespective of the initial distance between the vehicle and the designated parking spot. This dynamic adaptability guarantees successful parking under varying conditions, emphasizing safety and operational efficiency throughout the procedure.

1.2.3.3 Urban Driving Mode : The final use case addresses the complexities of urban driving, involving interactions at junctions, traffic lights, and pedestrian crossings amidst multi-lane traffic. Notably, the system's performance in this mode must meet a requirement for fast convergence and the ability to react effectively in previously unseen situations. The project emphasizes the utilization of end-to-end systems, as discussed in the relevant literature (e.g., [Codevilla, 2019]), where driving policies are learned from data rather than relying on manually created rules. This approach ensures adaptability and safety while navigating the complexities of urban driving.

1.3 Research Questions and Hypotheses

In the pursuit of advancing the safety and reliability of self-driving cars, our research endeavors revolve around a set of fundamental and probing questions. These research inquiries are not only the cornerstone of our investigation but also serve as the guiding stars illuminating our path toward safer autonomous transportation :

— Research Question 1 : How can we attain high-precision absolute localization of AV by utilizing cost-effective sensors such as GPS and IMU ?

4. https ://www.iso.org/standard/73768.html

— Research Question 2 : Which approach or methodology can effectively ensure the safety and efficient execution of autonomous parking operations, including planning and parking the car while avoiding static and dynamic obstacles and respecting specific criteria outlined in the ISO-16787 :2017 standard, such as the inclination angle and deviation ?

— Research Question 3 : What is the most suitable algorithm for self-driving vehicles to facilitate rapid convergence and responsive actions in unforeseen and dynamic situations, particularly within urban driving scenarios ?

1.4 Thesis contributions

To tackle these questions, the thesis contributes threefold : a new data fusion algorithm for AV localization, a dynamic adjustment of a Deep Reinforcement Learning reward function, and a new framework for safe urban driving.

1.4.1 Data fusion algorithm for AV localization

The quest for high-precision AV localization has become central to enhancing safety and accident prevention. Multi-sensor data fusion is the practice of integrating data gathered from multiple sensor devices. This integration improves reliability, reduces uncertainty, and augments performance accuracy in real-time [Bounini, 2016]. Such techniques find applications in sensor networks, robotics, image processing, and self-driving vehicles [Khaleghi, 2013]. The inherent complexities of real-world scenarios often necessitate data fusion for optimal results [Osório, 2019]. Various fusion methods, including Kalman Filters (KF), Extended Kalman Filters (EKF), and Error State Extended Kalman Filters (ES-EKF), have been employed, alongside probability estimation, machine learning, and neural networks [Nweke, 2019]. Despite these approaches, challenges persist. Some techniques are constrained to indoor robotics, while others rely on expensive sensors, ill-suited for embedded systems [DAlfonso, 2015][Shaukat, 2021]. Simulators like Carla provide ideal, noise-free data, which may not mirror real-world conditions [Castillo-Torres, 2021].

To address these limitations, in Chapter 4, we present a solution for achieving high-precision localization in AVs by utilizing cost-effective sensors, specifically GPS and IMU. Our approach includes calibration of the IMU's physical sensor and the implementation of a low-pass filter to reduce IMU noise, followed by the integration of IMU sensor data with GPS coordinates using ES-EKF. The study highlights the superior performance of ES-EKF compared to EKF in fusing GPS and IMU data, as evidenced by improved localization results within the Carla simulator. Furthermore, we validate our approach through real-time urban driving scenarios, deploying the ICM 20600 IMU and Quectel L80-M39 GPS in a FreeRTOS real-time environment on the STM32 Nucleo platform. This validation demonstrates a 92% accuracy, calculated by the proportion of data points where the Error State Extended Kalman Filter (ES-KF) estimated position (X, Y) falls within a 0.5 meter distance threshold of the corresponding ground truth position. Additionally, our approach exhibits an exceptional swift execution time of around 20 μs, rendering it highly suitable for practical industrial applications.

1.4.2 Dynamic Adjustment of Reward Function for Proximal Policy Optimization with Imitation Learning

The second contribution is in the context of the second use case, Automatic Parking Mode, where the primary focus is on ensuring safety and the efficient execution of autonomous parking operations. The project has outlined specific requirements designed to address the key objectives of this use case. It is essential to note that while various methods have been explored in the domain of autonomous parking systems (APS), such as geometric [Fraichard, 2004],[Gómez-Bravo, 2001],[Lini, 2011], sampling [Han, 2011],[Zheng, 2018], and numerical optimization techniques for path planning [Thrun, 2006],[He, 2019],[Park, 2018], as well as Ackerman steering models [Weinstein, 2010] for path tracking, many of these methods have faced challenges related to the non-linearity of vehicle dynamics. These challenges have led to deviations from the intended route, and in some instances, they need to fully adhere to the European standard BS ISO 16787-2017 for Intelligent Transport Systems. The state-of-the-art literature still needs to comprehensively address the specific research question posed in this context, leaving room for further exploration and developing a suitable methodology to meet the defined requirements while focusing on safety and efficiency.

In light of these challenges, there is a growing interest in harnessing recent advancements in machine learning to enhance the current state-of-the-art APS techniques. Specifically, reinforcement learning has emerged as a promising research area in which agents learn to make intelligent decisions by interacting with their environment, taking actions, and receiving rewards. Deep Reinforcement Learning (DRL), which combines deep learning with reinforcement learning, has shown remarkable success in various domains, including Atari video games, Go, and robot manipulation. Several recent works have explored the application of DRL in APS, achieving notable results. However, many of these approaches have focused on static obstacles within the environment, neglecting dynamic obstacles and the critical aspect of braking constraints.

Despite the progress in DRL, it faces challenges when operating in real-world scenarios compared to controlled video game environments. This is due to the complexity of formulating reward functions for APS tasks, which can be complex and challenging. In particular, the ability to ensure safe and efficient parking maneuvers through DRL remains a significant research question.

In response to these challenges, chapter 5 presents a series of contributions and results aimed at enhancing APS by addressing the specific requirements and ensuring the safety and efficiency of autonomous parking operations. These contributions include utilizing a DRL algorithm, Proximal Policy Optimization (PPO) [Schulman, 2017] for parking spot exploration, planning, and execution, with the capability to handle static and dynamic obstacles and adapt vehicle speed as needed. The DRL algorithm is improved through dynamic reward adjustments and combined with imitation learning to reduce training time. Additionally, a task-specific curriculum learning approach is developed to train the agent effectively in complex environments.

Experimental results validate the model's, with a 90% success rate in executing precise control commands during parking maneuvers. This success rate reflects instances where the vehicle successfully planned, navigated, and parked the car while avoiding static and dynamic

obstacles. Moreover, 97% of these successful parking actions align with parking standards, featuring inclination angles greater than $\pm 0{,}2^{\circ}$ and deviations of less than 0.1 meters, reflecting the precision of the vehicle's positioning within the parking space. These findings underscore the model's ability to meet rigorous parking requirements while ensuring safety and operational efficiency.

1.4.3 Guided Hierarchical Reinforcement Learning for Safe Urban Driving

Currently, most AV systems employ hand-crafted modular architectures [Müller, 2018]. Though widely used, these architectures exhibit limitations in highly interactive environments, such as urban driving, due to their tightly interconnected nature. In contrast, end-to-end architectures have emerged as a promising alternative where driving policies are learned and adapted without manual intervention [Xu, 2017]. This approach facilitates continuous tuning of driving policies, aiming for human-level performance. However, the quest for a safe end-to-end driving policy remains challenging, particularly in complex urban driving scenarios.

End-to-end AV driving policies fall into three main categories : rule-based methods [Furda, 2011], imitation learning (IL) [Bansal, 2018][Xu, 2017][Chen, 2019a], and reinforcement learning (RL) [Wolf, 2017]. Rule-based methods rely on predefined rules to determine driving actions, but their development can be demanding due to the need for comprehensive rule sets [Furda, 2011]. IL-based methods offer an attractive alternative by directly learning driving policies from expert demonstrations. However, they require substantial amounts of labeled training data, which can be resource-intensive.

To address data challenges, DRL has been applied in more straightforward driving scenarios like lane-keeping and steering control [Kendall, 2019]. Yet, ensuring the safety of autonomous driving, especially during the exploratory phase in complex urban settings, remains a concern [Gu, 2022]. DRL often demands extensive training data, limiting its real-world applicability [Sutton, 2018]. Strategies for enhancing DRL safety involve complex cost functions and shielding techniques [Achiam, 2017] [Xu, 2018] [Alshiekh, 2018], but they can add computational complexity and hinder the exploration of optimal policies [Yang, 2023].

Recent research has explored Hierarchical Reinforcement Learning (HRL) as a promising approach for urban driving scenarios [Bronstein, 2022]. HRL decomposes complex tasks into smaller, more manageable sub-tasks with simplified state spaces, potentially reducing the need for extensive exploration [Guo, 2021]. Additionally, combining IL with HRL can address cold start issues and improve agent performance by modeling expert demonstrations as humanlike reasoning with symbolic rules [Cropper, 2022]. Although promising, it's essential to consider potential safety issues and limitations, particularly exploring unsafe actions and generalizing new situations [Chen, 2019a].

The state-of-the-art provides a comprehensive overview of different methods used in AV driving policy development, including rule-based methods, IL, and RL. While it highlights the challenges and complexities associated with these methods, it needs to directly respond to the research question regarding the most suitable algorithm for achieving rapid convergence and responsive actions in unforeseen and dynamic urban driving scenarios. It mainly focuses on

these approaches' safety and training challenges, which, while necessary, do not directly address the specific algorithmic suitability for urban driving, as emphasized in Research Question 3.

Chapter 5 presents Guided Hierarchical Reinforcement Learning (GHRL) for end-to-end urban driving, incorporating vision and localization data and expert demonstrations expressed in the Answer Set Programming (ASP) rules to guide the PPO exploration policy. GHRL-ASP offers several advantages over alternative methodologies, simplifying the calculation of complex loss functions and avoiding the complexity associated with shielding techniques. It ensures a real-time and efficient decision-making process. Experimental evaluations conducted in urban driving scenarios using the Carla simulator demonstrate the approach's effectiveness in handling challenges such as traffic lights static and dynamic obstacles by outperforming the state-of-the-art methods on Carla's NoCrash benchmark by 20%. Particularly in hazardous situations, the system relies on ASP rules to ensure safety and adherence to predefined constraints, mitigating potential harm to pedestrians. The system combines learned policies and rule-based systems triggered by specific events, showcasing its adaptability and robustness in urban driving scenarios [Brewka, 2011].

1.5 Thesis Organization

This manuscript presents the research carried out throughout my thesis and is organized into seven chapters :

Chapter 2, titled *Preliminaries*, provides an essential foundation for exploring various Artificial Intelligence (AI) and Machine Learning (ML) concepts. After an introduction, the chapter explores the core of neural networks and deep learning (Section 2) paradigms. Then, the chapter progresses to cover other critical aspects of AI and ML, including *Supervised Learning* (Section 3) and *Reinforcement Learning Algorithm* (Section 4). Section 4, in particular, provides a detailed overview of the basic principle of reinforcement learning. The chapter further discusses the role of sensors (Section 5) in AI and ML applications, including various sensor types such as cameras, LiDAR, RADAR, and GPS/IMU. Finally, the chapter concludes with a discussion of *Answer Set Programming (ASP)* (Section 6), a fundamental methodology in AI, addressing topics like the ASP programming language and programming methodology and solving the frame problem with ASP. The chapter concludes by summarizing the preliminary knowledge required for the topics covered in this thesis.

Chapter 3, titled *State of the Art*, is a comprehensive exploration of the current landscape of autonomous systems and their associated technologies. The chapter begins with an *Introduction* (Section 1), providing context for the following topics. It then explores specific areas of interest, such as *Localization and Data Fusion* (Section 2), highlighting the importance of these aspects in autonomous systems. The chapter covers *Imitation Learning and Reinforcement Learning* (Section 3), including specialized subsections like *Autonomous Driving with Imitation Learning* and *Autonomous Driving with Reinforcement Learning*, detailing how these approaches are utilized in autonomous vehicle technology. It also discusses the *Automated Parking System* (Section 3.3), a critical application of autonomous systems. *Safety Challenges in Autonomous Systems* (Section 4) is also addressed, considering methods like *Rule-based Methods* and the importance

of safety in reinforcement learning. The chapter then explores the tools available for simulation environments (Section 5), encompassing games, indoor environments, and driving scenarios. Finally, *Synthesis and Research Trails* (Section 6) provides a forward-looking perspective on research and future directions. This chapter serves as a valuable resource for understanding the latest advancements and challenges in autonomous systems and lays the groundwork for further exploration of this dynamic field.

Chapter 4, titled *Localization of Autonomous Vehicle with low-cost sensors* introduces the first contribution. It commences with an Introduction, providing an overview of absolute localization for autonomous vehicles by integrating two low-cost sensors, namely *GPS* and *IMU*. Subsequently, Section 2, titled *System Overview*, is presented. This section provides in-depth information regarding key components and processes, such as *IMU* calibration, the application of low-pass filters, and the *ES-EKF* methodology. This chapter then proceeds to Section 3, which extensively explores *Experiments and Results*. It is here that various experiments are conducted and their outcomes analyzed. The first subsection within Section 3 is dedicated to experiments carried out in the *Carla* simulation environment, with detailed descriptions of the simulation environment and a comprehensive breakdown of the simulation results. Following the simulation experiments, the focus shifts to *Real Test* scenarios in subsection 3.2. This encompasses descriptions of the real test environment, *IMU* calibration, low-pass filtering procedures, and, significantly, the data fusion results obtained. Finally, the chapter concludes (section 4) with a summarizing section, encapsulating the core findings and insights.

Chapter 5, titled *Dynamic Adjustment of Reward Function for PPO with IL : Application to Automated Parking Systems*, describes the second contribution. It provides a detailed examination of a specific application of reinforcement learning in the context of automated parking systems. After identifying the challenges and issues this research aims to address, the core of the chapter revolves around the *DPPO-IL* algorithm by introducing various components like *Curiosity Reward* and *Imitation Learning* with subcategories such as *Behavior Cloning* and *Generative Adversarial Imitation Learning*. The chapter then presents the *DPPO-IL Algorithm* and emphasizes *Curriculum Learning* as a crucial element in the approach. *Experiments and Results* (Section 4) describe the test setup, the agent, the environment, and the outcomes obtained through various experiments. Finally, the chapter concludes (Section 5) by summarizing its findings and contributions to automated parking systems and reinforcement learning. This chapter serves as a valuable resource for understanding how reinforcement learning techniques, combined with the dynamic adjustment of reward functions, can be applied to enhance the performance of automated parking systems.

Chapter 6, titled *Guided Hierarchical Reinforcement Learning for Safe Urban Driving*, is dedicated to the third contribution. It delves into applying hierarchical reinforcement learning techniques to ensure safe driving in urban environments. The core of the chapter focuses on GHRL, a model-free on-policy RL algorithm. The GHRL learning process is further broken down into *Low-Level Policies*, *High-Level Policies*, *Safety Specification*, *Longitudinal Critical Situations*, and *Lateral Critical Situations*. The chapter then reports on the *Experiments* (Section 3), providing insights into the setup of the environment and the evaluation metrics used.

It presents the results obtained in various scenarios, including the *NoCrash Benchmark*, *Obstacle Avoidance*, training *Sub-Policies*, and conducting experiments with *Safe High-Policies*. The chapter concludes (Section 4) by summarizing its findings and contributions to safe urban driving through guided hierarchical reinforcement learning techniques. It is a valuable resource for understanding how reinforcement learning can enhance safety in urban driving scenarios.

Chapter 7, titled *Conclusion and Perspectives*, serves as the final chapter of this thesis. In this chapter, we summarize the key findings and contributions of the research. Then we explore potential avenues for future research and development. This section can act as a bridge for further investigations in the area of autonomous vehicles.

Chapitre 2

Preliminaries

Contents

1 Introduction

This chapter provides an in-depth exploration of deep learning in the context of autonomous vehicles, focusing on the central role of neural networks. The target audience includes individuals with a foundational understanding of machine learning and statistics, offering a comprehensive overview of neural networks and their critical significance in shaping the trajectory of autonomous driving. Neural networks, inspired by the structure and functionality of the human brain, form the core of deep learning. Within autonomous vehicles, these networks are vital in enabling perception and navigation within real-world environments. Neural networks can discern complex patterns from sensor data, making them indispensable for tasks such as image recognition, natural language processing, and machine translation within the autonomous driving domain. Fundamental to neural networks are artificial neurons, elemental components responsible for processing sensor inputs and generating driving commands. These neurons establish interconnected layers, and their weighted connections are pivotal in shaping the network's learning and performance. This chapter examines how artificial neural networks are applied to improve key functionalities of autonomous vehicles. We elucidate how networks receive input data from diverse sensors, encompassing cameras, LiDAR, radar, and GPS. This data undergoes processing through hidden layers, ultimately yielding output that translates into critical driving decisions. For tasks of utmost importance to autonomous driving, such as object detection and lane following, we explore convolutional neural networks (CNNs), a specialized neural network variant tailored for image recognition. CNNs leverage convolutional layers to extract complex features from sensor data, achieving state-of-the-art performance in domains crucial to autonomous vehicles. Our exploration extends beyond architectural considerations to encompass the essential training process. Neural networks are trained to execute specific driving tasks, with backpropagation emerging as a pivotal algorithm for weight adjustments, minimizing a loss function indicative of the network's performance on training data. Supervised learning, a foundational concept, assumes prominence in our discussion, involving model training on labeled datasets containing sensor data and corresponding driving commands. This empowers autonomous vehicles to predict driving commands for novel sensor data by discerning patterns within labeled examples. Furthermore, we explore the domain of reinforcement learning, where autonomous vehicles acquire the ability to make informed driving decisions through interactions with their environment. These models receive rewards and penalties based on their actions, eventually mastering hard tasks like navigating dynamic traffic scenarios. This chapter places significant emphasis on sensors, which serve as the sensory apparatus of autonomous vehicles. Sensors measure physical attributes such as distance, speed, and object proximity. They are the core of data acquisition, facilitating informed decision-making and ensuring the safety and reliability of autonomous driving. In conclusion, we touch upon answer set programming, a declarative language that addresses challenges within autonomous driving, encompassing route planning, task scheduling, and fault diagnosis. This programming paradigm has been pivotal in developing machine learning models that address complex problems within the autonomous vehicle domain.

This chapter serves as an insightful journey into the profound realm of deep learning within

the context of autonomous vehicles. It is structured into subsections, each contributing to a holistic understanding of deep learning in the context of autonomous vehicles. It commences with a comprehensive introduction that sets the stage for the subsequent discussions. The chapter then explores the fundamentals of neural networks and deep learning, covering aspects such as artificial neurons, artificial neural networks, convolutional neural networks, and the critical training and backpropagation processes. It further explores supervised learning and reinforcement learning algorithms, encompassing their definitions, classifications, and function approximation techniques. The chapter thoroughly examines sensors, a critical element in autonomous driving.

The latter include cameras, LiDAR, RADAR, and GPS/IMU systems, with in-depth insights into their roles and applications in autonomous vehicle technology. The chapter's content is rich and varied, providing a comprehensive overview of deep learning and its application in the context of autonomous vehicles. It culminates with a detailed exploration of Answer Set Programming (ASP) and its application in addressing complex problems related to autonomous vehicle decision-making. In conclusion, this chapter is a valuable reference for anyone seeking to comprehend the landscape of deep learning in autonomous vehicles.

2 Neural Network and Deep Learning

2.1 Artificial Neuron

The concept of an artificial neuron was first introduced in 1943 in "A logical calculus of the ideas immanent in nervous activity" by [Rosenblatt, 1962]. Modeled after biological neurons, an artificial neuron is a mathematical entity that follows specific rules. When given an input of $x = [x_0, \ldots, x_n]$, the neuron produces an output of $y = \sigma\left(\sum w_i x_i + b_i\right)$, where w_i and b_i are the neuron's internal parameters, representing its weight and bias, respectively. The activation function σ may add non-linearity, and common activation functions include sigmoid ($\sigma(x) = \frac{1}{1+e^{-x}}$), hyperbolic tangent (tanh)$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, and rectified linear unit (ReLU) ($\sigma(x) = \max(0, x)$).

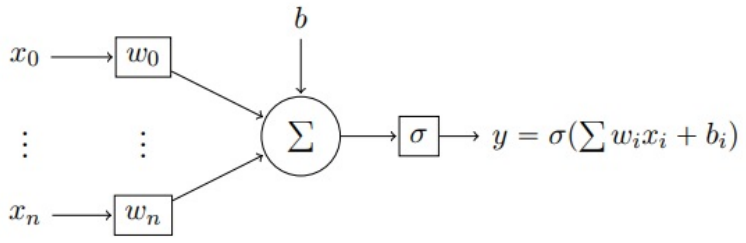


FIGURE 2.1 – Structure of an artificial neuron.

Initially, the first artificial neuron was a perceptron. It is a binary classifier whose activation function is $sign()$, and there is no bias

$$\text{output} = \begin{cases} 1 \text{ if } \sum w_i x_i > 0 \\ 0 \text{ otherwise} \end{cases} \tag{2.1}$$

2.2 Artificial Neural Network

Due to its construction, the perceptron can only classify data separated by a single line, making it unsuitable for modeling complex functions like the XOR function. Multiple perceptrons are needed to model the function, leading to the development of a Multi-Layer Perceptron (MLP). It is important to note that while an MLP consists of several perceptrons with binary activation functions, an artificial neural network (ANN) can have various activation functions for its neurons. Figure 2.2 illustrates a multi-layer ANN with neurons represented as circles, typically organized into layers, where the outputs of one layer become the inputs of the next. Hidden layers refer to the internal layers, and the term "deep learning" is used when an ANN has more than two hidden layers.

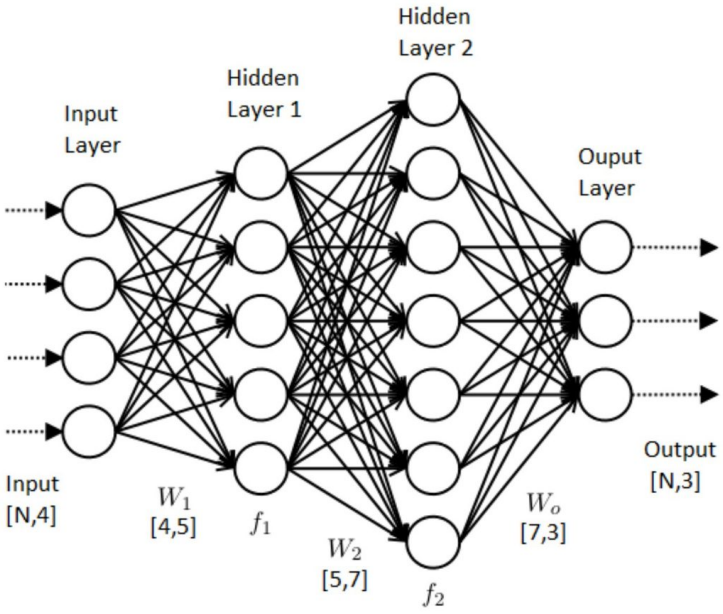


FIGURE 2.2 – Multi layer artificial Neural Network.

2.3 Convolutional Neural Network

Fully connected layers are organized so that each neuron is connected to every neuron of the previous layer. However, this design could be better for image processing as it would require many parameters due to the size of the images. For instance, RGB images of size 224×224, which are used in the ImageNet dataset, would require $150,528 (= 224 \times 224 \times 3)$ weights for each neuron of the first layer. To address this problem, Convolutional Neural Networks (CNNs) were introduced in 1998 by[LeCun, 1998] in their paper "Gradient-based learning applied to document recognition" to enable handwriting recognition (refer to Figure 2.3). CNNs consist of convolution layers that process the image by area instead of pixels to detect patterns, typically

followed by pooling layers to subsample the image. CNNs reduce the number of parameters and offer translation invariance as another advantage. Typically, CNNs include fully connected layers after convolutional and pooling layers. Classical computer vision architectures, such as VGG [Simonyan, 2014], Resnet [He, 2016], and Inception [Szegedy, 2015], are widely used. There is an ongoing active pursuit to develop new architectures that are either lighter or more powerful.

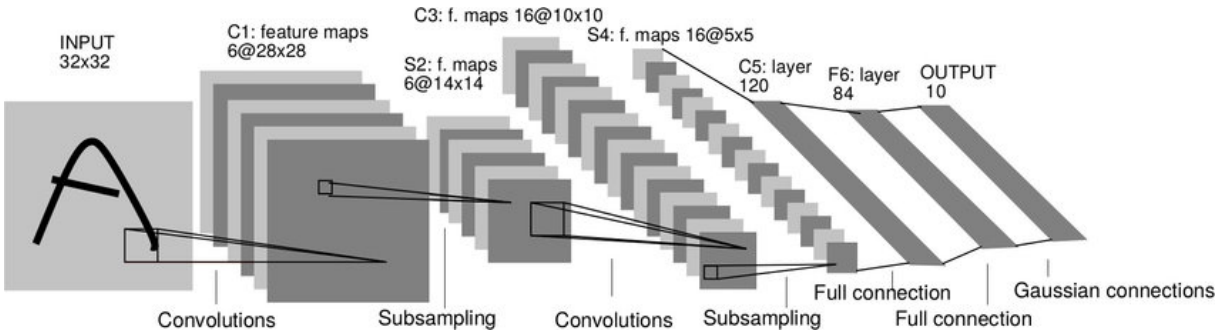


FIGURE 2.3 – Convolutional network for handwriting recognition [LeCun, 1998]

2.4 Training and Backpropagation

Training an artificial neural network involves determining the optimal parameters θ - which comprise the weights and biases of the network - for a specific task. To accomplish this, a loss function $\mathcal{L}$ is defined, with the form of the function depending on the type of learning and the problem at hand. Specific examples of loss functions in supervised or reinforcement learning can be found in the subsequent sections. The network's parameters are then adjusted based on the derivative of the loss function, with each $\theta_i \in \theta$ being modified in accordance.

$$\Delta\theta_i = \frac{\partial \mathcal{L}}{\partial \theta_i} \tag{2.2}$$

The gradient descent rule is used to update the parameter θ_i. When the goal is to maximize the loss, this is referred to as gradient ascent.

$$\theta_i \leftarrow \theta_i - \alpha\Delta\theta_i \tag{2.3}$$

The gradient descent rule utilizes the learning rate α, and the commonly used optimizers include Stochastic Gradient Descent with momentum (SGD) [Qian, 1999] and the Adam optimizer [Kingma, 2014]. Machine learning has three main learning frameworks : supervised, reinforcement, and unsupervised. Supervised learning involves learning from annotated data. Reinforcement learning utilizes a reward function ; unsupervised learning deals with unlabeled data. This thesis focuses on the first two frameworks, supervised and reinforcement learning, which will be discussed in the following sections.

3 Supervised Learning

Supervised learning is the most widely used type of machine learning, where the model is trained on labeled data. This approach is frequently employed in computer vision for object detection, image classification, depth prediction, and semantic segmentation tasks. The labeled data consists of pairs (x_n, y_n), where $n \in \mathbb{N}$, x_i represents the input (e.g., an image), and y_i represents the corresponding desired output (e.g., the object's type in the image). The model can be represented by a function f with parameters θ representing the neural network's weights and biases. The objective is to learn these parameters, such that $f(x_i, \theta) = y_i$ for each pair in the dataset. The L2 loss or mean square error is a standard loss function used in supervised learning.

$$\mathcal{L} = MSE(f(x, \theta), y) = \frac{1}{N} \sum^{N} (f(x_i, \theta) - y_i)^2 \tag{2.4}$$

The cross-entropy or log loss (equation 2.5) is a commonly used loss function in classification problems. In this scenario, the model's output is $f(x, \theta) = p$, where p denotes the probability of the input belonging to a particular class, and the ground truth is represented by $y \in [0, 1]$. For multi-class classification, the loss is computed as the sum of losses across all classes (equation 2.6).

$$\mathcal{L} = CE(p, y) = -y \log(p) + (1 - y) \log(1 - p) \tag{2.5}$$

$$\mathcal{L} = -\sum_{i} y_i \log(p_i) \tag{2.6}$$

Supervised learning has several advantages : being relatively fast, stable, and typically demonstrating good convergence. However, it requires labeled data, which can be expensive to obtain. Additionally, datasets are limited in size and can be biased due to human factors. Supervised learning can be affected by a distributional shift in scenarios requiring a sequence of actions (e.g., autonomous driving). This means that the distribution of the training dataset is not identical to the distribution encountered during testing. In such cases, a single mistake can cause errors to accumulate, leading to the model being unable to recover from its mistakes.

4 Reinforcement Learning Algorithm

Reinforcement learning provides an alternative to supervised learning, often called "learning by trial and error." Unlike supervised learning, it does not require labeled data but rather a reward function that specifies desirable or undesirable behavior. This approach reduces the influence of external bias, but learning can take longer as the model does not have access to demonstrations of optimal behavior.

4.1 Reinforcement Learning principle

4.1.1 Definition

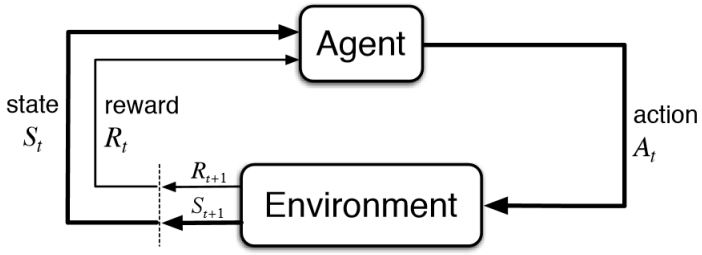


FIGURE 2.4 – Reinforcement learning illustration [Sutton, 2018]

In reinforcement learning, an agent interacts with an environment at each time step t. The environment generates a state s_t. Based on the current state s_t, the agent selects an action a_t according to a policy $\pi(a_t \mid s_t)$. The policy determines the probability of choosing action a_t in state s_t. In a deterministic environment, $\pi(s_t) = a_t$. After the agent takes action a_t, the environment provides a new state s_{t+1} and a reward r_t. The reward reflects the desirability of the new state. The agent aims to maximize the cumulative rewards by achieving the highest sum of rewards over time : $\max \sum r_t$. To avoid unbounded sums, the cumulative rewards are often discounted. The cumulative rewards, considering the discounting factor, are referred to as the return R_t. γ is the discount factor. It quantifies the significance of future rewards. It not only delineates the extent of importance attributed to future rewards but also aids in approximating the variability or uncertainty associated with such rewards. Gamma, ranging from 0 to 1. When Gamma approaches zero, the agent tends to prioritize immediate rewards over future ones. Conversely, as Gamma approaches one, the agent accords greater weight to future rewards, demonstrating a propensity to defer gratification in favor of potentially higher long-term gains.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.7}$$

We also define the V-function and Q-function as follows :

$$V_\pi(s) = \mathbb{E}_\pi \left[R_t \mid S_t = s \right] \tag{2.8}$$

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[R_t \mid S_t = s, A_t = a \right] \tag{2.9}$$

The V-function, or the value function, computes the return associated with a particular state. Conversely, the Q-function, or quality function, computes the expected return associated with a state-action pair. These functions evaluate the value or quality of a state (or state-action pair) under a given policy π. Furthermore, the advantage function can be defined as :

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{2.10}$$

The advantage defines how much better action a is compared to the action chosen by the policy π.

4.1.2 MDP and POMDP

In reinforcement learning, the agent makes decisions by selecting actions based on the received state. The state signal is assumed to possess the Markov property, which depends solely on the previous state. As a result, the transition probability can be expressed as a function of the previous state and action : $p(s_{t+1} \mid s_t, a_t)$. The reinforcement learning problem can be formulated as a Markov Decision Process (MDP), which is characterized by a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$:

— $\mathcal{S}$ is a set of states

— $\mathcal{A}$ is a set of actions

— $\mathcal{P}$ is a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$

— $\mathcal{R}$ is a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R} \to [0, 1]$ or in deterministic cases $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$

— γ is the discount factor

However, it may only sometimes be feasible to obtain the complete state with all the information from the environment at a given time step. In certain environments, only partial information is available. For example, in computer vision problems, there may be occlusions in the visual input. In such cases, the observation is defined as incomplete information derived from the state. This gives rise to the concept of a Partially Observable Markov Decision Process (POMDP), which is represented by a 7-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \Omega, \mathcal{O})$. In this tuple, $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$, and γ are defined as in an MDP, while Ω represents the set of possible observations. The function $\mathcal{O}$ is a conditional observation probability function that specifies the probability of observing o when the system is in state s, mapping from $\mathcal{S} \times \Omega$ to the interval $[0, 1]$. Addressing POMDPs can be approached in various ways. One approach involves inferring or estimating the underlying state from the observed information using a probability distribution known as a belief state. Another approach is to ignore the hidden state altogether. However, this approach can lead to challenges since different states can produce similar observations, causing ambiguity.

4.1.3 Model-free vs Model-based

Reinforcement learning algorithms can be classified into two main categories : model-free and model-based. Model-based algorithms aim to learn the underlying Markov Decision Process (MDP) by modeling the problem. By acquiring knowledge of the reward and transition functions, the problem can be solved through optimization techniques. On the other hand, model-free algorithms focus on directly optimizing the reward without explicitly building a model of the MDP. Examples of model-free algorithms include Q-learning and Policy Gradient. In the specific context of autonomous driving, learning a model of the environment in a high-dimensional space, such as images, is challenging. Additionally, it is impossible to know the future actions of other surrounding vehicles or actors. Therefore, in this thesis, our emphasis will be on model-free approaches, which do not rely on explicit modeling of the environment or the knowledge of the future actions of other agents.

4.1.4 Exploration vs Exploitation

Exploration versus exploitation is a well-known challenge in reinforcement learning, where exploitation refers to choosing the best option based on current knowledge. In contrast, exploration involves testing alternative options to gather new information. Achieving the right balance between these two aspects is critical for successful learning. It is generally advisable to prioritize exploration at the beginning of the learning process when little is known and gradually shift towards exploitation as the agent's performance improves. Two common approaches for the exploration-exploitation dilemma are ϵ-greedy and softmax. The ϵ-greedy approach selects the best action (i.e., the greedy action) with probability 1 - ϵ and a random action with probability ϵ. With the ϵ-greedy approach, all actions have an equal probability of being chosen. Commonly chosen values for ϵ range from 0.01 to 0.1. In contrast, the softmax approach considers the quality of each action Q(a,s) and selects the following action based on the Boltzmann distribution :

$$\pi(a, s) \sim \frac{e^{\frac{Q(a,s)}{T}}}{\sum_{a_i \in \mathcal{A}} e^{\frac{Q(a_i,s)}{T}}}, \tag{2.11}$$

Where T is a temperature parameter that controls the degree of exploration, when τ is high, actions with lower Q-values have a higher probability of being chosen, leading to more exploration. In contrast, low values of τ tend to favor exploiting the currently best-known actions.

Both ϵ-greedy and softmax approaches are designed for discrete action spaces. Still, in the case of continuous action spaces, exploration is often done by sampling random actions around the output of the policy model using a Gaussian probability distribution. This is known as Gaussian exploration. The exploration is done by adding a small amount of noise sampled from a Gaussian distribution to the policy output. The standard deviation of the Gaussian distribution is typically chosen as a hyperparameter and determines the amount of exploration. A higher standard deviation will lead to more exploration, while a lower standard deviation will lead to more exploitation of the current policy. Gaussian exploration is commonly used in deep reinforcement learning algorithms for continuous control tasks.

$$\pi(a, s) \sim \mathcal{N}(f(s), \sigma) \tag{2.12}$$

Where $f(s)$ is the model output, and σ is the standard deviation of the Gaussian.

4.1.5 On-Policy vs Off-Policy

In reinforcement learning, the learning process consists of two main phases : data collection through interaction with the environment and policy updates based on the collected data. On-policy algorithms collect data using the current policy, while off-policy algorithms use a different policy (e.g., an older or more greedy policy) to gather data. On-policy algorithms are more stable because they update the policy using the data collected by the same policy. However, they often have slower learning rates compared to off-policy algorithms. On the other hand, off-policy algorithms are sample-efficient, meaning that the collected data can be reused multiple times.

This improves the utilization of data. Furthermore, data collection and policy updates can be performed simultaneously in off-policy algorithms, accelerating the training process.

4.2 Classification of Reinforcement Learning Algorithms

Reinforcement learning algorithms can be classified into two main categories : tabular methods and function approximation. Tabular methods are designed for MDPs with a finite number of states and actions, while function approximation can be used for MDPs with an infinite number of states and actions. The following paragraph will briefly overview tabular methods before delving into function approximation.

4.2.1 Tabular Methods

In Reinforcement Learning, Tabular methods are a class of algorithms used when the MDP is finite, when the number of states and actions is limited, and can be represented in a table-like structure. Tabular methods usually store and update the values of state-action pairs or state-action-state transitions in a table called a Q-table or a V-table. These methods include dynamic programming, Monte Carlo methods, and TD-learning, which are used for value iteration, policy iteration, and Q-learning. Tabular methods work well for small-scale problems with a few states and actions but become computationally infeasible for large-scale problems.

4.2.2 Dynamic Programming

Dynamic programming is a technique used for planning and control in model-based RL, assuming perfect knowledge of the Markov Decision Process (MDP). Its main objective is to find the optimal policy of a finite MDP using the Policy Iteration algorithm. This algorithm consists of two parts : policy evaluation, where the value function V is computed for the current policy π, and policy improvement, where π is improved using the calculated value function V until the optimal policy π^* is found. However, this algorithm can be computationally expensive as policy and value are computed simultaneously. To overcome this, V-values can be directly computed, and the optimal policy can be derived from the optimal V-values using the following equation : $\pi \approx \pi *$ such that $\pi(s) = \text{argmax}_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right]$.

4.2.3 Monte-Carlo Methods

Dynamic programming relies on having prior knowledge of the environment, whereas Monte Carlo methods are model-free algorithms that learn from experience without assuming knowledge of the transition function. Monte Carlo methods learn from complete episodes and do not use bootstrapping, unlike TD (Temporal Difference) learning. These algorithms use mean values, such as $Q(s, a)$, representing the average return when starting from state s and taking action a. See Algorithm 1 for more details.

Algorithm 1 First-visit Monte-Carlo policy evaluation [Sutton, 2018]

Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}$:
$Q(s,a) \leftarrow$ arbitrary
Returns $(s,a) \leftarrow$ empty list for all $s \in \mathcal{S}$
$\pi(a \mid s) \leftarrow \epsilon$-greedy policy
for every update **do**
  Generate an episode using π
  for each state (s,a) appearing in the episode **do**
    $R \leftarrow$ return following the first occurrence of s
    Append R to Returns (s,a)
    $Q(s,a) \leftarrow$ average(Returns(s,a))
  end for
end for

4.2.4 Temporal Difference Learning (TD-learning)

In contrast to Monte Carlo methods, TD-learning algorithms do not need to wait until the end of the episode to estimate the value. Instead, they can update values during the episode faster than the Monte Carlo methods. Two major TD algorithms are SARSA (Algorithm 2) and Q-learning (Algorithm 3), both used for computing the Q-table. SARSA is an on-policy algorithm, while Q-learning is an off-policy algorithm.

Algorithm 2 SARSA [Sutton, 2018]

Initialize $Q(s,a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$ and Q(terminal state)
for each episode **do**
  Initialize s
  choose action a from s using policy derived from Q (e.g. ϵ-greedy)
  for each step of the episode **do**
    take action a, observe reward r and next state s'
    choose action a' from s' using policy derived from Q (e.g., ϵ-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q\left(s', a'\right) - Q(s,a)\right]$
    $s \leftarrow s'; a \leftarrow a'$
  end for
end for

The SARSA algorithm is named after the five elements $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ required for its implementation. In each step, the agent selects an action based on its current policy, occasionally choosing a random action for exploration. The agent receives a reward and transitions to a new state s'. Then, based on its current policy, the agent selects another action and updates the Q-table using the reward r, the new state s', and the new action a'. SARSA is an on-policy algorithm because the new action a' is chosen according to the current policy. In contrast, Q-learning is an off-policy variant of SARSA. In Q-learning, the agent does not have to select a second action a', and the update is performed using the Q-value of the best possible action ($\max_a Q(s,a)$) instead. This allows Q-learning to learn the optimal action-value function while following a different (e.g., greedy) policy for exploration.

As SARSA is on-policy, its convergence properties depend on the policy used, whereas this

Algorithm 3 Q-learning [Sutton, 2018]

Initialize $Q(s, a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$, and $Q($ terminal state, $\cdot)$
for each episode **do**
  Initialize s
  for each step of the episode **do**
    choose action a from s using policy derived from Q (e.g.ϵ-greedy)
    take action a, observe reward r and next state s'
    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_a Q\left(s', a\right) - Q(s, a)\right]$
    $s \leftarrow s'$ until s is terminal
  end for
end for

is not the case for Q-learning, which is independent of the policy being followed.

4.3 Function Approximation

The previous section presented three classes of algorithms that assumed a finite MDP, which can be limiting for ongoing or high-dimensional problems. This section will explore function approximation algorithms, which constitute the second class of reinforcement learning algorithms. Unlike tabular methods, these algorithms may not converge to the optimal solution since they operate in an infinite-dimensional space. Richard Bellman coined the term 'curse of dimensionality' in 1961 to describe the difficulties of dealing with large amounts of data. Function approximation algorithms can be categorized into three classes. The first class is value-based algorithms, which infer Q for each state-action pair and derive the optimal policy from it. While the state space can be vast in function approximation, value-based algorithms still require a finite action space. The second class is policy-based algorithms, which compute the policy directly, allowing for large or infinite action spaces with no intermediate steps. However, a third class of algorithms, actor-critics, is more stable and preferred over policy-based algorithms. Actor-critics simultaneously calculate the policy and state value.

4.3.1 Value-based Algorithms

Value-based algorithms rely on computing the value of each state and, more specifically, the Q-value for each state-action pair. As discussed in the previous paragraph, algorithms such as Q-learning and SARSA utilize the Q function to obtain the optimal policy $\pi^*(s) = \operatorname{argmax} Q(a, s)$. For problems with nonfinite state spaces, Deep Q learning is the most well-known value-based algorithm. This algorithm is designed to compute the Q-value in such cases. In this section, we will introduce Deep Q learning and some of its variations.

4.3.1.1 Deep Q-learning [Mnih, 2013] first introduced Deep Q-learning in their paper "Playing Atari with Deep Reinforcement Learning", which utilized Q-learning with image inputs. Traditional tabular methods struggle with high dimensional state spaces, which motivated the use of a convolutional neural network to represent the Q-function in Deep Q-learning.

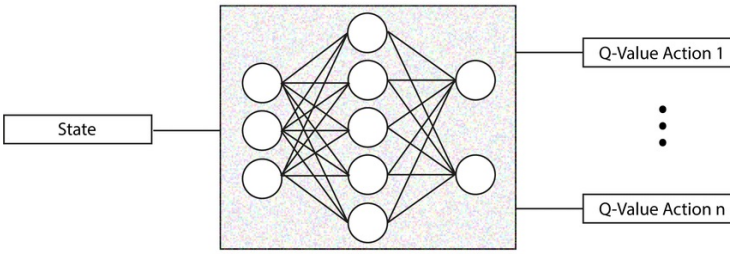


FIGURE 2.5 – Deep Q learning illustration

This deep neural network, hence the name Deep Q Learning, takes preprocessed images (e.g., current state and previous states) as inputs and outputs the predicted Q-values for every possible action. The neural network weights, denoted as θ, represent the Q parameter in DQN. The loss function that is optimized in DQN is :

$$\text{Loss} = \mathbb{E}\left[\left(y_i^{DQN} - Q(s, a, \theta)\right)^2\right] \tag{2.13}$$

with

$$y_i^{DQN} = r_i + \gamma \max_{a'} Q\left(s_{i+1}, a' \mid \theta\right) \tag{2.14}$$

However, the assumption of independent and identically distributed data, which many learning algorithms rely on, is invalid in reinforcement learning due to states' sequential and correlated nature. To address this problem, [Mnih, 2013] introduced experience replay, first proposed in 1993 and later formalized and tested by other researchers, such as [Kingma, 2014], in their work on real-time reinforcement learning control. Experience replay involves storing samples in a buffer during training and performing model updates, such as neural network weight updates, with randomly sampled mini-batches from the buffer. Deep Q-learning, detailed in Algorithm 4, achieved state-of-the-art results on seven Atari games.

4.3.1.2 Improvements and Variants In 2015, [Mnih, 2015] made further advancements to Deep Q-learning by successfully applying it to all 49 available Atari games, as detailed in their paper "Human-level control through deep reinforcement learning" [Mnih, 2015]. One essential improvement was the introduction of a target network, which significantly enhanced the algorithm's stability. The concept of Double Q-learning, initially proposed by Hasselt in 2010 [Hasselt, 2010] for standard Q-learning, was later adapted for Deep Q-learning in the paper "Deep Reinforcement Learning with Double Q-Learning" [Hasselt, 2016]. In Double Q-learning, a target network Q with parameter θ_1 is employed to compute the target y_i in Equation 2.16, thereby reducing the instability caused by y_i being dependent on Q. This approach also helps alleviate the issue of value overestimation that arises from the maximum operation in 2.14. To ensure training stability, the target network Q is fixed for several iterations.

Algorithm 4 Deep Q-learning with Experience Replay [Mnih, 2013]

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialize sequence $s_1 = x_1$ and preprocessed sequences $\phi_1 = \phi(s_1)$
  for t=1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    $y_i = \begin{cases} r_j & \text{for terminal } \theta_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non terminal } \theta_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))$ according to equation 2.13
  end for
end for

$$y_i^{DDQN} = r_i + \gamma \max_{a'} Q(s_{i+1}, a' \mid \theta') \tag{2.15}$$

The update frequency of the target network's parameters is every N step, achieved by setting $\theta' \leftarrow \theta$. In [Hasselt, 2016], DQN and DDQN (Double Deep Q Network) were compared on six Atari games. DDQN exhibited better performance and stability than DQN, with a significantly smaller Q value estimation overshoot. DDQN is considered a more stable and effective alternative to DQN. [Osband, 2016] proposed Deep Exploration via Bootstrapped DQN, where Q functions are sampled over distribution at the start of an episode to address the exploration issue in reinforcement learning. This allows for deep exploration and consistency within a single episode. Dueling Network Architectures for Deep Reinforcement Learning" by [Wang, 2016c] introduces an alternative approach by proposing a new network architecture that utilizes two streams : one for estimating the state value function (V function) and another for estimating the advantage function. By combining these two streams, the Q-values can be computed as $Q(s,a) = V(s) + A(s,a)$. This architecture explicitly computes the value and the advantage, allowing the network to learn which actions are significant in particular states. Compared to a single-stream architecture, the dueling Q network converges faster and performs better, as shown by [Wang, 2016c] on 57 Atari games. Their dueling architecture has a mean score over the human performance of 591.9%, while DQN from [Mnih, 2015] has 341.2%. Other improvements to the DQN algorithm include Prioritized Experience Replay [Schaul, 2015], where transitions are given probabilities based on their TD error $\delta = r_{t+1} + \gamma \max a' Q\text{target}\ (s_{t+1}, a') - Q(s_t, a_t)$, and Hindsight Experience Replay (HER) [Andrychowicz, 2017] proposed replays unsuccessful episodes by changing the initial goal to the state the agent reached. This method addresses sparse rewards in environments where exploration may not be sufficient to reach the goal. HER stores the goal as a variable of the transition. The combination of DDPG and HER in the Mujoco simulator [Todorov, 2012] achieved high success rates in three tasks with sparse rewards : pushing a box,

sliding a puck, or picking and placing a box. In Rainbow : Combining Improvements in Deep Reinforcement Learning [Hessel, 2018], a team from DeepMind combines multiple improvements to DQN, such as double Q learning [Hasselt, 2010] and experience replay [Schaul, 2015], to assess their complementarity.

4.3.2 Policy Gradient

Deep Q learning is a practical algorithm for discrete action spaces, but it faces the curse of dimensionality when applied to continuous actions. Discretization of continuous actions can lead to many possible actions, making the Q function output dimension grow exponentially with the number of action dimensions. Policy Gradient is a RL algorithm that optimizes the policy and is effective in high-dimensional or continuous action spaces. It can deal with stochastic policies and has good theoretical convergence properties. In Policy Gradient methods, the policy is represented by a parameter vector θ, which usually corresponds to the weights and biases of a neural network. This policy takes a state s as input. When dealing with discrete action spaces, the policy network produces a probability vector representing each possible action's likelihood. On the other hand, in continuous action spaces, the output is the mean (and sometimes variance) of a Gaussian distribution. The distinctive aspect of Policy Gradient methods is that they directly calculate the policy itself instead of first estimating the quality of a state and then constructing a policy that guides toward favorable states, as done in Q-learning and Value iteration algorithms.

The first policy gradient algorithm described in detail in Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning by Williams (2012) is known as the REINFORCE algorithm. Although developed in 2012, various enhancements have been made to improve its efficiency. The fundamental concept behind REINFORCE (outlined in Algorithm 5) is to optimize the expected sum of rewards for a trajectory $\tau : \mathbb{E}_\pi[R(\tau)]$ by performing gradient ascent on the policy parameter θ. Unlike minimizing a loss, as we want to maximize the reward, gradient ascent is used. We recommend that readers consult the lectures of John Schulman [Schulman, 2017] and David Silver [Belousov, 2021] for more information on this topic.

$$\theta = \theta + \nabla_\theta \mathbb{E}_\pi[R(\tau)] \tag{2.16}$$

Later we will use the notation $J(\theta) = \mathbb{E}_\pi[R(\tau)]$, so the previous equation becomes :

$$\theta = \theta + \nabla_\theta J(\theta) \tag{2.17}$$

The gradient $\nabla_\theta J(\theta)$ can be derived as follows.

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_\pi[R(\tau)] \\
&= \nabla_\theta \int p(\tau \mid \theta) R(\tau) d\tau \\
&= \int \nabla_\theta p(\tau \mid \theta) R(\tau) d\tau \\
&= \int p(\tau \mid \theta) \frac{\nabla_\theta p(\tau \mid \theta)}{p(\tau \mid \theta)} R(\tau) d\tau \\
&= \int p(\tau \mid \theta) \nabla_\theta \log p(\tau \mid \theta) R(\tau) d\tau \\
&= \mathbb{E}_\pi \left[\nabla_\theta \log p(\tau \mid \theta) R(\tau) \right] \qquad (2.18)
\end{aligned}$$

Using the fact that the gradient of the log probability of $p(\tau \mid \theta)$ is :

$$p(\tau \mid \theta) = p(s_0) \prod_{t=0}^{T-1} \left[\pi(a_t \mid s_t, \theta) P(s_{t+1}, r_t \mid s_t, a_t) \right] \qquad (2.19)$$

where $P(s_{t+1}, r_t \mid s_t, a_t)$ is the transition probability and $p(s_0)$ is the initial state s_0 probability. The gradient of the log probability of $p()$ is :

$$\log p(\tau \mid \theta) = \log p(s_0) + \sum_{t=0}^{T-1} \left[\log \pi(a_t \mid s_t, \theta) + \log P(s_{t+1}, r_t \mid s_t, a_t) \right]. \qquad (2.20)$$

When we differentiate by θ as $\log p(s_0)$ and $\log P(s_{t+1}, r_t \mid s_t, a_t)$ do not depend on θ, we obtain :

$$\nabla_\theta \log p(\tau \mid \theta) = \nabla_\theta \sum_{t=0}^{T-1} \log \pi(a_t \mid s_t, \theta) \qquad (2.21)$$

So :

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_\pi[R(\tau)] \\
&= \mathbb{E}_\pi \left[R(\tau) \nabla_\theta \log p(\tau \mid \theta) \right] \\
&= \mathbb{E}_\pi \left[R(\tau) \nabla_\theta \sum_{t=0}^{T} \log \pi(a_t \mid s_t, \theta) \right] \\
&= \mathbb{E}_\pi \left[\left(\sum_{t=0}^{T} r(s_t, a_t) \right) \left(\nabla_\theta \sum_{t=0}^{T} \log \pi(a_t \mid s_t, \theta) \right) \right] \qquad (2.22)
\end{aligned}$$

To obtain an approximation of the expected value, N trajectories are generated and the expected value is computed as the average of these trajectories.

The key idea behind policy gradient is to assign higher probabilities to actions that result in high rewards. However, Equation 2.22 treats all actions in a trajectory equally, regardless of their quality. Additionally, policy gradient tends to have high variance and slow convergence. To address these issues, one crucial trick is to incorporate causality into the algorithm. This means

Algorithm 5 REINFORCE

Initialize θ
while not done **do**
  sample $\{\tau^i\}$ from $\pi_\theta(a_t \mid s_t)$ (run the policy to sample trajectories)
  $\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_i^N \left[\left(\sum_{t=0}^T r\left(s_t^i, a_t^i\right)\right)\left(\nabla_\theta \sum_{t=0}^T \log \pi\left(a_t^i \mid s_t^i, \theta\right)\right]\right)$
  Update $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$
end while

that the policy at time t can only impact rewards at a later time t' if $t' > t$.

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_i^N \left(\nabla_\theta \sum_{t=0}^T \log \pi\left(a_t^i \mid s_t^i, \theta\right)\right)\left[\left(\sum_{t=0}^T r\left(s_t^i, a_t^i\right)\right)\right] \\
&\approx \frac{1}{N}\sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi\left(a_t^i \mid s_t^i, \theta\right)\right)\left[\sum_{t'=0}^T r\left(s_{t'}^i, a_{t'}^i\right)\right] \\
&\approx \frac{1}{N}\sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi\left(a_t^i \mid s_t^i, \theta\right)\right)\left[\sum_{t'=t}^T r\left(s_{t'}^i, a_{t'}^i\right)\right]
\end{aligned} \tag{2.23}$$

The updated approach enhances actions based on their anticipated future rewards, effectively resolving the issue of treating all actions within a trajectory equally. Furthermore, policy gradient methods are characterized by considerable variance and slow convergence, but two techniques can mitigate these challenges. The first technique, called causality, ensures that the policy at a given time "t" exclusively impacts rewards occurring at a later time "t'" if "t'" is more significant than "t." This constraint guarantees a temporal ordering of policy and rewards. The second technique involves reducing variance by incorporating a baseline. This baseline allows only actions that outperform the average to receive a boost. One popular choice for the baseline is the expected sum of rewards. By employing this baseline, the variance in policy gradient estimation is minimized.

$$b = \frac{1}{N}\sum_{i=0}^{N} r\left(\tau^i\right) \tag{2.24}$$

We can add a baseline since :

$$\begin{aligned}
\mathbb{E}_\pi\left[\nabla_\theta \log \pi_\theta(\tau) b\right] &= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) b d\tau \\
&= \int \nabla_\theta \pi_\theta(\tau) b d\tau \\
&= b \nabla_\theta \int \pi_\theta(\tau) d\tau \\
&= b \nabla_\theta 1 \\
&= 0
\end{aligned} \tag{2.25}$$

The gradient then becomes :

$$\nabla_\theta J(\theta) = \frac{1}{N} \sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi \left(a_t^i \mid s_t^i, \theta \right) \right) \left[\sum_{t'=t}^T r \left(s_{t'}^i, a_{t'}^i - b \right) \right] \tag{2.26}$$

output the mean $\mu(s)$ of a Gaussian distribution, and the variance $\sigma^2(s)$ can also be fixed. The action can then be sampled from the distribution as follows : $a \sim \mathcal{N}\left(\mu(s), \sigma^2(s)\right)$.

4.3.2.1 Actor Critic The following section draws inspiration from Sergey Levine's lecture [Levine 2017]. The Actor-Critic algorithm is a combination of both Policy Gradient and Q-learning. After the Policy Gradient algorithm, the result obtained (without considering the baseline) is as follows :

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi \left(a_t^i \mid s_t^i, \theta \right) \right) \left[\sum_{t'=t}^T r \left(s_{t'}^i, a_{t'}^i \right) \right] \tag{2.27}$$

The primary objective of the equation above is to guide actions toward maximizing rewards. However, the term $\sum_{t'=t}^T r\left(s_{t'}^i, a_{t'}^i\right)$ represents a single-sample estimate of future rewards obtained by starting from state s_t^i and taking action a_t^i. Since this estimate relies on a single sample, the policy gradient approach is susceptible to high variance, which can impede convergence speed and quality. To address this, reducing variance becomes crucial for achieving faster and more optimal convergence. A more effective approach involves obtaining a more accurate estimation of future rewards when initiating from state s_t^i and taking action a_t^i. One such estimator is the Q-value, defined as $Q_\pi(s,a) = \mathbb{E}_\pi \left[R_t \mid S_t = s, A_t = a \right]$. This Q-value estimator better assesses the expected rewards when following policy π starting from a specific state-action pair. Utilizing this estimator, the policy gradient method can yield improved performance and convergence. The equation 2.27 becomes :

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi \left(a_t^i \mid s_t^i, \theta \right) \right) Q \left(s_t^i, a_t^i \right) \tag{2.28}$$

In addition to the Q value estimator, a baseline is added to reduce bias. A commonly used baseline is the sum of rewards $\frac{1}{N} \sum_{i=0}^N r\left(\tau^i\right)$, but in this case, a more precise average can be computed by using the Q value as the baseline. Specifically, the baseline is set as $b = \frac{1}{N} \sum_i^N Q\left(s_t^i, a_t^i\right) \approx \mathbb{E} a_t \sim \pi\theta\left(a_t \mid s_t\right)\left[Q\left(s_t, a_t\right)\right] = V\left(s_t\right)$. With this baseline, the gradient becomes :

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi \left(a_t^i \mid s_t^i, \theta \right) \right) \left[Q \left(s_t^i, a_t^i \right) - V \left(s_t^i \right) \right] \tag{2.29}$$

We defined earlier the advantage $A(s,a) = Q(s,a) - V(s)$, which indicates how much action a is better than the average action.

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \sum_{t=0}^T \left(\nabla_\theta \log \pi \left(a_t^i \mid s_t^i, \theta \right) \right) \left[A \left(s_t^i, a_t^i \right) \right] \tag{2.30}$$

To compute the advantage, the following approximation is used :

$$Q\left(s_t, a_t\right) \approx r\left(s_t, a_t\right) + \gamma V\left(s_{t+1}\right) \tag{2.31}$$

This leads to the following :

$$A\left(s_t, a_t\right) = Q\left(s_t, a_t\right) - V\left(s_t\right) = r\left(s_t, a_t\right) + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \tag{2.32}$$

An example of an Actor-Critic algorithm :

Algorithm 6 Actor Critic

Initialize θ and ϕ
while not done **do**
  Sample $\{s_i, a_i\}$ from $\pi_\theta(a \mid s)$ (run the policy to sample trajectories)
  Fit $V_\phi^\pi(s)$ to sampled reward sums
  $A^\pi\left(s_i, a_i\right) = r\left(s_i, a_i\right) + \gamma V_\phi^\pi\left(s_i'\right) - V_\phi^\pi\left(s_i\right)$
  $\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_i^N \nabla_\theta \log \pi_\theta\left(a_i \mid s_i\right) A^\pi\left(s_i, a_i\right)$
  Update $\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$
end while

In essence, actor-critic algorithms consist of two components - an actor represented by the policy parameterized by θ and a critic (which could be Q, V or A depending on the specific algorithm) represented by the parameter ϕ that evaluates the actor's output. In the subsequent sections, we will overview various actor-critic algorithms.

4.3.2.2 Deterministic Policy Gradient : DDPG, RDPG, D4PG Traditional policy gradient algorithms are typically designed for stochastic policies. However,[Xiong, 2022] introduced a modified actor-critic algorithm, the "Deterministic Policy Gradient Algorithms," to handle deterministic policies. The deterministic policy gradient computation is more efficient since it involves differentiating the Q value solely concerning actions. To extend the Deterministic Policy Gradient (DPG) approach, the authors presented an off-policy algorithm that learns a deterministic policy using a stochastic policy for exploration. In "Continuous Control with Deep Reinforcement Learning" [Lillicrap, 2015], introduced the DDPG (Deep Deterministic Policy Gradient) algorithm, which combines techniques from DPG and DQN [Mnih, 2015]. DDPG utilizes an actor-critic framework to learn a deterministic policy similar to DPG. However, it incorporates Deep Q learning methods to estimate the critic, including experience replay and a target network. Additionally, the target network is updated slowly using the equation $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$, where τ is a small value close to 0. DDPG has demonstrated significant speed improvements on most Atari games, solving them in significantly fewer steps than DQN. Algorithm 7 presents the implementation of DDPG. However, studies conducted by [Henderson, 2018] and [Duan, 2016] have shown that DDPG is sensitive to hyperparameters and less stable than batch algorithms. Unlike many actor-critic algorithms that update the policy after sampling a batch of trajectories, DDPG updates the policy at each step.

In the paper [Heess, 2015], a variant of the RDPG (Recurrent Deterministic Policy Gradient) algorithm is introduced. This modified algorithm incorporates recurrent neural networks

Algorithm 7 DDPG from [Lillicrap, 2015]

Initialize critic network $Q(s, a \mid \phi)$ and actor $\mu(s \mid \theta)$ with weights ϕ and θ
Initialize target network Q' and μ' with weights $\phi' \leftarrow \phi, \theta' \leftarrow \theta$
Initialize replay buffer R
for episode $= 1, \mathrm{M}$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for $\mathrm{t} = 1,\ \mathrm{T}$ **do**
    Select action $a_t = \mu(s_t \mid \theta) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1})
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta') \mid \phi')$
    Update critic by minimizing the loss $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i \mid \phi))$
    Update the actor policy using the sampled policy gradient :

$$\nabla_\theta J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a \mid \phi) \Big|_{s=s_i, a=\mu(s_i)} \nabla_\theta \mu(s \mid \theta) \Big|_{s_i}$$

    Update the target networks :

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$$

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$

  end for
end for

and utilizes experience replay and target networks. It is highly effective in partially observed environments where the agent needs to retain information about previous states. Another extension of DDPG is the D4PG (Distributed Distributional Deterministic Policy Gradients) algorithm [Barth-Maron, 2018], which employs multiple actors for trajectory sampling and incorporates distributional critic updates [Bellemare, 2017]. Unlike traditional methods that focus on learning the expected return, D4PG aims to learn the distribution of returns, considering them as random variables. The algorithm also includes techniques such as Prioritized Experience Replay and N-step return. Notably, the utilization of N-step return yields significant performance improvements.

4.3.2.3 Reinforcement Learning with Multiple Workers : A3C, A2C, ACER

The A3C (Asynchronous Advantage Actor-Critic) algorithm, developed by Google DeepMind, utilizes parallelization to eliminate the need for a replay buffer. Instead of propagating with a batch of episodes, A3C performs smaller backpropagations in parallel with smaller batches. The algorithm consists of a global agent and multiple workers, each equipped with local networks. The global agent maintains up-to-date networks while each worker interacts with its instance of the environment. The workers copy the global networks locally, execute episodes, and compute gradients accordingly. The global network then gets updated with the gradients from the

local worker, and the process repeats as workers continue copying the global networks, running episodes, and so on. Since these tasks are executed in parallel by each worker, the update of the global network occurs asynchronously. A3C follows the actor-critic paradigm, which involves computing two gradients : one for the policy and another for the value function. The global agent and workers possess two networks with parameters θ and ϕ for the policy and value functions. Figure 2.6 visually represents the A3C algorithm, while Algorithm 8 presents the pseudocode for a single worker. It is important to emphasize that A3C is specifically designed for parallel computing and can significantly accelerate the training process.

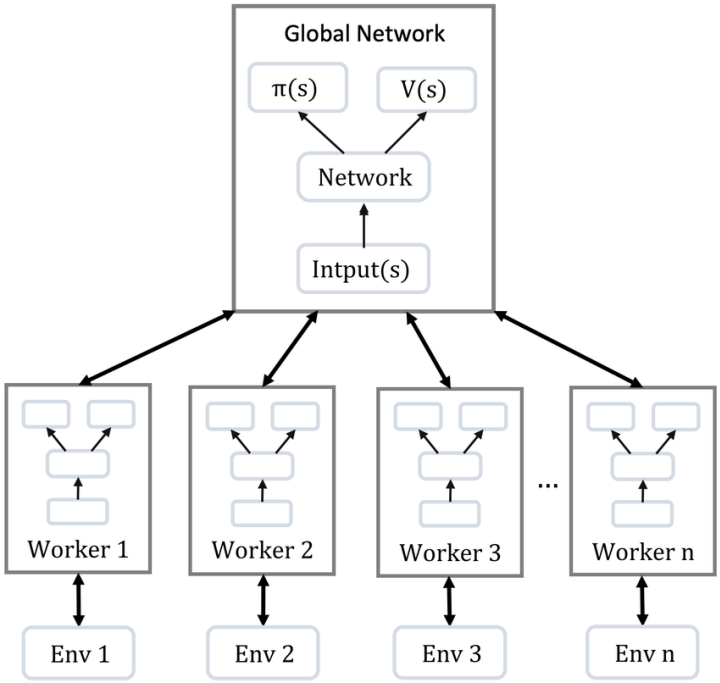


FIGURE 2.6 – A3C algorithm

The A3C algorithm efficiently performs various tasks, including Atari games and labyrinth navigation. Several extensions of A3C have been proposed to improve its performance further and address its limitations :

— A2C (Advantage Actor-Critic) is a synchronous version of A3C. It waits for all workers to finish their tasks before updating the global agent and copying the up-to-date weights to the workers. A2C has been found to perform similarly or even better than A3C.

— ACER [Wang, 2016b] is an off-policy extension of A3C that uses a replay buffer to improve sample efficiency.

— Impala [Espeholt, 2018] (Importance Weighted Actor-Learner Architecture) separates the acting and training processes. The actors do not compute gradients but send complete transitions to the global agent, which performs the updates. The training process can also be split into several learners to improve efficiency. Impala has been shown to perform well in large-scale distributed environments.

Algorithm 8 A3C - pseudo-code for one worker i (one actor-learner thread) from [Mnih, 2016]

Initialize thread step counter $t \leftarrow 1$
while $T > T_{\max}$ **do**
  Reset gradient : $d\theta_i \leftarrow 0$ and $d\phi_i \leftarrow 0$.
  Synchronize thread-specific parameters $\theta_i = \theta$ and $\phi_i = \phi$
  $t_{\text{start}} = t$
  Get state s_t
  while episode done **do**
    Perform a_t according to policy $\pi_{\theta_i}(a_t \mid s_t)$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  end while
  $R = \begin{cases} 0 & \text{for terminal state } s_t \\ V_{\phi_i}(s_t) & \text{otherwise} \end{cases}$
  for k from t-1 to t_{start} **do**
    $R \leftarrow r_k + \gamma R$
    Accumulate gradients wrt $\theta_i : d\theta_i \leftarrow d\theta_i + \nabla_{\theta_i} \log \pi_{\theta_i}(a_k \mid s_k)(R - V_{\phi_i}(s_k))$
    Accumulate gradients wrt $\phi_i : d\phi_i \leftarrow d\phi_i + \partial (R - V_{\phi_i}(s_k))^2 / \partial \phi_i$
  end for
  Perform asynchronous update of θ using $d\theta_i$ and of ϕ using $d\phi_i$
end while

4.3.2.4 Trust Region Algorithms : TRPO, ACKTR, PPO

In order to perform actions in continuous space, policy gradient algorithms require careful selection of the step size α. If the value of α is too small, the learning process can be slow, and if it is too large, it can be overwhelmed by noise, resulting in the next batch being collected under a poor policy. To address this issue, [Schulman, 2015] proposed Trust Region Policy Optimization, which modifies the policy at every update to improve performance while remaining close to the previous policy. The primary objective of this algorithm is to maximize $\eta(\theta) = \mathbb{E}_\pi \left[\gamma^t r(s_t)\right]$, which is difficult to compute directly. Therefore, the authors introduce a surrogate function to approximate this objective.

$$L(\theta) = \mathbb{E}\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}(a_t|st_t)}} A_t\right] \tag{2.33}$$

That is a local approximation of the true objective and which first-order derivatives are equal.

$$\begin{aligned} \nabla_\theta L(\pi_\theta)|_{\theta_{\text{old}}} &= \mathbb{E}_{s,a\sim\pi_{\text{old}}}\left[\frac{\nabla_\theta \pi_\theta(a \mid s)}{\pi_{\text{old}}(a \mid s)} A^{\pi_{\text{old}}}(s,a)\right]\Big|_{\theta_{\text{old}}} \\ &= \mathbb{E}_{s,a\sim\pi_{\text{old}}}\left[\nabla_\theta \log \pi_\theta(a \mid s) A^{\pi_{\text{old}}}(s,a)\right]|_{\theta_{\text{old}}} \end{aligned} \tag{2.34}$$

We get the formula of the gradient derivative as in equation 2.30, so at first order, when θ and θ_{old} are close.

$$\nabla_\theta L\left(\pi_\theta\right)|_{\theta_{\text{old}}} = \nabla_\theta \eta\left(\pi_\theta\right)|_{\theta=\theta_{\text{old}}} \tag{2.35}$$

The objective of the TRPO algorithm is to maximize the expected discounted return $\eta(\theta) = \mathbb{E}\pi[\sum t = 0^{\infty}\gamma^t r(s_t)]$, subject to a constraint on the divergence between the new policy π and the old policy π_{old}. The Kullback-Leibler (KL) divergence is used to measure this constraint. This ensures that the new policy is not too far from the old policy, preventing extensive policy updates that may lead to instability. The algorithm iteratively finds the policy that maximizes this objective while keeping the KL divergence between the old and new policies under a certain threshold.

$$\begin{cases} \underset{\theta}{\text{maximize}} & \sum_n^N \frac{\pi_\theta(a_n|s_n)}{\pi_{\theta_{old}(a_n|s_n)}} A_n \\ \text{subject to} & \overline{KL}\left[\pi_{\text{old}}, \pi\right] < \delta \end{cases} \tag{2.36}$$

A team from OpenAI proposed a modified version of the TRPO algorithm called the Proximal Policy Gradient Algorithm [Schulman, 2017] ACKTR [Wu, 2017] is another variation of TRPO that uses Kronecker-factored approximate curvature instead of Kullback-Leibler divergence for trust region optimization. The authors of both algorithms suggest two different surrogate functions to ensure that the new policy π is close to the old one π_{old}. The first surrogate function, L^{KLPEN}, directly adds the KL penalty from Equation 2.36 to the objective function instead of using a separate constraint.

$$L^{KLPEN}(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} A_t - \beta KL\left[\pi_{\theta_{old}}, \pi_\theta\right]\right] \tag{2.37}$$

The parameter β is updated as follows. With $d = \mathbb{E}_t\left[KL\left[\pi_{\theta_{old}}, \pi_\theta\right]\right]$

$$\begin{array}{lll} \text{if} & d < d_{\text{targ}}\ /1.5 & \beta \leftarrow \beta/2 \\ \text{if} & d > d_{\text{targ}}\ \times 1.5 & \beta \leftarrow \beta \times 2 \end{array} \tag{2.38}$$

Instead of using the KL penalty, a clipped surrogate function L^{CLIP} can be used as a more stable alternative. Here, $r_t(\theta)$ is defined as the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, and A_t is still the advantage at timestep t. The clipped surrogate function is then defined as follows :

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}\left(s_t(\theta), 1-\epsilon, 1+\epsilon\right)A_t\right)\right] \tag{2.39}$$

This clipped function will force the $r_t(\theta)$ to stay in $[1-\epsilon, 1+\epsilon]$ range.

In case the neural network parameters are shared between the policy and the value function in the Actor-Critic algorithm, the objective function is as follows :

$$L(\theta) = \mathbb{E}_t\left[L_t^{PG}(\theta) + c_1 L_t^{VF}(\theta) - c_2 S\left[\pi_\theta\left(s_t\right)\right]\right] \tag{2.40}$$

The objective function used in the Actor-Critic algorithm includes either the clipped surrogate function L^{CLIP} or the Kullback-Leibler penalty surrogate function L^{KLPEN}. Additionally, a squared error value loss L^{VF} is used to compute the value function V, and an entropy bonus

S is included to encourage exploration. These algorithms were evaluated using the Atari game benchmark and robotics tasks in the Mujoco simulator [Todorov, 2012]. PPO is simpler than TRPO and ACER and allows the value and policy networks to share parameters, unlike TRPO, which has a constraint on the policy. The PPO algorithm outperformed A2C and TRPO on the robotics tasks and was competitive with ACER on the Atari games, outperforming it on the Enduro game.

4.3.2.5 Entropy with Reinforcement Learning : SQL, SAC A common challenge in reinforcement learning is the tendency of agents to become overly specialized and only learn a single approach to solving a particular task. A robot trained with traditional reinforcement learning will exclusively focus on the optimal upper passage and struggle to adapt if that route becomes blocked. To address this issue and promote the exploration of alternative solutions, one approach is to incorporate entropy into the learning process. By maximizing both the sum of rewards and the entropy, agents are encouraged to learn multiple ways of solving the task. This approach is demonstrated in the paper "Reinforcement Learning with Deep Energy-Based Policies" by [Haarnoja, 2017], where the optimal policy is defined as :

$$\pi* = \underset{\pi}{\text{argmax}} \mathbb{E}_{\pi} \left[\sum_{t} r_t + \mathcal{H} \left(\pi \left(\cdot, s_t \right) \right) \right] \tag{2.41}$$

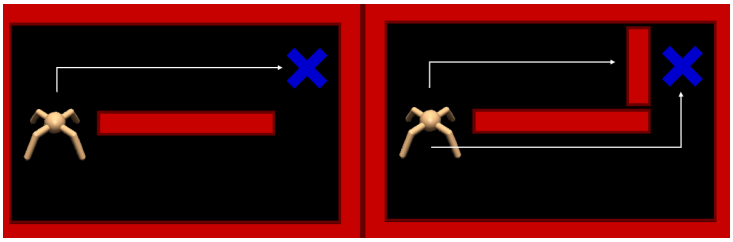


FIGURE 2.7 – In a maze navigation task, shown in a figure by [Wei, 2018], classical reinforcement learning would cause the robot to learn only the optimal path to reach the goal (left). However, by adding entropy regularization to the learning process (right), the robot is encouraged to learn multiple paths to reach the goal, which enables it to adapt to changes in the environment.

The agent's tendency to become too specialized in solving a specific task is a common issue in reinforcement learning. For instance, in the example shown in Figure 2.7, a robot trained using classical RL will focus only on the upper passage, which is the optimal route. However, if this passage is blocked, the robot will have to start from scratch and might be unable to learn an alternative route. Entropy can be added to the learning process to address this problem and encourage the robot to explore and learn all possible ways to solve the task. This method, called Soft Q-Learning (SQL), involves maximizing both the sum of rewards and the entropy. Using SQL, the robot will learn both passages but still prefer the optimal route. The robot can quickly switch to the alternative route if the optimal passage is blocked.

Soft Q-Learning (SQL) serves as the predecessor of Soft Actor-Critic (SAC), which is presented in the paper "Soft Actor-Critic : Off-Policy Maximum Entropy Deep Reinforcement Learning

with a Stochastic Actor" [Haarnoja, 2018]. SAC combines off-policy updates with maximum entropy to improve exploration and learning. Later work by [Haarnoja, 2018] in 2018 extended SAC by introducing automatic tuning of hyperparameters, resulting in more regular training. Many reinforcement learning algorithms, such as A3C and PPO, now incorporate entropy in their surrogate functions for improved exploration and robustness, as discussed in the paper by [Ahmed, 2019].

The primary goal of a RL agent is to find a policy that maximizes the total reward over various possible paths the agent can take while following that policy. During the exploration of state and action spaces to determine the optimal policy, the agent traverses various paths, each with an expected length defined by the task horizon. When faced with large state and action spaces and a long task horizon, traditional RL methods encounter challenges [Nachum, 2019][Pathak, 2017].

4.4 Hierarchical Reinforcement Learning

Hierarchical Reinforcement Learning (HRL) offers a solution by breaking down complex tasks into simpler subtasks through a hierarchy of learned policies. In this hierarchy, the top-level policy selects subtasks of the main task as its actions [Pateria, 2021]. This policy is trained to execute the main task by sequencing its subtasks based on rewards obtained during the main task. At lower levels of the hierarchy, each subtask chosen by the higher-level policy becomes a reinforcement learning problem itself. A lower-level policy is trained to execute that subtask using internal rewards associated with it, with the option of including the main task reward. The lowest-level policies select basic actions known as primitive actions.

4.4.1 Definition of a Subtask in Hierarchical Reinforcement Learning

This section presents a formal definition of a subtask within the context of HRL. It combines various interpretations from different HRL methodologies and should be viewed as a comprehensive understanding.

To begin, we designate the primary long-horizon task as Γ and denote its corresponding task policy as π_Γ, situated at the apex of a hierarchical structure. A subtask is represented by ω, and its definition encompasses the following components :

- The policy of the subtask, denoted as π_ω, which maps environment states to either primitive actions or subtasks within ω [Sutton, 1999].
- Objective components :
  - r_ω, the subtask reward utilized for training π_ω, often distinct from the reward linked with the main task [Vezhnevets, 2017].
  - g_ω, a subgoal or set of subgoals associated with ω, which might encompass a state $s \in S$ itself [Pateria, 2021], an abstract state representation [Pateria, 2021] a learned embedding [Vezhnevets, 2017], among other possibilities. The reward r_ω may be defined in relation to these subgoal(s).
- Execution components :

— I_ω, the initiation condition of ω, which may be defined as a set of states enabling the execution of ω [Sutton, 1999], a function influencing the likelihood of selecting ω in a given state, a series of logical conditions, etc.

— β_ω, the termination condition of ω, which might be articulated as a set of states indicating when ω should conclude if it is being executed [Pateria, 2021], a function modulating the likelihood of terminating ω in a specific state [Sutton, 1999], a fixed time constraint, etc. If g_ω is specified, the subgoal state typically dictates a state at which ω must conclude [Pateria, 2021].

4.4.2 Formulation of HRL within the Framework of Semi-Markov Decision Processes (SMDP)

The underpinning of HRL is grounded in the theory of Semi-Markov Decision Processes (SMDP) [Sutton, 1999]. Unlike Markov Decision Processes (MDP) discussed in Section 4.4.1, an SMDP introduces the element of time regarding the duration for which an action persists after selection. In the context of HRL, these actions with temporal considerations correspond to subtasks. Assuming an agent starts from a state $s_t \in S$ and selects a subtask $\omega_t \in \Omega$, where Ω denotes the set of subtasks (or the subtask space), the transition function of the SMDP is defined by a joint distribution :

$$P(s_{t+c_{\omega_t}}, c_{\omega_t} \mid s_t, \omega_t) = P(s_{t+c_{\omega_t}} \mid s_t, \omega_t, c_{\omega_t}) P(c_{\omega_t} \mid s_t, \omega_t). \tag{2.42}$$

Here, c_{ω_t} represents the number of time steps for which ω_t is executed, commencing from state s_t. This duration c_{ω_t} is determined by the termination condition β_{ω_t}, one of the execution components elaborated in Section 4.4.1.

The reward obtained when performing subtask ω_t starting from state s_t is denoted as $R(s_t, \omega_t)$, calculated as :

$$R(s_t, \omega_t) = \mathbb{E}_{a \sim \pi_{\omega_t}(s)} \left[\sum_{i=0}^{c_{\omega_t}-1} \gamma^i r(s_{t+i}, a_{t+i}) \mid s_t, a_t = \pi_{\omega_t}(s_t) \right]. \tag{2.43}$$

This equation illustrates that the reward $R(s_t, \omega_t)$ is the expected cumulative reward acquired while following the subtask policy π_{ω_t} from time t until the termination of ω_t after c_{ω_t} time steps.

An optimal task policy aims to achieve the maximum desired Q-value :

$$Q\left(s_t, \omega_t\right) = R\left(s_t, \omega_t\right) + \sum_{s_{t+c_{\omega_t}}, c_{\omega_t}} \gamma^{c_{\omega_t}} P\left(s_{t+c_{\omega_t}}, c_{\omega_t} \mid s_t, \omega_t\right) \max_{\omega_{t+c_0 t}} Q\left(s_{t+c_{\omega_t}}, \omega_{t+c_{\omega_t}}\right), \tag{2.44}$$

for all $s \in S$ and all $\omega \in \Omega$.

It's noteworthy that the Q-value in Equation 2.44 also relies on $R(s_t, \omega_t)$ and $P(s_{t+c_{\omega_t}}, c_{\omega_t} | s_t, \omega_t)$. These quantities are contingent on the execution of ω_t using its policy π_{ω_t}. Hence, an agent needs to learn multiple policies at various levels of a task decomposition hierarchy, encompassing π_Γ

and the policies of all subtasks. We extend the notations of subtasks and policies for a multi-level hierarchy as defined below.

— ω^l : A subtask at level l of the hierarchy.

— Ω_{ω^l} : Set of subtasks under the subtask ω^l such that $\omega^{l-1} \in \Omega_{\omega^l}$.

— $\pi_{\omega^l} : S \times \Omega_{\omega^l} \to [0,1]$: Policy of the subtask ω^l, where ω^{l-1} is chosen by π_{ω^l}.

— $\Omega_{\omega^1} = A$: The output space of a subtask at the lowest level ($l = 1$) is the primitive action space A.

— π_Γ and Ω_Γ denote the main task policy and the set of subtasks at the highest level, respectively.

Combining all the definitions provided above, two principal components of an HRL agent emerge :

— Subtask space $\Omega_{\text{hierarchy}}$: The encompassing set of all subtasks employed in a hierarchy, $\Omega_{\text{hierarchy}} = \{\Omega_{\omega^2}, \Omega_{\omega_3}, \Omega_{\omega_4}, ..., \Omega_\Gamma\}$.

— Hierarchical policy $\pi_{\text{hierarchy}}$: The primitive action chosen by the HRL agent is the outcome of recursive selections of subtasks. Here, the primary policy π_Γ selects a level 2 subtask, $\omega^2 = \pi_\Gamma(s)$, where $\omega^2 \in \Omega_\Gamma$. The policy of ω^2 is executed until its termination according to β_{ω^2}. It selects the lowest-level subtask $\omega_1 = \pi_{\omega^2}(s)$, where $\omega_1 \in \Omega_{\omega^2}$. The policy of ω^1 is executed until its termination according to β_{ω^1}. This lowest-level policy chooses a primitive action, $a = \pi_{\omega^1}(s)$. This complete mapping from state to subtask to action based on π_{ω^1} is termed the hierarchical policy, denoted as $\pi_{\text{hierarchy}}$. The primitive action taken by the HRL agent as a whole can be equivalently expressed as $a = \pi_{\text{hierarchy}}(s)$. This description extends to hierarchies with more than three levels.

Based on the aforementioned definitions, the expected discounted cumulative reward received by the HRL agent can be expressed as :

$$Q^{\text{hierarchy}}\ (s_t, a_t) = \mathbb{E}_{a \sim \pi_{\text{hierarchy}}\ |\Omega_{\text{hierarchy}}} \left[\sum_{i=0}^{\infty} \gamma^{t+i} r\left(s_{t+i}, a_{t+i}\right) \mid s_t, a_t \right], \tag{2.45}$$

where $a \sim \pi_{\text{hierarchy}} \mid \Omega_{\text{hierarchy}}$ signifies that a primitive action a is selected following the hierarchical policy $\pi_{\text{hierarchy}}$, considering the available subtask space $\Omega_{\text{hierarchy}}$.

4.4.3 Problem Definition of HRL

The overarching objective of Hierarchical Reinforcement Learning (HRL) is to determine the optimal hierarchical policy $\pi^*_{\text{hierarchy}}$ and the optimal subtask space $\Omega^*_{\text{hierarchy}}$ as the solution to :

$$\Omega^*_{\text{hierarchy}}, \pi^*_{\text{hierarchy}} = \text{argmax}_{\Omega_{\text{hierarchy}}} \text{argmax}_{\pi_{\text{hierarchy}} | \Omega_{\text{hierarchy}}} Q_{\text{hierarchy}}(s,a), \forall s \in S, a \in A. \tag{2.46}$$

Equation 2.46 encapsulates the HRL problem, which can be dissected into two primary facets.

The first facet involves learning the hierarchical policy, where the aim is to identify the optimal hierarchical policy conditioned on the available subtask space (i.e., maximizing $\pi_{\text{hierarchy}} \mid \Omega_{\text{hierarchy}}$). This aspect is crucial as the hierarchical policy governs the behavior of an HRL agent across various tasks. Strategies for learning the policies at different levels of $\pi_{\text{hierarchy}}$ may involve simultaneous learning in an end-to-end manner or sequential learning from bottom to top.

The second facet pertains to subtask discovery, which involves automatically identifying the optimal subtask space utilizing the experiential data of the HRL agent (i.e., maximizing $\Omega_{\text{hierarchy}}$). While subtask discovery is not obligatory, as a subtask space could be manually designed based on specific domain knowledge, it becomes indispensable for generalized HRL applications aiming to eliminate reliance on manual crafting.

5 Sensors

In the context of autonomous vehicles, ensuring reliable and accurate environmental perception is paramount to safe driving. To capture a comprehensive understanding of the surroundings, it is crucial to fuse data from multiple sensors [Hu, 2020]. Autonomous vehicles heavily rely on many sensors installed on the ego vehicle, including cameras, radar, LiDAR, and ultrasonic sensors, to sense and perceive the environment. These sensors work in synergy to provide a holistic view of the surroundings, detecting obstacles, road conditions, and other relevant information.

In addition to the above, sensors, such as the Global Positioning System (GPS), the Inertial Measurement Unit (IMU), and vehicle odometry sensors, are utilized to determine the vehicle's relative and absolute localization. The GPS aids in obtaining the vehicle's global position, while the IMU provides information about the vehicle's orientation and acceleration. Vehicle odometry sensors are vital in estimating the vehicle's motion, including speed and direction.

By integrating and fusing data from diverse sensors, autonomous vehicles can create a detailed and accurate perception of the environment. This comprehensive perception is essential for making informed decisions and ensuring the vehicle's and its occupants' safety.

5.1 Camera

Among the sensors used in the computer vision stack of autonomous vehicles, the camera plays a fundamental role and is widely employed. Generally, three types of cameras are utilized in the context of autonomous vehicles : monocular, binocular, and multi-modal.

Monocular cameras are the most prevalent in advanced driver-assistance systems (ADAS), serving as the backbone of visual perception in autonomous vehicles. They operate by capturing single images and providing real-time feedback on the vehicle's surroundings. Monocular cameras excel in offering clear visuals in close-range scenarios, facilitating tasks such as lane-keeping, pedestrian detection, and traffic sign recognition. However, their effectiveness diminishes when tasked with capturing distant objects or extensive views. This limitation arises from their inability to accurately gauge depth and spatial relationships beyond a certain distance, leading to reduced performance in tasks like long-range obstacle detection and environment mapping.

On the other hand, Binocular cameras address the limitations of monocular cameras by utilizing a pair of cameras. This configuration enables depth perception and better clarity in various scenarios. Each camera captures a slightly different perspective, enabling the system to calculate distances through disparity analysis. This depth of information is pivotal for tasks like safe navigation and object recognition. Additionally, binocular cameras employ techniques such as stereo correspondence and 3D reconstruction to improve image clarity by reducing noise and generating richer 3D representations of the environment. This capability enhances understanding of complex scenes and enables better occlusion handling.

Multi-modal cameras integrate a visual camera with additional sensors like LiDAR, thermal cameras, and RADAR. This integration enhances environmental understanding crucial for autonomous vehicles, robotics, and surveillance systems. However, challenges such as system complexity, higher costs, and computational demands accompany this integration.

Despite the advantages of binocular and multi-modal cameras, monocular cameras remain prevalent in autonomous driving due to their affordability compared to the other types and lower computational demands. While binocular cameras offer enhanced depth perception, their higher costs hinder broader adoption. Overall, the choice of camera type in autonomous driving scenarios is influenced by factors such as cost, computational resources, and system effectiveness.

5.2 LiDAR

LiDAR (Light Detection And Ranging) is a remote sensing technique employed in various fields to measure the distance between objects. It operates by emitting laser beams and detecting the signals reflected from the targets. LiDAR sensors analyze the time interval between the emission and reception of the laser pulses, enabling accurate distance estimation.

LiDAR sensors generate point data, commonly known as point cloud data (PCD), in dimensions such as 1D, 2D, and 3D. These point clouds provide spatial information about the surroundings. Additionally, LiDAR sensors capture data on object intensity, which helps distinguish objects based on their reflectivity.

In the case of 3D LiDAR sensors, the point cloud data comprises the x, y, and z coordinates, representing the position of objects in three-dimensional space. The intensity information of the obstacles present in the scene or surroundings is also recorded along with spatial coordinates. This spatial and intensity data combination facilitates accurate mapping and detection of objects in LiDAR's field of view.

By leveraging LiDAR technology, autonomous vehicles and other applications can create detailed and precise representations of their environment. LiDAR sensors contribute valuable spatial data, allowing for advanced perception, object recognition, and navigation in complex and dynamic surroundings.

5.3 RADAR

RADAR technology, first introduced into vehicles for automated parking assistance systems [Jeong, 2010], is vital in enhancing vehicle safety and awareness. The RADAR emits electro-

magnetic (EM) waves into the region of interest, detecting scattered waves or reflections from various targets. By utilizing the Doppler effect of electromagnetic waves, RADAR systems can accurately determine the distance and relative speed of detected obstacles [Shahian Jahromi, 2019]. RADAR's ability to emit and receive EM waves provides valuable data for analyzing the surrounding environment. This data enables the calculation of object ranges and velocities, contributing to advanced driver assistance systems and collision avoidance mechanisms. By integrating RADAR technology, vehicles can improve their ability to detect and respond to potential hazards in real-time, enhancing safety for vehicle occupants and other road users. Combining RADAR and other sensors like LiDAR and cameras allows for a multi-modal perception system in autonomous vehicles. By fusing the data from these sensors, a comprehensive understanding of the environment can be achieved, facilitating robust decision-making and safe navigation.

5.4 GPS/IMU

The combination of GPS/IMU is a cost-effective and real-time localization method in autonomous driving [Atia, 2017]. By fusing GPS and IMU data, it becomes possible to achieve real-time localization with minimal delay, high precision, and a high frequency of updates.

GPS can provide precise location information at the centimeter level in dynamic environments. However, it is essential to note that the satellite signal can be obstructed, resulting in occasional disruptions in the availability of position updates. Furthermore, the frequency of GPS position updates may be relatively modest.

To address these limitations, IMUs and odometers play a crucial role. They continuously gather data on displacement and direction deviations during the interval between two successive GPS positioning updates. This data is then utilized to refine and adjust the localization estimates. While IMUs and odometers provide frequent updates, their accuracy degrades over time due to inherent error accumulation.

The fusion of GPS and IMU enables autonomous vehicles to achieve real-time localization with a balance of accuracy and reliability. By leveraging the high precision of GPS when available and compensating for temporary signal loss or limitations, the IMU data bridges the gaps and maintains localization information. This approach allows for precise navigation and control in autonomous driving scenarios.

6 Answer Set Programming

Answer Set Programming (ASP) represents a form of declarative programming tailored explicitly for tackling challenging, primarily NP-hard search problems. It has emerged due to research focused on employing nonmonotonic reasoning in knowledge representation, making it highly valuable for knowledge-intensive applications. ASP foundation lies in the stable model (answer set) semantics of logic programming, which draws from concepts in autoepistemic logic and default logic to handle negation as failure.

In the context of ASP, search problems are transformed into the computation of stable models. Answer set solvers, programs designed to generate stable models, are then employed to

conduct the search process [Gomes, 2008]. These solvers often build upon the Davis-Putnam-Logemann-Loveland (DPLL) procedure and exhibit similarities to the algorithms utilized in efficient SAT solvers. A notable characteristic of ASP's algorithms is their capacity to permanently terminate in principle, distinguishing them from the Selective Linear Definite Clause Resolution with Negation as Failure (SLDNF) resolution utilized in Prolog. The ASP methodology has existed for approximately a decade. An early demonstration of answer set programming can be found in the planning method proposed by [Dimopoulos, 1997], which establishes a connection between plans and stable models, as described in [Subrahmanian, 1995]. Moreover, [Tiihonen, 2003] applied the principles of answer set programming, now recognized as such, to address the product configuration problem. The application of answer set solvers for the search was first identified as a new programming paradigm in [Marek, 1999], with the term "answer set programming" being used for the first time in [Niemelä, 1999]. The basic idea behind ASP is to express a given problem in the form of a logic program, for which we need to search for stable models representing the solutions to the original problem. We first use rules in first-order logic to concisely express the problem. Consequently, the problem will be expressed by a logic program, often referred to as a non-terminal program, containing predicates with variables. To find stable models, the most efficient solvers adopt a two-phase approach. The first phase is the instantiation of the variables, generally called grounding. It involves transforming a logic program expressed in first-order logic into a propositional program. The resulting program will no longer contain any variables but keep stable patterns identical to the original. The second phase is the resolution, which calculates the program's stable models of the program.

6.1 Answer Set Programming Language

The System LPARSE [Gebser, 2007] was initially developed as a front-end tool for the answer set solver SMODELS [Syrjänen, 2001]. Nowadays, it serves the same purpose as most other answer set solvers. In LPARSE programs, it can find traditional "Prolog-style" rules, like :

$$\begin{cases} \text{r1 : p :- q.} \\ \text{r2 : q :- } \neg r. \end{cases} \tag{2.47}$$

A collection of such Prolog-style rules typically has a unique stable model, which often includes all the queries Prolog would respond "yes." For example, with the two rules mentioned above, the stable model consists of p and q. In addition to Prolog-style rules, LPARSE also accommodates "choice rules," such as :

$$\{\text{r3 : s, t :- p.} \tag{2.48}$$

This rule means that if p is part of the stable model, it can arbitrarily choose to include either s or t. When using SMODELS to find all stable models of a program P containing these three

rules, the output would be as follows :

$$\begin{cases} \text{Answer 1 : Stable Model : p q} \\ \text{Answer 2 : Stable Model : t p q} \\ \text{Answer 3 : Stable Model : s p q} \\ \text{Answer 4 : Stable Model : s t p q} \end{cases} \tag{2.49}$$

Choice rules can include numerical bounds in their heads. For instance, the rule :

$$\{\text{r4. 1 s, t :- p.} \tag{2.50}$$

States that if p is generated, then at least one of the atoms s or t must also be generated. By substituting this rule for the last one in P, Answer 1 would disappear from the SMODELS output.

The definition of a stable model from [Gelfond, 1988] was extended by [Niemelä, 1999] to encompass programs with choice rules and other rules involving numerical bounds. [Ferraris, 2005] provided a general definition of a stable model covering these rules and others.

In LPARSE, a constraint is a rule with an empty head, like :

$$\{\text{r5. :- s, } \neg t. \tag{2.51}$$

Adding a constraint to a program removes some of its stable models. For instance, the constraint above forbids generating s if t is not generated, which would eliminate Answer 3. Before passing a program with variables to SMODELS, LPARSE grounds it (replaces it with an equivalent program without variables). For example :

$$\begin{cases} \text{Original :} \\ \text{p(a). p(b). p(c).} \\ \text{q(X) :- p(X).} \\ \text{r(X) : p(X).} \\ \text{Grounded :} \\ \text{p(a). p(b). p(c).} \\ \text{q(a) :- p(a).} \\ \text{q(b) :- p(b).} \\ \text{q(c) :- p(c).} \\ \text{r(a), r(b), r(c).} \end{cases} \tag{2.52}$$

6.2 Programming Methodology : Generate, Define, Test

Consider a self-driving car program to navigate a complex intersection. We want the car to identify safe paths that avoid collisions with other vehicles and adhere to traffic laws. Here, we

can leverage ASP to achieve this goal using a generate-and-test methodology.

6.2.1 Generate Potential Solutions

The first step is to generate a set of potential paths the car can take. Let's represent the intersection as a graph, where roads are edges and intersections are vertices. We can use the following ASP rule to define a predicate path(X,Y) :

$$\Big\{ \text{path(X,Y) :- edge(X,Y).} \tag{2.53}$$

This rule presents a potential path between two spatial points, X and Y, whereby the presence of a direct edge linking X to Y signifies the existence of such a path. Essentially, it conceptualizes any sequence of interconnected road segments (edges) as a plausible route connecting the two spatial points.

6.2.2 Define Constraints (Test)

Next, we need to define constraints (rules) to eliminate unsafe paths from our potential solutions. Here are some examples :

6.2.2.1 No Collisions : We can define a rule to eliminate paths where the car would be on the same road segment (edge) as another vehicle at the same time. Here's the corresponding ASP rule :

$$\Big\{ \text{collision\_avoidance(X, Y, T) :- path(X, Y), not occupied(X, T)} \tag{2.54}$$

This rule ensures safe movement between points X and Y at time T. It verifies that a path exists between these points and that point X is not occupied at the specified time.

6.2.2.2 Traffic Signals : Similarly, a rule can be defined to allow to proceed that violates traffic signals, like going through a red light :

$$\Big\{ \text{not\_allowed\_to\_proceed(X, Y, T) :- path(X, Y), traffic\_light(red, X, T).} \tag{2.55}$$

Here, the condition checks if there is a path from location X to location Y, along with a red light at intersection X at time T. If this condition is met, it implies that the vehicle is not allowed to proceed along the specified path at the indicated time due to the presence of a red light at the intersection.

6.2.2.3 Traffic Laws We can incorporate specific traffic laws into the program. Here's an example of a rule that forbids to proceed at an intersection if another vehicle is coming on the right.

$$\begin{cases} \text{not\_allowed\_to\_proceed(X, Y, T) :- path(X, Y), path(Z, X), occupied (Z, T),} \\ \text{coming\_from\_right(Z, T).} \end{cases} \tag{2.56}$$

This rule states that if a path from location Z to location X is located to the right of a path from location X to location Y and the former is occupied by a vehicle (coming on the right), then the vehicle at location X is prohibited from moving forward.

6.2.3 Combining the Rules

The first rule generates the brake(T) action if the brake conditions are satisfied. The second rule specifies these conditions ; for instance, here, it is when a collision may occur. The third rule defines the conditions of having a collision. These three rules can be activated in cascade.

$$\begin{cases} \text{brake(T) :- brake\_conditions(T).} \\ \text{brake\_conditions(T) :- not\_collision\_avoidance(X, Y, Z, T).} \\ \text{not\_collision\_avoidance(X, Y, T) :- path(X, Y), not occupied(X, T)} \end{cases} \tag{2.57}$$

6.3 Representing Incomplete Information

From the knowledge representation perspective, a set of atoms can be seen as a complete state of knowledge. Atoms in the set are known to be accurate, while atoms not in the set are considered false. Representing a possibly incomplete state of knowledge involves using a consistent but incomplete set of literals. If an atom "p" and its negation do not belong to the set, it is uncertain whether "p" is true or false.

In logic programming, this distinction gives rise to two types of negation : negation as failure, demonstrated above, and strong (or "classical") negation, denoted in the language of LPARSE by "¬". To illustrate the difference between these types of negation, consider the example of a bus crossing railway tracks only when there is no approaching train :

The rule cross : − ¬train. is not an adequate representation because it implies that crossing is acceptable when there is no information about an approaching train. The more suitable rule that uses strong negation is cross : − ¬train., stating that crossing is permissible when we know that no train is approaching.

By combining both forms of negation in the same rule, we can express the closed world assumption, which assumes that a predicate does not hold unless there is evidence to support it [Reiter, 1980]. For example, the rule $\neg q(X, Y)$:- $\neg q(X, Y), p(X), p(Y)$.

indicates that the binary relation q is considered false for a pair of elements from p unless there is supporting evidence suggesting otherwise.

In an ASP program with strong negation, the closed-world assumption rules can be applied to some predicates while leaving other predicates under the open-world assumption. This allows for a more flexible and nuanced representation of knowledge and uncertainty.

7 Conclusion

In conclusion, this chapter emphasizes the importance of reliable and precise environmental perception in autonomous vehicles. Autonomous vehicles heavily rely on diverse sensors, including cameras, LiDAR, radar, ultrasonic sensors, GPS, IMU, and vehicle odometry sensors, to create a comprehensive view of their surroundings. This multifaceted perception is indispensable for making informed decisions and ensuring the vehicle's and its occupants' safety.

The chapter further explores individual sensor types, elucidating their specific roles and contributions to environmental perception. Each sensor type, such as monocular cameras, LiDAR, radar, GPS, and IMU, has advantages and limitations, making them suitable for various autonomous driving applications. The integration of these sensor modalities empowers autonomous vehicles with multi-modal perception, enabling robust decision-making and safe navigation, even in complex and dynamic scenarios.

As autonomous vehicle technology advances, sensors will remain at the forefront of environmental perception. Ongoing developments in sensor technologies and sensor fusion algorithms are set to further enhance the safety and capabilities of autonomous vehicles on our roads.

This chapter provides a comprehensive exploration of deep neural networks, supervised learning, and reinforcement learning algorithms, with a particular emphasis on model-free deep reinforcement learning. While supervised learning relies on annotated data, reinforcement learning has demonstrated its prowess in solving sequential decision problems without such annotations. The thesis prioritizes reinforcement learning for vision-based navigation, using a continuous action-capable algorithm, Proximal Policy Optimization (PPO), to achieve the task.

PPO stands out as a state-of-the-art method for image-based reinforcement learning and offers stability and efficiency, making it an excellent choice for the study's objectives. The subsequent chapter will further delve into applying reinforcement learning to vision-based navigation tasks, building upon the foundation established in this chapter.

Lastly, the chapter introduces readers to Answer Set Programming (ASP), a powerful declarative programming paradigm designed for addressing complex, often NP-hard search problems. ASP leverages nonmonotonic reasoning in knowledge representation, offering valuable capabilities for knowledge-intensive applications. This approach relies on the stable model (answer set) semantics of logic programming, effectively handling negation as failure. Within ASP, complex search problems are translated into the computation of stable models, and answer set solvers are employed to facilitate the search process.

In summary, this chapter has laid the foundation for understanding and utilizing ASP, which will be further explored and applied in subsequent chapters to address complex problems and decision-making in autonomous systems. In the following chapter, our primary focus will be on addressing safety concerns related to self-driving vehicles. We aim to conduct an extensive and methodical examination of the current scientific literature and cutting-edge technologies about the safety aspects of autonomous vehicles.

Chapitre 3

State of the art

Contents

1 Introduction

The primary objective of this chapter is to discuss safety related to self-driving vehicles. Our main objective is to conduct a comprehensive review of the existing literature related to the safety aspects of autonomous vehicles. This research includes an in-depth analysis of existing knowledge and technological advances. Such comprehensive research is the cornerstone for building a deep and nuanced understanding of the multifaceted challenges, notable advances, and gaps we need to see in automotive safety. This chapter explores the design, hardware, AI-based technologies, safety concerns, and current solutions for autonomous vehicles.

Autonomous vehicles (AVs) and the associated technologies have garnered significant attention from the research community due to their potential benefits. To navigate their surroundings, AVs use sensor technologies, including computer vision, odometry, GPS, laser lights, sensors, and mapping systems. By leveraging these technologies, AVs can assess their environment, determine their location, and identify suitable routes while avoiding obstacles and adhering to traffic signs [Mukhtar, 2015][Mukhtar, 2015]. The envisioned advantages of AVs include reducing vehicle accidents, improving traffic flow and mobility, decreasing fuel consumption, driverless operation, and enhanced efficiency in business operations and transportation [Clements, 2017][Delhi, 2016][Speranza, 2018][Mahmassani, 2016].

Since the mid-1980s, numerous car companies, research institutes, universities, and industries worldwide have studied and developed AVs. To promote AV technology, various renowned competitions have been organized. For instance, the Defense Advanced Research Projects Agency (DARPA) initiated the first competition of the DARPA Grand Challenge in the USA. In this challenge, AVs were required to navigate a 142-mile desert track within a 10-hour. However, during the initial miles, all AVs failed to navigate successfully. Subsequently, the second DARPA Grand Challenge took place in 2005, featuring a 132-mile track encompassing mountain passes, numerous turns, narrow tunnels, and flat, dry lake beds [Buehler, 2007]. Out of the 23 finalists, only four AVs completed the track within the designated time. Stanford University's "Stanley" secured first place, followed by Carnegie Mellon University's "Sandstorm" and "Highlander" in second and third place, respectively. The third competition, the DARPA Urban Challenge, was held in California in 2007. AVs were required to navigate a 60-mile urban track, which included human-driven cars, simulating a realistic urban environment within a six-hour time limit [Buehler, 2009]. Among the 11 finalists, six AVs completed the challenge. Carnegie Mellon University's "Boss" claimed first place, followed by Stanford University's "Junior" in second place, and Virginia Tech's "Odin" in third place. Nevertheless, these competitions did not encompass the complex challenges encountered in everyday traffic scenarios. Following the DARPA competitions, numerous trials and competitions have been conducted by different organizations. Notable examples include ELROB, an ongoing competition since 2006 [Schneider, 2011], the AV Competition from 2009 to 2017, and the Intelligent Vehicle Future Challenge from 2009 to 2013 [Xin, 2014]. Industry and academic communities have recently intensified their research efforts in AVs. Leading companies such as Google, Argo AI, Nvidia, Mercedes Benz, Ford, Volvo, Lyft, Baidu, WeRide and Aptiv have been actively engaged in cutting-edge AV research. Additionally, esteemed univer-

sities, including Virginia Tech, MIT, Carnegie Mellon University, Stanford University,Tsinghua University, Beijing Jiaotong University, and the University of Ulm, have significantly contributed to AV research. To standardize vehicle automation, the SAE J3016 standard categorizes vehicle automation into six levels ranging from 0 to 5 [International, 2018][Favarò, 2017]. Each level possesses distinct functionalities : Level 0 signifies no automation, with the individual driver solely responsible for all operating activities. At Level 1, the vehicle is controlled by a human driver, while an automation system assists in operating tasks (e.g., Tesla Autopilot) [Linja, 2022]. Level 2 involves utilizing automated features, but human intervention is required for control and environmental aspects of the driving process (e.g., Tesla Autopilot) [Linja, 2022]. Level 3 requires the human driver to remain prepared to assume vehicle control at any moment (Automated Conditional Driving). At Level 4, the automation system can drive the car autonomously under specific conditions, but the human operator can take control (e.g., Waymo driverless cars) [Waymo, 2017]. Lastly, Level 5 corresponds to full automation, where the automation system can drive the car autonomously under all conditions, with the human operator having the option to intervene (e.g., fully autonomous Waymo driverless cars) [Waymo, 2017]. The driving decisions of an AV are classified into three levels : tactical (lane-keeping and lane-changing), operational (brake and pedal control), and strategic (routing) levels [Di, 2021]. The tactical and operational controls further involve lateral and longitudinal control categories [Di, 2021]. Many researchers and organizations strive to achieve Level 5 automation, and this chapter comprehensively addresses its associated challenges. AI serves as a crucial technology for enabling efficient autonomous vehicle functionality. AVs leverage AI and sensor technologies to mitigate risks and enhance performance. Deep learning techniques have proven highly effective in object detection, computer vision, and semantic segmentation, surpassing the previous benchmarks on various object detection datasets [Everingham, 2010][Lin, 2014]. Within the realm of AVs, deep learning methods are commonly employed for detecting pedestrians [Zhang, 2017][Chen, 2017b], vehicles [Fang, 2016][Li, 2017], road signs [Lee, 2018][Luo, 2017], and traffic lights [Bach, 2017][Behrendt, 2017]. AI techniques play a pivotal role in perception, decision-making, localization, and mapping, contributing to the overall performance of AVs [Notomista, 2017]. Perception entails the continuous scanning and monitoring of the environment by AVs through sensors, simulating human vision [Bojarski, 2016]. However, perception remains challenging for AVs, and numerous deep-learning approaches have been applied to improve perception capabilities [Bojarski, 2017]. Furthermore, AI facilitates AV decision-making processes such as automatic parking [Notomista, 2017] and path planning [Akermi, 2020]. Simultaneous localization and mapping (SLAM) is a computational challenge associated with creating or updating maps of uncertain areas [Alcantarilla, 2018], and it plays a critical role in AVs.

However, despite these immense potentials, there exist various unresolved issues related to safety, security, legal and regulatory frameworks, social acceptance, ethics, and technological challenges [Chopra, 2019][Neumann, 2016][Parkinson, 2017][Złotowski, 2017]. To ensure the success of AV systems, it is crucial to address these problems comprehensively.

As previously examined, self-driving cars encounter challenges in ensuring safety across various aspects of their operation. These challenges encompass perception, decision-making,

human-machine interaction, and the regulatory landscape. However, addressing these challenges and achieving the ultimate goal of safe autonomous driving hinges significantly on advancements in localization technologies. Localization involves determining the precise position and orientation of the autonomous vehicle within its environment. The vehicle can accurately perceive its surroundings by combining various localization techniques, such as GPS and sensor-based, using lidar, radar, and cameras. However, more than localization is required for intelligent decision-making. It needs to be complemented by data fusion, which integrates information from different sensors to create a comprehensive and reliable understanding of the environment. This seamless data fusion allows the autonomous vehicle to make informed decisions based on a holistic view of its surroundings. Therefore, we will investigate the pivotal role played by localization and data fusion in the context of autonomous vehicles.

2 Localization and Data Fusion

Accurate localization is at the heart of self-driving car safety. With precise knowledge of their position and environment, autonomous vehicles can make informed decisions and navigate safely [Vitale, 2022]. This is particularly pertinent in addressing challenges related to perception and decision-making. By delving into the state-of-the-art localization techniques, we explore how technological advancements such as GPS, IMU, and sensor fusion methods like Extended Kalman Filters provide the foundational data required for safe autonomous operation. Additionally, we analyze recent research findings and advancements in localization technologies, highlighting their pivotal role in enhancing the safety of self-driving cars. Accurate and robust localization enables self-driving cars to perceive their surroundings better, make timely decisions, and interact safely with the environment and other road users. Therefore, advancements in localization technologies are a critical component of the broader safety framework in autonomous driving.

Data fusion for self-driving cars integrates sensor data to increase reliability and minimize uncertainty in AV localization. When different sensors are combined, sensor noise and uncertainty can be substantially reduced. In addition to increasing the robustness, ensure data reliability and reduce the number of inaccurate [Banos, 2012]. Different data fusion algorithms have been proposed in the context of automated and autonomous driving. Generally, these approaches can be categorized into groups based on statistical and control functions, probabilistic methods, graph and knowledge-based methods, or data-fusing techniques based on machine learning and neural networks. [Pires, 2016] [Luo, 2011]. One of the first and classical data fusion methods for AV localization is the Kalman Filter [Kalman, 1960]. The Kalman filter algorithm iteratively estimates the state variables, such as the position and velocity of a projectile, within a noisy linear dynamical system. It achieves this by minimizing the mean-squared estimation error of the current state as it receives noisy measurements and tracks the system's evolution over time. With each update, it furnishes the most recent unbiased estimate of the system variables along with an indication of the uncertainty through a covariance matrix. This updating process is versatile and allows for effective tracking of system dynamics under varying conditions [Gelb, 1974]. In the case of nonlinear situations, several extensions of the basic KF have been

proposed, but the Extended Kalman filter (EKF) [Hoshiya, 1984] and Unscented Kalman Filter (UKF)[Wan, 2001] remain the most reliable alternatives in nonlinear environments. The former uses Taylor series expansions to linearize the state transition and observation matrices. The latter approximates the posterior probability density arising from the nonlinear transformation of a random variable. Indeed, the EKF has lower accuracy than the UKF ; however, the UKF requires additional computational time, which is unsuitable in real-time environments [Feraco, 2021]. Particle Filters (PF) are based on sequential stochastic algorithms (Monte Carlo) for Bayesian filtering to perform multi-sensor fusion problems[Wan, 2001]. However, PF requires a trade-off between estimating accuracy and computing burden by altering the number of employed particles. Recently, a neural network (NN) has been successfully employed for data fusion in robotics. In [Forechi, 2018], the authors use a deep neural network to perform the localization using a one-shot image. In contrast, in [Cattaneo, 2020], authors trained a convolutional neural network with a data set containing images and location information.

The successful adoption of neural network (NN)-based localization methods has showcased impressive results. However, it is essential to acknowledge the associated challenges. One significant limitation is the reliance on high-cost sensors, such as lidar or cameras, which can significantly increase the overall cost of autonomous vehicles [Khankalantary, 2020]. Moreover, these NN-based methods often exhibit high computational complexity, demanding substantial processing power and potentially limiting their use in real-time applications, especially on resource-constrained hardware. Additionally, these methods typically require extensive labeled training data, which can be challenging and costly to obtain, mainly when dealing with new environments or maps. Ground-truth data is essential for supervised training, posing practical challenges when transitioning to new geographic areas or navigating previously uncharted territories. While NN-based localization methods show promise, they underscore the need for alternative approaches that balance accuracy and affordability, particularly in safety-critical autonomous driving scenarios. In data fusion for autonomous vehicle (AV) localization, the Extended Kalman Filter (EKF) has traditionally been the preferred solution due to its ability to integrate sensor data effectively. However, it has limitations, most notably linearization errors [Madyastha, 2011]. An innovative alternative to the EKF is the Error-State Extended Kalman Filter (ES-EKF), which has gained prominence in various domains, including robotics and air vehicle attitude estimation. The ES-EKF adopts a distinct approach to address the challenges posed by nonlinear dynamics compared to the traditional EKF. Instead of estimating the entire state, as done in the EKF, the ES-EKF focuses on estimating the error state, typically composed of small-magnitude linear error dynamics. This emphasis on the error state enables optimal prediction and update of the error state covariance, resulting in efficient and timely localization. This approach aligns well with the imperatives of AV safety, ensuring precise localization within dynamic and unpredictable environments.

The upcoming chapter will explore the ES-EKF more comprehensively, offering readers a deeper understanding of its principles and applications in autonomous vehicle localization. By addressing the challenges posed by linearization errors and nonlinear dynamics, the ES-EKF represents a promising avenue for enhancing the reliability and safety of AV localization. In

conclusion, it is vital to recognize the profound impact of localization data quality and reliability on a self-driving car's ability to make practical, real-time decisions. Accurate localization serves as the linchpin for robust and safe autonomous navigation. This connection between precise localization and effective decision-making paves the way for a critical aspect of self-driving car safety : reinforcement learning. The subsequent discussion will shed light on how reinforcement learning plays a pivotal role in enhancing the safety and autonomy of self-driving vehicles.

3 Imitation Learning and Reinforcement Learning

The ability to make well-informed and adaptive decisions is a core competency of paramount importance. The emergence of reinforcement learning (RL) has significantly advanced the capability of autonomous vehicles in this regard. In this section, we introduce the concepts of imitation learning and reinforcement learning in the autonomous vehicle (AV) world. We then embark on an academic exploration, delving deep into the application of RL algorithms for decision-making in the context of self-driving cars. Our rigorous analysis encompasses a comprehensive overview of various RL approaches and their pragmatic implementations within the domain of autonomous driving scenarios, with a specific emphasis on auto-parking scenarios.

Reinforcement learning, situated within machine learning, equips autonomous vehicles with the capacity to autonomously learn and optimize their actions based on interactions with their dynamic environments. It follows an initial phase of imitation learning, where the AV learns from human demonstrations. Within this academic inquiry, we traverse the multifaceted landscape of RL and unveil its multifarious applications in self-driving car technology.

This inquiry involves exploring different RL methods specifically designed for the challenges faced by autonomous driving. These methods, based on strong theoretical frameworks, go beyond theory to become practical solutions that shape self-driving car technology. Our focus is particularly on automated parking situations.

Auto-parking may seem like a simple task, but it embodies the essence of autonomous decision-making. It requires accuracy, adaptability, and quick responses. In our study, we examine RL algorithms crafted to guide a self-driving car as it parks itself. Furthermore, we engage in a scholarly examination of the versatility of RL in addressing a broad spectrum of autonomous driving challenges. Our inquiry extends beyond the confines of auto-parking, encompassing complexities such as navigation through urban environments, highway driving, and collision avoidance. By focusing our academic lens on these practical applications, we offer a comprehensive scholarly account of how RL emerges as an indispensable tool in elevating self-driving cars' safety, efficiency, and dependability.

3.1 Autonomous Driving with Imitation Learning

In a sequential decision-making problem, such as autonomous driving, imitation learning, behavior cloning, and supervised learning are often used interchangeably. Imitation learning involves learning from demonstrations, where a learning agent is trained on examples of human driving and tasked with replicating the observed behavior. However, supervised learning in

autonomous driving requires substantial annotated data, which can be challenging to collect in real-world scenarios due to the vast number of potential driving situations (varying weather conditions, time of day, road types, etc.). One solution to this challenge is to use simulated data, but the issue of transferring learned behaviors from simulation to the real world must be addressed.

In 1989 Navlab introduced ALVINN, the first autonomous vehicle to utilize a neural network trained with supervised learning. ALVIN consisted of three layers : an input layer with 1217 units, a hidden layer with 29 units, and an output layer with 46 units. The input layer comprised 960 units from a 30 x 32 road image, 256 units from an 8 x 32 laser rangefinder, and one feedback unit from the output layer. This configuration enabled ALVINN to perform lane-following tasks on simple real-world roads. However, since then, neural networks have advanced significantly in size and capabilities.

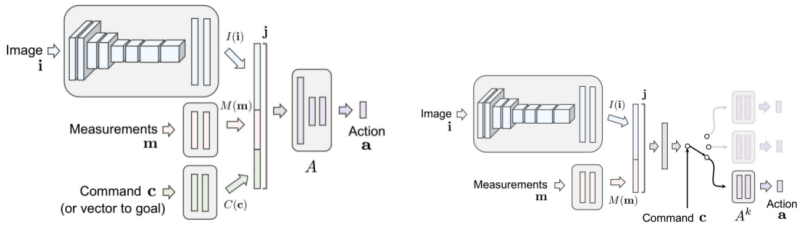
In a later study by [Muller, 2005], a mobile robot was trained to navigate off-road while avoiding obstacles. In this case, the input size was much larger than that of ALVINN. It consisted of a 149 x 58 x 6 input derived from two stereo cameras, and a convolutional neural network (CNN) with 72000 trainable parameters was employed. This approach demonstrated practical obstacle avoidance in real-world navigation scenarios.

More recently, [Bojarski, 2016] trained a CNN using a large dataset, approximately 72 hours of driving, to perform lane following. The trained model was then deployed on a real car, showcasing successful lane-following capabilities. Similarly,[Rausch, 2017] employed the idea of end-to-end control of autonomous vehicles with a driving simulator using a deep neural network policy. However, it is essential to note that both the [Bojarski, 2016] study and the [Rausch, 2017] study focused solely on controlling the steering angle during lane-following tasks and did not address controlling the vehicle's speed.

Autonomous driving is a complex decision-making process that can benefit from incorporating temporal information to enhance the robustness and consistency of the driving model. [Fernando, 2017] and [Xu, 2017] introduced the use of Short-Term Long Memory (LSTM) layers to handle memory in the context of autonomous driving. In [Xu, 2017], a single LSTM layer is employed, which merges the output of the Convolutional Neural Network (CNN) with the trajectory features (such as speed and angle) before passing through the LSTM layer. This architecture aims to capture the temporal dependencies within the input data by leveraging the LSTM's ability to retain and recall past information. In contrast, [Fernando, 2017] proposes a different approach by utilizing two distinct memory modules : the spatial memory module for the CNN output and the trajectory memory module for the trajectory features. These memory modules are maintained separately and later merged to incorporate both types of information. This architecture allows for the differentiation of memories. It facilitates the capture of long-term dependencies, which can benefit a more comprehensive understanding and decision-making in autonomous driving scenarios.

Although various approaches have been proposed for achieving end-to-end autonomous driving, most focus solely on the lane-following task and do not include any specific driving commands, such as "turn left" or "go straight." In their work introducing the CARLA simulator,

[Dosovitskiy, 2017] proposed a benchmark for evaluating driving agents on goal-directed navigation called the CARLA CoRL benchmark. This benchmark includes four tasks, such as straightforward navigation and navigation with dynamic obstacles, and evaluates agents' ability to reach their destination successfully. Six weather conditions and two towns are used to assess the agents' generalization capabilities. The benchmark highlights that both the modular pipeline and imitation learning approaches achieve satisfactory results, particularly under training conditions, while reinforcement learning performs poorly. However, both modular pipeline and imitation learning approaches show a drop in performance in unseen towns. [Codevilla, 2018] provide more detail on the imitation learning training process, including transfer to the real world. The supervised learning algorithm takes as input the current road image, a vector of measurements containing the current speed, and a high-level command indicating where to go. Their work studies two global architectures, as illustrated in Figure 3.1.



(a) command input architecture (b) branched architecture

FIGURE 3.1 – "End-to-End Driving via Conditional Imitation Learning," [Codevilla, 2018] discuss two network architectures for conditional driving. The first architecture, shown on the left, uses the driving command as input and concatenates its features with the features extracted from the image and measurements. The second architecture, shown on the right, splits the network into command-specific branches using the discrete driving command as a guide.

[Codevilla, 2018] introduced two network architectures for conditional driving. In the first architecture, the inputs (image, command, and measurement) are processed independently by separate networks : a convolutional network for the image and two fully connected networks for the command and measurement. The outputs from these networks are concatenated and fed into a control module, represented by a fully connected network. This architecture, known as the "command input" approach, suffers from the drawback of not explicitly considering the command in the algorithm's decision-making process. The authors propose a second approach called the "branched" architecture to overcome this limitation. This approach retains the image and measurement networks while the command is treated as a discrete variable. After concatenating the image and measurement, a sub-policy corresponding to the command is selected from a predefined sub-policy set. This approach has demonstrated superior performance to the classical concatenation method and has gained popularity in recent research. In [Cultrera, 2020], the imitation learning architecture is enhanced by incorporating an attention module. This module enables the model's predictions to be explained and improves the driving agent's performance compared to simple imitation learning. In [Ohn-Bar, 2020], the autonomous driving policy comprises a mixture of experts and a context embedding system. The objective is to enhance the

performance of the autonomous driving agent in situational driving scenarios. The approach involves several steps. First, a mixture of models is trained using imitation learning. This initial training phase lets the agent learn from expert demonstrations and acquire basic driving skills. Next, a context embedding system is developed using a Variational Autoencoder (VAE). The VAE extracts relevant information from the input image while reducing its dimensionality. This context embedding provides a compact representation of the driving environment and aids in capturing crucial situational cues. Finally, the policy is refined using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm. However, this refinement is applied only to the last layers of the policy, allowing for fine-tuning and adaptation to specific driving tasks. By leveraging task-driven rewards and the evolutionary nature of CMA-ES, the policy is optimized to handle various situational driving scenarios effectively. Figure 3.2 illustrates the integration of expert demonstrations context embedding, and evolutionary optimization to achieve superior performance in autonomous driving tasks, particularly in complex and dynamic driving situations.

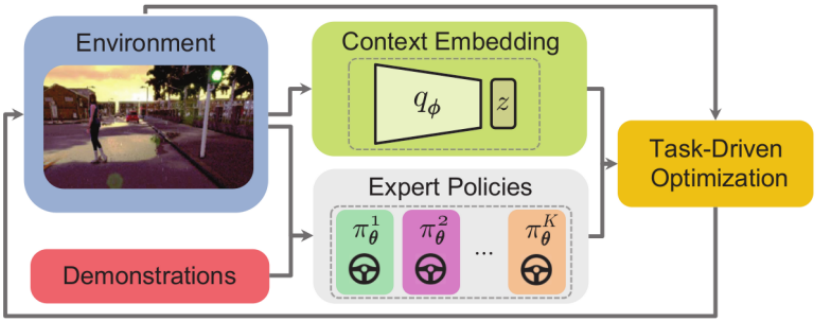


FIGURE 3.2 – Learning Situational Driving.

The model proposed in [Codevilla, 2018] was initially trained and developed for urban driving scenarios using simulation. However, to validate its performance in the real world, the model was also tested and trained on a small robotic truck deployed in a residential area. Real-world data collected from the truck and simulated data were incorporated into the training process. This approach aimed to enhance the model's robustness and adaptability to real-world driving conditions. The branch architecture employed in the model played a crucial role in ensuring that the robot successfully navigated through crossovers without missing any essential points. By considering multiple branches, the model could effectively handle complex scenarios and make accurate driving decisions. However, collecting real-world data can be a challenging and costly endeavor. Therefore, many approaches focus on transitioning from simulation to real-world environments instead of relying solely on real-world data for training. This transition bridges the gap between simulation and reality by aligning the model's performance with real-world driving conditions. In the context of this transition, the approach presented in [Müller, 2018] takes a different approach by not training the driving policy end-to-end. Instead, it leverages segmented images to extract valuable information and generate waypoints for the driving path. By producing waypoints rather than specific driving commands like steering angle and accele-

ration/braking, this approach aims to facilitate the transition to real-world driving conditions. The system architecture is shown in Figure 3.3

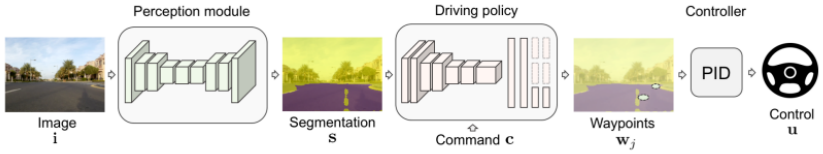


FIGURE 3.3 – The network architecture for the driving system in [Müller, 2018] consists of three main modules : the perception module, the driving policy module, and the PID controller module. The perception module is responsible for segmenting the input image, while the driving policy module computes waypoints based on the segmented image. Finally, the PID controller module computes driving commands using the computed waypoints.

The network architecture proposed in [Müller, 2018] for the driving system comprises three main components : the perception module, the driving policy, and the PID controller. The perception module, trained on the Cityscapes dataset, utilizes an encoder-decoder network with ERFNet architecture to segment the input image. The segmented map is then passed to the driving policy, trained on the CARLA simulator. The driving policy takes the segmented map and the driving instruction as inputs and generates two waypoints for each frame. This architecture is similar to the branched network used in [Dosovitskiy, 2017]. Finally, the waypoints are converted into driving commands using a PID controller. Extensive ablation studies were conducted to evaluate the model's generalization ability, comparing the proposed approach to various alternative configurations. The results demonstrated that the proposed approach outperformed the other configurations regarding generalization in both simulation and real-world scenarios. In contrast, the approach presented in [Bewley, 2019] focused solely on training a driving agent using imitation learning in a simulated environment. A translation network was employed to map real-world images to the latent space that resembles the simulated images to bridge the gap between the simulation and the real world. This enabled the agent to drive in the real world without requiring any real driving examples. The method primarily relied on imitation learning in the simulated environment to acquire the necessary driving skills. In summary, while [Müller, 2018] incorporated a perception module, driving policy, and PID controller in their network architecture and demonstrated superior generalization in simulation and real-world scenarios, [Bewley, 2019] focused on training a driving agent solely through imitation learning in simulation and utilized a translation network to adapt to real-world conditions without real-world driving examples. As a result, the agent could drive in the real world without seeing any real driving examples 3.4.

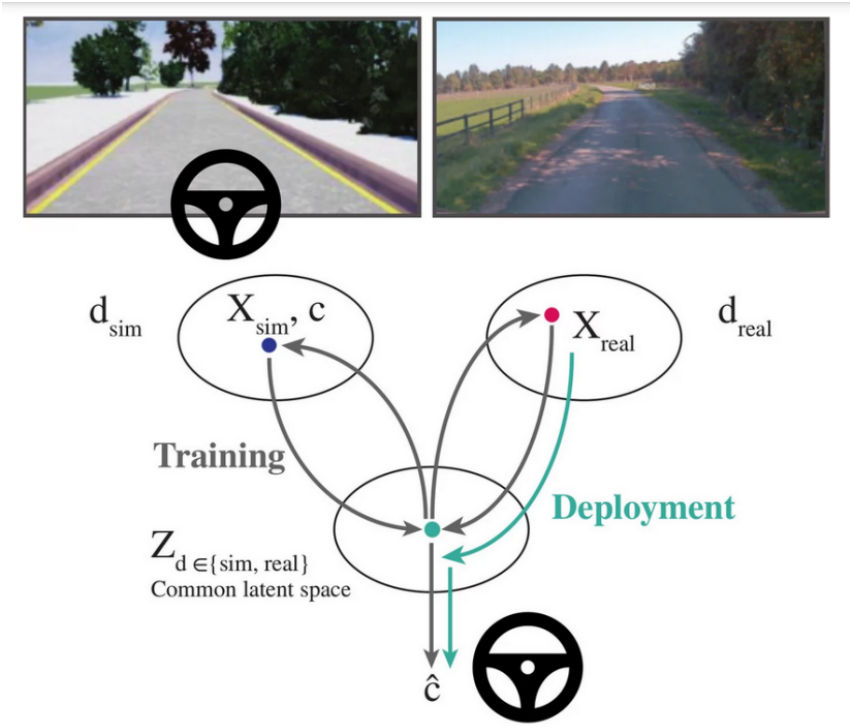


FIGURE 3.4 – To facilitate the transition from simulated to real-world driving, [Bewley, 2019] introduces a Sim2Real transition network for autonomous driving. This network utilizes a common latent space (Z) between the real and simulated images to compute the driving commands. To achieve this, two Variational AutoEncoders (VAEs) are employed, one for each domain, which are used to generate images and compute the corresponding latent representations in the common space. The Sim2Real network is trained solely on simulated data using imitation learning and then tested on real-world scenarios, demonstrating its ability to generalize and perform well even without real-world training data.

Autonomous driving can be approached through two conventional methods : end-to-end driving and modular pipeline. However, both approaches have their limitations. The modular pipeline requires optimizing each module independently, while end-to-end driving is challenging to interpret and train effectively. To address these limitations, an alternative solution is to employ intermediate driving representations, known as Direct Perception. In the work of [Chen, 2015], the authors proposed this approach, utilizing a convolutional neural network (CNN) to process input images and extract affordance information such as distances to other vehicles and lane markings. Subsequently, a straightforward controller utilizes these extracted affordances to make driving decisions, including steering commands and acceleration/braking based on factors like the distance to the preceding vehicle and its position within the lane. The CNN was trained using 12 hours of human driving data from the TORCS racing game [Wymann, 2000]. The controller's performance was evaluated on TORCS tracks, real-world data sourced from smartphone videos, and the KITTI dataset [Geiger, 2013]. The results demonstrated the effectiveness of the proposed approach in both simulated and real-world environments. Subsequent advancements in this field include the work of [Al-Qizwini, 2017], who improved upon the initial approach by using a more accurate feature extractor and reducing the number of computed affordances.

Furthermore, intermediate representation has been successfully applied to conditional driving in the CARLA simulator through the work of [Sauer, 2018]. This approach uses video input instead of frame-by-frame analysis to address conditional driving, traffic light management, and traffic sign detection. It has proven to be effective according to the CARLA CoRL benchmark.

In work presented by [Mehta, 2018], affordances are trained as additional tasks in the autonomous driving system. This involves generating two types of auxiliary predictions : visual affordances, which include information like the distance to the vehicle ahead and the lateral distance to an approaching pedestrian, and action primitives, such as slowing down, turning right, or speeding up. The action primitives act as a planner, considering inputs from the planner, visual cues, and speed to predict the appropriate action to take at any given moment. For example, if the planner input indicates a right turn, the action primitive predicts whether the vehicle should turn immediately, wait until it reaches a crossing, or slow down to stop at a traffic light. These auxiliary predictions are trained concurrently with the driving task and are utilized as input for the final block of the architecture.

The loss function in this scenario consists of both the auxiliary loss (for action primitives and visual affordances) and the driving control loss. Additionally, the affordances serve as input for the subsequent step, providing valuable information for the intermediate representation. Additional tasks are incorporated in [Hawke, 2020], although the intermediate representation is not explicitly defined. The perception module is trained to reconstruct various interpretable outputs such as semantic segmentation, depth, and optical flow, capturing temporal dynamics. However, only the encoded features are utilized for the driving tasks. The perception module is pre-trained on multiple large datasets, including KITTI [Geiger, 2013], Cityscape [Cordts, 2016], and Deeplab [Chen, 2017a]. In some experiments, three cameras are employed (forward, left-facing, and right-facing), and each image is processed independently by the perception module. The intermediate representations are subsequently merged using a sensor fusion module.

In [Kang, 2019], the research is entirely conducted in real-world settings. Data collection involves driving around Cambridge, and tests are performed with a human driver available as a backup in emergencies. The authors can identify crucial aspects of autonomous driving by utilizing supervised learning. They emphasize the importance of data distribution and employ techniques to balance the dataset during training, preventing dominant behaviors such as staying stationary or driving straight from overpowering the system. Furthermore, they observe that the system's performance improves with increased exposure to data. The authors also highlight the significance of developing a robust representation of the environment for practical, real-world driving.

In [Li, 2018], the exact auxiliary predictions, specifically semantic segmentation and depth prediction, are utilized. However, unlike the approach described in [Hawke, 2020], the perception module is trained separately and remains fixed during the training of the driving module. Learning to drive can be categorized into two main principles : understanding and acting within the environment. As discussed earlier, one method to assist the agent in understanding the environment is through intermediate representations. Another approach, LBC, as presented in citechen2020learning, involves training a privileged agent with access to expert trajectories

and detailed information about the environment's layout (such as bird's-eye view and actor positions). Subsequently, this privileged agent trains a second agent with only raw image input and no other ground truth information. This two-step approach allows for focused learning : the privileged agent can concentrate on learning to drive while having complete knowledge of the environment. In contrast, the second agent focuses on "learning to see" and comprehending the information provided through image input. This approach has achieved a perfect score on the CARLA CoRL benchmark and demonstrated effective generalization in previously unseen towns.

Despite the application of data augmentation techniques, utilizing human-provided data in imitation learning and supervised learning more broadly presents a challenge. The fundamental question arises : How can a car achieve flawless driving if it is solely trained using imperfect human driving examples ? This is where reinforcement learning becomes relevant. By employing reinforcement learning, it is possible to circumvent human bias as the system learns through self-exploration. The agent is not explicitly instructed on how to behave ; instead, it receives positive and negative rewards based on the quality of its actions. The objective is to provide the agent with minimal human-specified features to avoid bias. For instance, in the context of an autonomous car, the agent is not explicitly taught to follow road lines but instead receives a negative reward if it deviates from the road. In [Codevilla, 2019], the authors investigate and examine the boundaries of behavior cloning when applied to autonomous driving. This study sheds light on the limitations associated with behavior cloning, including :

— The term "distributional shift" refers to the disparity between the distribution of the data used for training and the distribution encountered during testing. This discrepancy is particularly evident in imitation learning for autonomous driving due to the driving task's sequential and non-i.i.d. (independent and identically distributed) nature. As a result, minor errors in previously unseen situations may accumulate over time, resulting in significant mistakes during testing.

— The work by [Codevilla, 2019] brings attention to the limitations of behavior cloning in autonomous driving. One fundamental limitation identified is dataset bias, where the collected data predominantly comprises straightforward scenarios such as driving straight. However, the autonomous agent must also learn to respond effectively in more complex situations that occur less frequently in the dataset.

— Causal confusion is a limitation that arises when the neural network discovers correlations that are artificially present only in the dataset. This can lead to misleading predictions and a lack of generalization. Additionally, distributional shift, also referred to as covariate or dataset shift, is another limitation caused by the sequential nature of the driving task, which violates the assumption of independent and identically distributed data. As a result, minor errors occur during the testing phase, and these errors accumulate, leading to more significant mistakes.

— Models trained using identical parameters demonstrate significant variability in their outcomes, particularly when facing demanding test scenarios involving numerous dynamic objects. The variance in performance can be as substantial as 40% when comparing twelve

models trained with different seeds.

The authors of this study used a conditional network similar to that of [Codevilla, 2018] with additional changes to enhance its robustness, such as incorporating a module to predict the speed and using the Mean Absolute Error instead of Mean Square Error. They also proposed a new NoCrash benchmark to test the agent's behavior under traffic conditions. The study found that the best performance was achieved with only 10 hours of demonstration data, as the dataset needed more diversity, and additional data would lead to overfitting. Besides, the dataset bias was highlighted, as the agent performed better when encountering vehicles with standard models and colors. The study also explored the variance issue, training identical models with different random seeds and noting a significant difference in results on the NoCrash benchmark. To address this issue, pretraining the CNN on ImageNet was found to reduce variance. Finally, the study identified the inertia problem caused by causal confusion, where the agent remained unmoving in many test episodes due to traffic lights or other vehicles. Although the speed prediction branch mitigated this issue, it was not eliminated. The study provides insights into the difficulties of supervised learning in autonomous driving and suggests future research directions.

Expanding the training dataset is a viable approach to mitigate the challenges posed by distributional shifts. In imitation learning, the system cannot recover from mistakes, leading to error accumulation when confronted with unseen scenarios during testing. To address this issue, researchers have explored the augmentation of their training datasets. Several papers, such as [Gandhi, 2017], [Codevilla, 2018], and [Bojarski, 2016], have focused on enlarging the datasets for autonomous driving scenarios using a single camera.

In [Gandhi, 2017], the authors intentionally crashed a drone more than 11,000 times to create an extensive crash database. A naive and random algorithm was employed to eliminate human bias. The dataset was enriched with trajectories capturing both erroneous behavior and corresponding correct behavior, enhancing the agent's robustness.

In [Bojarski, 2016] and [Codevilla, 2018], two additional cameras were introduced to capture a broader range of samples, including "bad" samples. These cameras, positioned on each side of the vehicle, provided images from different angles. The images from these additional cameras were labeled based on the camera rotation, which facilitated the inclusion of more "bad" samples during training. Furthermore, the dataset was augmented by randomly shifting and rotating the recorded images, and the corresponding ground truth was appropriately adjusted to introduce further variations during training.

In [Toromanoff, 2020], a dataset augmentation method without additional sensors. They utilize a single fisheye camera and apply a cylindrical projection to the captured image, transforming it into a format similar to a standard camera. This transformed image is then used to predict steering angles. To augment the dataset, lateral offset images are created through cylindrical projections of the fisheye camera image. While this approach offers more flexibility compared to using multiple conventional cameras for data collection, it does require post-processing of the data.

Increasing the dataset is crucial in robustifying autonomous driving systems within behavior cloning frameworks. The paper by [Bansal, 2018] introduced the ChauffeurNet model by intro-

ducing perturbations to the dataset and modifying the imitation loss function. Perturbations are applied to alter the trajectories in the dataset, enabling the agent to learn how to recover from challenging situations. Additional losses, including collision, on-road, and geometry losses, are incorporated to prevent undesired events. These losses help reduce collisions, encourage the agent to stay on the road, and ensure consistent trajectory, regardless of speed.

Moreover, the ChauffeurNet model predicts the positions of other dynamic objects in the scene, and auxiliary losses are employed to handle these predictions. While most existing works in this domain utilize a single front-facing camera, ChauffeurNet leverages mid-level inputs such as a bird's-eye view of the road map, traffic lights, dynamic object boxes, and the planned route. The model then generates a driving trajectory. This research sheds light on the limitations of behavior cloning and proposes methodologies to enhance its application in autonomous driving.

Several approaches have been explored in this section, each with challenges and opportunities. Imitation Learning faces dataset size and diversity issues, posing challenges for training fully autonomous vehicles. It also needs help to account for safety-critical aspects of autonomous driving and may not generalize well to novel scenarios. The lack of a standardized evaluation approach further complicates its assessment.

3.2 Autonomous driving with Reinforcement Learning

Although Imitation Learning offers benefits such as quick implementation and efficiency, it has several limitations. Due to the increasing complexity of problems in autonomous driving, reinforcement learning has emerged as a viable alternative. While reinforcement learning excels at learning through trial and error, its reliance on large datasets for convergence can limit its real-world application at this stage. Extensive simulation environments are often used to train these algorithms before deployment. This is especially true when it comes to urban navigation.

Despite the application of imitation learning in autonomous driving, there are limitations to its use. For example, some researchers have focused solely on controlling the lateral movement of the vehicle while leaving the speed control to a separate controller. In one approach presented by [Wolf, 2017], an end-to-end control system is utilized. In another study by [Li, 2019], the learning process is divided into two modules : a perception module that predicts track features and a control module that computes the steering angle based on these features and information about the vehicle's speed. The perception module is trained using multi-task supervised learning, while the control module is trained using reinforcement learning with DPG (Deterministic Policy Gradient). However, in both approaches, the vehicle's speed is not controlled using reinforcement learning and requires the involvement of an additional controller.

In the lane following task, other works incorporate speed control commands (acceleration/-brake) as part of a racing game. These works include [Sallab, 2016], [Sallab, 2017], [Perot, 2017], and [Jaritz, 2018]. While [Perot, 2017] and [Jaritz, 2018] utilize road images as input, [Sallab, 2016], and [Sallab, 2017] use more direct inputs, such as the position of the track borders.

[Perot, 2017] and [Jaritz, 2018] employ the A3C (Asynchronous Advantage Actor-Critic) algorithm with discrete actions and incorporate LSTM (Long Short-Term Memory) in the network to capture temporal dependencies based on the visual inputs. These agents are trained

with various driving conditions, such as weather conditions, road structures, or adherence levels, using the racing game World Rally Championship 6 (WRC6). The trained agents demonstrate the ability to generalize to unseen tracks.

Recent research has focused on tasks resembling driving in urban environments, such as lane changing and traffic light management. [Chen, 2019a] introduce Attention-based Hierarchical Deep Reinforcement Learning for lane change behaviors. They use hierarchical reinforcement learning with image inputs to train agents to change lanes. This hierarchical approach decomposes the policy into sub-policies, which allows for smoother transfer to other tasks and improves interpretability.

[Sun, 2023] introduced NavTL, a graph-enhanced bi-directional hierarchical reinforcement learning framework designed to collaboratively manage traffic signal phases and navigation directions of autonomous vehicles (AVs). However, the authors solely focus on addressing the collaborative control of traffic signals and navigation directions, thus overlooking other crucial aspects of autonomous vehicle navigation and control, such as AV accelerations and lane changes at intelligent intersections.

[Al-Sharman, 2023] presented a hierarchical reinforcement learning-based decision-making scheme tailored for automated unprotected left-turn maneuvers at unsignalized intersections. The proposed integrated scheme combines soft-actor-critic (SAC) and model predictive control (MPC) principles for high-level behavioral planning and low-level motion planning layers, respectively.

[Gu, 2023] present Safari, a state-based safety enhancement method for autonomous driving in highway scenarios, utilizing Direct Hierarchical Reinforcement Learning. Safari integrates a dynamic continuous-lattice module into policy training, considering factors like basic safety, temporal-spatial continuity, and kinematic feasibility when generating future driving goals. However, the method primarily concentrates on constrained reinforcement learning for general driving tasks, potentially neglecting extremely dangerous scenarios.

[Wang, 2024] introduces a hierarchical reinforcement learning framework aimed at enhancing mixed traffic control at intersections by integrating high-level decisions with low-level longitudinal and lateral maneuvers. Additionally, the safety mechanism is incorporated to ensure the integrity of the framework and the safety of all vehicles crossing the intersection. While the proposed framework presents promising advancements in mixed traffic control, it primarily focuses on intersections and may not address other aspects of traffic management beyond this specific context.

Similarly, hierarchical reinforcement learning is employed to manage traffic lights [Chen, 2018]. Hierarchical reinforcement learning provides better interpretability and facilitates a smoother transfer to other tasks than flat reinforcement learning. The authors first train a primitive controller individually to determine the acceleration/brake intensity based on the distance to the crosswalk. Then, they assemble these primitive controllers into a hierarchical structure to handle the traffic light scenario. However, it is essential to note that the system's state includes information about when the traffic light turns red, which may not be available in real-world driving scenarios. Nonetheless, the hierarchical reinforcement learning approach outperforms flat

reinforcement learning in the traffic light scenario.

The researchers devised a reward system to encourage the car to stay in its lane while maintaining a preferred speed of 35 km/h. Suppose a slower vehicle is detected within a 100-meter range. In that case, the agent is incentivized to overtake it after deciding to change lanes (high-level command) and calculating the appropriate driving commands (low-level command). Their training approach is based on DDPG, where the actor generates the high-level command from the current state, and the critic produces the driving command using both the current state and the high-level command. Similar to a study by [Chen, 2018], the authors in [Nosrati, 2018] compared hierarchical reinforcement learning with flat reinforcement learning and found that the former performs better in lane change tasks. They achieved even higher performance by incorporating spatial and temporal attention mechanisms and utilizing a Long Short-Term Memory (LSTM) network to process temporal information.

In [Mirchevska, 2018], the focus is solely on making high-level decisions using reinforcement learning. These decisions involve determining whether to stay in the current lane or change lanes to the right or left. The reward function is based solely on the agent's speed, with higher rewards for getting closer to the desired speed. The agent's input consists of its current speed, the relative speed of adjacent vehicles, and the distance to adjacent vehicles. Although this specific input is not available in real-world scenarios where Vehicle-to-Everything (V2X) communication infrastructure is lacking, the study demonstrates that the agent can autonomously decide to overtake a slower vehicle. Notably, both [Mirchevska, 2018] and [Chen, 2018] highlight that the agent can independently determine whether or not to change lanes.

Besides the learning mentioned above approaches, there is an increasing focus on integrating symbolic reasoning with deep learning in autonomous driving systems. Neuro-symbolic approaches combine logical and rule-based reasoning capabilities with the expressive power of neural networks. By leveraging symbolic representations, these approaches can provide decision-making processes that are interpretable and explainable. The fusion of symbolic reasoning and deep learning presents a promising direction for developing intelligent and dependable autonomous driving systems that can effectively reason about complex rules, constraints, and high-level objectives.

The approaches mentioned earlier primarily focus on reinforcement learning for autonomous driving, explicitly emphasizing road following. However, a critical challenge in urban driving scenarios is navigating intersections. Addressing this challenge, [Dosovitskiy, 2017] introduced conditional driving with reinforcement learning. They employed a combination of imitation learning and a modular pipeline to pioneer end-to-end RL training for conditional driving using vision-based input. The training process involved 10 million iterations with the A3C algorithm. However, the results were lower than those of supervised learning or the modular pipeline. During training, the RL agent completed only 14% of the episodes, whereas the supervised training achieved an 86% completion rate.

In a recent study by [Liang, 2018], a different approach called "CIRL : Controllable Imitative Reinforcement Learning for Vision-based Self-driving" was proposed. This approach combines supervised learning and reinforcement learning by pretraining the network using supervised

learning.

Following a similar approach to [Dosovitskiy, 2017], CIRL also incorporates a branched network to incorporate driving commands, maintaining a similar experimental setup. The training process of the RL agent utilizes the DDPG algorithm. The actor and critic networks are treated as separate entities, and only the actor-network leverages the pre-trained weights obtained from imitation learning as an initializer. The imitation learning component undergoes training for 14 hours of driving, mirroring the approach in [Dosovitskiy, 2017]. For RL finetuning, 0.5 million steps are performed, a smaller number compared to the 10 million steps used for RL training from scratch in [Dosovitskiy, 2017]. Remarkably, CIRL performs better than supervised and reinforcement learning training methods on the Carla CoRL benchmark.

In [Rhinehart, 2018], the authors introduced a combined supervised and reinforcement learning approach for conditional driving. This combination aims to address the limitations of using reinforcement learning alone, which can lead to unsafe behavior during training due to the suboptimal nature of the model. In contrast to the model-free approach used in CIRL, [Rhinehart, 2018] adopts a model-based approach. The proposed approach involves constructing an imitative model that prioritizes the expert's trajectory instead of considering all possible options as in traditional reinforcement learning. The imitative model learns from LiDAR points and the agent's previous positions to plan a trajectory towards a specific objective, such as a point a few meters or several meters away from the agent. A PID controller generates commands based on the planned trajectory during testing. This combined approach of supervised and reinforcement learning allows for a shorter and safer training period, mitigating the risks associated with pure reinforcement learning.

[Toromanoff, 2020] proposed IARL, a reinforcement learning approach for autonomous driving with two steps. In the first step, a supervised learning approach trains an encoder that computes affordances. These affordances include semantic segmentation, the state and distance to traffic lights, the distance to crossings, and the angle of the ego vehicle concerning the road.

In this work, the encoder is initially trained in a supervised manner to compute the affordances above. Once trained, the encoder is frozen, and its output is utilized for reinforcement learning training. The input to the system consists of four road images that incorporate temporality and the current vehicle speed. The driving instruction, such as turning left or changing lanes, is provided as a high-level command, and a branched architecture, similar to the approach used in [Dosovitskiy, 2017], is employed to incorporate this instruction. The proposed approach achieves impressive results on CARLA benchmarks and the CARLA Autonomous Driving Challenge.

[Zhao, 2022] proposed CADRE framework that employs a Cascade DRL approach for vision-based autonomous urban driving. It divides the driving task into perception and control subtasks. Initially, a Co-attention Perception Module (CoPM) pre-trains by establishing relationships between visual and control information. Then, an on-policy algorithm, PPO, trains the agent in a distributed manner. Through reward shaping and sequential model LSTM, the PPO agent achieves high success rates in complex urban environments, even in dense traffic conditions.

The works of [Michels, 2005] and [Riedmiller, 2007] mark the early stages of applying rein-

forcement learning to autonomous driving in real-world settings. In [Michels, 2005] work, an RL agent is trained to navigate in an unstructured outdoor environment while avoiding obstacles. The vision system is initially trained using supervised learning to predict depth from monocular camera input images. Subsequently, the controller is trained using reinforcement learning, taking the depth input into account to generate control commands. Similarly, [Riedmiller, 2007] presents their work where the RL controller is trained to drive a real car by following a predefined track. The input to the controller consists of the car's position and current speed, and the reward function is designed based on the proximity to the desired track, with higher rewards for staying in the middle.

These initial experiments demonstrate the feasibility of using reinforcement learning for autonomous driving in real-world scenarios. However, it is essential to note that these experiments were conducted with small robot cars. In a subsequent study by [Kendall, 2019], the scalability of RL-based autonomous driving was demonstrated. Using only images from a front-facing camera, their RL agent successfully learned to perform lane-following tasks in simulated and real-world environments. This research showcased the potential of reinforcement learning in enabling autonomous vehicles to navigate real-world scenarios.

DDPG, introduced by [Lillicrap, 2015], serves as the actor-critic algorithm in [Kendall, 2019]. To obtain optimal hyperparameters, the DDPG algorithm is fine-tuned using simulations. However, during real-world vehicle tests, a safety driver assumes control when the vehicle deviates significantly from the center of the road, marking the end of an episode. To accelerate and improve the learning process, [Kendall, 2019] incorporate a Variational AutoEncoder (VAE) as an auxiliary task that runs concurrently with the driving task. The VAE is trained to reconstruct the original image, enhancing the training efficiency. Notably, the training is exclusively performed on the actual car, without any reliance on transitioning from simulation to the real world.

It is essential to acknowledge the difficulty of directly applying reinforcement learning in real-world environments, primarily due to the trial-and-error nature of the algorithm, which necessitates the agent learning from its own mistakes through interactions with the environment. Consequently, to address safety concerns, most research in autonomous driving focuses on simulations, with only a limited number of real-world experiments, typically involving road-following tasks in simple and uncluttered environments.

Researchers are actively exploring specialized approaches to tackle the challenge of transitioning from simulation to the real world. One such approach is presented in work by [Pan, 2017a], where a translation network is proposed to bridge the gap between simulated and real-world images. This translation network facilitates the transformation of simulated images into realistic ones through a segmentation process. The network's first component segments the simulated image to extract semantic information, while the second component transforms the segmented image into a realistic representation. Initially trained as a Generative Adversarial Network (GAN), the translation network trains a driving agent using the A3C algorithm. By exclusively exposing the driving agent to realistic images, a seamless transition to real-world scenarios is achieved.

In [Agarwal, 2019] and [Chen, 2019b], the agents are trained to drive using the route information, eliminating the need for detailed route planning and simplifying the driving task compared to high-level instruction-based approaches. It is worth noting that humans can follow high-level commands while unconsciously planning their trajectory and also follow GPS-guided trajectories, indicating that these methods are not mutually exclusive.

In [Agarwal, 2019] framework proposed for autonomous driving, bird-view segmented images and route waypoints are utilized for training the agent. The segmented image is first processed through an encoder and then concatenated with the waypoint features. The output of this concatenation is used to determine the steering angle and target speed. A PID controller converts the target speed into appropriate throttle/brake commands. During training, the encoder is initially trained as part of an autoencoder, reconstructing the segmented image. Subsequently, the encoder is frozen and used in the RL (Reinforcement Learning) driving training. However, the encoder is regularly fine-tuned using the collected segmented images to improve stability throughout the training process.

This work achieved remarkable results in the Carla CoRL benchmark [Dosovitskiy, 2017] by using reinforcement learning as the sole method for driving policy. They achieved a success rate of over 90% on the navigation task. However, when it came to more dynamic navigation tasks, such as roundabouts or in unseen towns, the success rate dropped to 79% or even 60%.

[Chen, 2019b] work explores more complex driving scenarios, including roundabouts, in contrast to the simpler crossroads considered in [Agarwal, 2019]. Like [Agarwal, 2019], their approach consists of two separate modules : a perception module with latent encoding and a control module with the RL agent. They train the encoder using supervised learning with a Variational AutoEncoder (VAE) and then freeze it during RL training for driving control by the agent. The input to their system includes a preview image of the map containing the routing information, the ego vehicle, and surrounding objects. Their approach also achieved high scores in the Carla CoRL benchmark, with a success rate of over 90% on the navigation task. However, similar to the previous work, the success rate dropped to 79% or even 60% in unseen towns during the dynamic navigation task.

While it is possible to input road information through GPS, obtaining the positions of surrounding objects and generating semantic segmented bird' s-eye view images like those used in Agarwal's study is a more complex task in real-world scenarios, requiring additional data processing before implementing such algorithms.

3.3 Automated Parking System

Early approaches focused on path planning for identified parking spaces in the initial stages of automated parking system (APS) development, followed by path tracking execution. Path planning methods included geometric techniques such as Reeds–Shepp (RS) curves [Fraichard, 2004], B-spline curves [Gómez-Bravo, 2001], and n3-splines [Lini, 2011], as well as sampling methods like Rapidly-exploring Random Tree (RRT) [Han, 2011] and target bias RRT [Zheng, 2018], alongside numerical optimization methods. Correspondingly, path tracking solutions encompass Ackerman steering model-based open-loop control [Thrun, 2006] and vehicle dynamics

model-based closed-loop control [He, 2019] to ensure precise parking execution.

It is imperative to recognize that irrespective of the chosen path planning or tracking method, the non-linear dynamics inherent to vehicles necessitate accurate execution adjustments during automated parking tasks.

However, with the advent of deep neural networks and RL algorithms, novel approaches emerged, aiming to emulate human-like end-to-end models for APS execution. For instance, Folker et al. [Folkers, 2019] explored parking space identification using two reward functions ("Driver" and "Stopper") within a state space that encompassed perception maps, vehicle position, speed, orientation, target position, and orientation. The Proximal Policy Optimization (PPO) algorithm was employed for agent training, with positive feedback tied to target proximity and negative feedback associated with high wheel angles or excessive agent speed. Notably, this approach lacked explicit control over braking actions, resulting in binary stop-and-go behavior with no gradual deceleration and acceleration akin to real-world scenarios. Furthermore, constraints such as inclination angles and deviations from the parking spot should have been addressed.

Zhuang et al. [Zhuang, 2018] proposed a model divided into three stages : "get closer," "plan and park," and "adjust," exclusively focusing on parking maneuvers. The model leveraged a PPO algorithm with Long Short-Term Memory (LSTM) neural networks trained through curriculum learning. Nevertheless, this approach's limitation was that it only considered wheel angle adjustments in the agent's action space, neglecting acceleration, braking, proximity to the parking spot, or angular deviations. Additionally, it did not account for obstacle presence.

In works by Zhang et al., [Zhang, 2019] and [Zhang, 2020], the DDPG algorithm was applied to APS through a path planning and tracking framework, segmented into "get closer" and "park" steps. Parking spot detection and angle estimation were facilitated using four cameras, with minor distinctions in tracking techniques. However, both approaches shared a common drawback of failing to address parking spot exploration, relying on separate policies for longitudinal and lateral control.

The study by Du et al. [Du, 2020] centered on the Deep Q-Network (DQN) and Deep Recurrent Q-Network (DRQN) algorithms, focusing on intelligent parking scenarios where parking spot information is communicated to the car for path planning. Notably, this work should have included the consideration of braking actions and the distance to the parking spot, which are crucial elements in real-world scenarios.

Feher et al. [Fehér, 2019] and [Fehér, 2020] developed a model employing the DDPG algorithm, addressing longitudinal and lateral trajectory planning and tracking. Another DDPG-based approach was proposed by Bejar et al. [Bejar, 2019], but it solely focused on parking tasks, providing limited details that hindered requirement assessment. Lastly, Thunyapoo et al. [Thunyapoo, 2020] implemented an auto-parking framework utilizing the PPO algorithm, focusing solely on parking execution without path planning considerations.

It is important to note that the discussed works rarely detail how vehicles come to a stop within the parking space and manage their speed during parking maneuvers, encompassing acceleration, deceleration, and braking actions. The criteria for episode success varied among

these works, with some defining success as strict alignment within the parking space and others considering approximate positioning as successful. Additionally, agent speed was often restricted to a fixed value due to the absence of a dedicated braking system. Simulation environments and software details were only sporadically provided.

Moreover, some works inconsistently adhered to the recommended normalization of the state space and reward function, as advocated in [Schulman, 2017]. Notably, papers focusing solely on Deep Reinforcement Learning (DRL) algorithms rarely conducted comparative analyses with other DRL techniques, a facet that this study addresses comprehensively. Furthermore, the simulation environments in most papers were simplified and needed more standardization, posing challenges in direct comparison. Additionally, these simulations often overlooked the physical constraints and behaviors inherent to real-world vehicles.

Safety is of paramount importance as we progress in autonomous driving technology. Addressing safety concerns involves tackling challenges related to uncertain and dynamic environments, handling system failures, implementing fault-tolerant mechanisms, and making ethical decisions. Safety assurance and verification methods are crucial in testing and validating autonomous systems to ensure they meet rigorous safety standards. As the deployment of autonomous vehicles on public roads increases, regulatory and legal aspects also come into play, necessitating careful consideration and adherence to established guidelines and protocols.

4 Safety Challenges in Autonomous Systems

Safe end-to-end autonomous driving has been an active research topic in recent years. Most end-to-end systems now fall under one of three paradigms : rule-based, imitation learning, and deep reinforcement learning. Despite much research on end-to-end urban driving, this chapter is concerned explicitly with safe end-to-end urban driving. While many previous studies have explored various aspects of urban driving automation, the safety of passengers and other road users must remain a top priority.

4.1 Rule-based Methods

The first studies adopting AV rules are [Montemerlo, 2008] and [Furda, 2011]. Authors in [Montemerlo, 2008] adopted a 13-state finite state machine (FSMs) to select maneuvers for Junior in DARPA (the Defense Advanced Research Projects Agency) and implemented a behavior rule to switch between diverse driving states. In [Furda, 2011], FSM is integrated with multi-criteria decision-making for the execution of each driving maneuver. However, they did not consider any safety features. In [Collin, 2020], the authors presented an AV decision-making system that respects safety considerations and traffic laws following ISO/PAS 21448 [Radlak, 2020]. [Xiao, 2021] developed a system that imposes the safety of AV by validating the rule priority structure for each decision. [Lin, 2022] proposed a framework to repair the AV trajectory following traffic rules formalized in temporal logic. [Kothawade, 2021] proposed a common sense reasoner using ASP for AV end-to-end decision-making by simulating the mind of a human driver. The rule-based system played a pivotal role in ensuring the safety of an AV decision-making system.

However, all the possible scenarios must be manually anticipated and encoded, particularly in urban driving, where the traffic environment is highly dynamic. Indeed [Collin, 2020],[Radlak, 2020],[Xiao, 2021],[Lin, 2022] and [Kothawade, 2021] presented end-to-end systems while emphasizing safety features. However, such a system relies on a fixed set of rules and needs more flexibility to adapt to unexpected scenarios. Also, this can result in reduced performance and even failure in complex and dynamic driving situations. Additionally, rule-based systems can be challenging to maintain and update as they require manual adjustments for each new scenario.

4.2 Safety in Reinforcement Learning

Learning-based methods aim to learn the driving policies without human intervention. However, ensuring their safety is a significant concern. Several works tried to overcome this issue by bypassing the functional requirements of the neural network [Li, 2021b][Filos, 2020] or verifying the safety of each action in post-hoc methods [Phan-Minh, 2022][Krasowski, 2022].

In [Alshiekh, 2018], a shield is employed to proactively prevent the agent from taking actions that could potentially result in safety breaches during the exploration phases of both model-based DRL [Yang, 2023] [Jansen, 2020] and model-free DRL [Kimura, 2021]. This shield is a logical component designed to carefully consider safety constraints during the exploration of an environment[Yang, 2023], while limiting the agent's actions to avoid catastrophic failures during the learning process [Leurent, 2020].

Shielding strategies encompass rejection and suggestion-based approaches, often grounded in formal verification methods, which offer robust safety assurances compared to other safe exploration techniques [Yang, 2023]. In a related study [Kimura, 2021], a shielding technique based on logical neural networks (LNNs) [Riegel, 2020] recommends safe actions and avoids unnecessary ones. Their research demonstrates that leveraging external knowledge through LNN to suggest the appropriate actions represents a promising approach to shielding. This approach reduces the need for extensive training trials and ensures safety throughout the learning process.

However, synthesizing an offline shield for discrete-event systems demands an exhaustive, upfront safety analysis for all potential state-action combinations, resulting in exponential complexity in state and action dimensions [Alshiekh, 2018] and becoming over-restrictive [Jansen, 2020]. Online shielding lacks worst-case computation time guarantees, potentially allowing the agent to reach the next decision state before the shield determines which action to block. It is suitable in scenarios where alternative actions, like "waiting," can be taken.

Some works designed hierarchical methods that learn high-level policies and extract low-level policies by imitating an expert with an additional layer of safety for each option. [Li, 2021a] and [Jamgochian, 2022] proposed HRL with safety constraints. However, it has only been tested on roundabouts and did not include other features of urban driving that we have considered in our study. To overcome these limitations, we propose combining HRL with rule-based systems. HRL can provide the flexibility and adaptability to navigate complex and dynamic urban environments. In contrast, rule-based systems provide a set of predefined rules that can be applied in critical situations to ensure safety. By combining both approaches, an AV can learn from its experiences in a hierarchical manner and make decisions based on its learned policies. However, in

a dangerous or unpredictable situation, the vehicle can switch to a rule-based system to reduce exploration and ensure safety. This hybrid approach can balance exploration and safety, making autonomous urban driving safer and more efficient.

Having explored the different aspects of autonomous vehicle projects, ranging from localization and data fusion to imitation learning, reinforcement learning, neuro-symbolic approaches, and safety considerations, it is crucial to summarize the essential findings and reflect on the emerging trends and future directions. The upcoming chapter will elucidate the tools and simulation environment employed for self-driving cars.

5 Tools : Available Simulation Environments

This section outlines a list of available environments that can be utilized for working purposes. The environments have been categorized into games, indoor environments, and driving environments - comprising simulators or datasets. It is important to note that this list is non-exhaustive.

5.1 Games

Regarding reinforcement learning based on visual input, games are often chosen as the environment due to their simplicity and efficient agent-environment interaction. However, these games typically offer a limited range of actions and relatively basic scenarios. The Atari games [Bellemare, 2013] were among the first games used in this context, with Enduro being particularly relevant to driving.

There are also game environments specifically designed for AI research, such as ViZDoom [Wydmuch, 2018] and DeepMind Lab [Beattie, 2016]. DeepMind Lab provides a maze-based environment, while ViZDoom is a first-person shooter game with a customizable 3D environment and first-person view. ViZDoom, unlike DeepMind Lab, requires the agent to navigate and engage with enemies simultaneously. However, it is essential to note that these virtual game environments lack realistic rendering and actions that can be directly transferred to real-world scenarios.

5.2 Indoor Environments

In our initial exploration of navigation, we also considered indoor environments, which led us to examine various indoor simulators and datasets. These resources play a crucial role in facilitating research on indoor navigation.

Two examples of indoor datasets containing many images and offering semantic segmentation are SceneNet RGB [McCormac, 2016] and the SUNCG dataset [Song, 2017]. While both datasets are synthetic, SceneNet RGB stands out for its photorealistic rendering.

Indoor simulators such as Minos [Savva, 2017], House3D [Wu, 2018], and HoME [Brodeur, 2017] allow agents to navigate within virtual houses, providing a controlled environment for tes-

ting navigation algorithms. These simulators offer different features and capabilities to support diverse research requirements.

Another notable indoor simulator is AI2 THOR [Zhu, 2017 ; Kolve, 2017], which consists of 3D photorealistic indoor scenes. In AI2 THOR, agents can interact with the environment by moving objects or pushing them, enabling more complex and interactive navigation scenarios.

5.3 Driving Environments

With the advancement of autonomous driving technology, a wide range of learning environments has emerged, falling into the categories of datasets and simulators. Among the unique datasets, the Kitti dataset [Geiger, 2013] holds a prominent position in vision-based driving tasks. It comprises an extensive collection of real-world images captured in Karlsruhe, Germany. It provides ground truth data for various tasks such as stereo vision, optical flow, visual odometry, 3D object detection, and 3D tracking. The dataset includes challenging scenarios encountered on the road, including median strips, narrow roads with parked cars, and a tramway line.

Several derivatives of the Kitti dataset have been introduced to cater to specific needs. For example, the work by [Menze, 2015] extended the dataset to include dynamic scenes, capturing the dynamic nature of objects in the environment. Kitti-Road [Fritsch, 2013] was explicitly developed for road and lane detection tasks, focusing on providing accurate annotations in those areas of interest. Another unique dataset, the Virtual Kitti dataset [Cabon, 2020], serves as a virtual extension of the Kitti dataset. It replicates scenes from the original dataset in a virtual environment, offering different weather and lighting variations. Ground truth data for depth maps, semantic segmentation, optical flow, and object detection and tracking are provided in this virtual dataset. Collectively, these datasets contribute to advancing research in autonomous driving by providing valuable resources for training and evaluation purposes.

The SYNTHIA (SYNTHetic collection of Imagery and Annotations) dataset [Ros, 2016] is a comprehensive photorealistic driving that includes semantic segmentation and depth prediction. It comprises vast data, with over 200,000 labeled images captured under different lighting conditions, weather conditions, seasons, and diverse urban environments such as European towns, modern cities, highways, and green areas.

On the other hand, cityscape [Cordts, 2016] is a real-world urban dataset specifically designed for semantic segmentation tasks. It comprises images collected from various cities across Germany, providing a realistic representation of urban environments.

Waymo has recently released its dataset [Sun, 2020], which was gathered from three cities in the United States : San Francisco, Phoenix, and Mountain View. This dataset offers a wealth of camera and LiDAR data, providing a rich source of real-world images for autonomous driving research.

Additionally, several other datasets are available for autonomous driving research, each with unique characteristics and focus. These include CamVid [Brostow, 2009] and Torontocity [Wang, 2016a], which are suitable for semantic segmentation tasks. The Robotcar dataset [Maddern, 2017] provides a diverse set of data collected from a sensor-rich autonomous vehicle operating in

urban environments. The Audi Autonomous Driving Dataset (A2D2) [Geyer, 2020] offers a large-scale dataset with various sensor modalities, including camera, LiDAR, and radar. Appoloscape [Huang, 2019] is a comprehensive dataset designed for autonomous driving, covering diverse scenarios such as urban, highway, and nighttime driving. Lastly, the BDD100K - DeepDrive Dataset from Berkeley [Yu, 2020] contains a vast collection of diverse driving scenes captured from many dashcams.

While datasets provide valuable insights into autonomous driving, they cannot interact with the environment. Driving simulators offer a solution to address this limitation. One such simulator is Javascript Racer [Gordon, 2012], a simple driving game that can run on any computer. Although it lacks photorealistic graphics, its simplicity makes it accessible for basic driving simulations.

Another widely used driving simulator in research projects is Torcs [Wymann, 2000], which offers a more realistic design than Javascript Racer. Torcs provides a richer driving experience and a platform for testing and evaluating autonomous driving algorithms.

Additionally, a simulator called highway-env [Leurent, 2018] emulates the dynamics of a highway environment with multiple lanes and vehicles. It is specifically tailored for the development and assessment of decision-making algorithms for autonomous vehicles.

Furthermore, VISTA [Amini, 2022] emerges as an open-source, data-driven simulator tailored for the multi-sensor perception of embodied agents. Utilizing real-world data, VISTA constructs ego-agent viewpoints as they navigate through dynamically evolving trajectories within the environment. Equipped with efficient and high-fidelity sensors, this simulator facilitates online perception learning, evaluation, and seamless deployment from simulation to real-world scenarios.

For urban driving simulations, two popular open-source options are Carla [Dosovitskiy, 2017] and Airsim [Shah, 2018]. Airsim, developed by Microsoft AI & Research, supports drone flight and car driving simulations. On the other hand, Carla is an urban driving simulator supported by Intel and Toyota. Both simulators offer photorealistic graphics and a wide range of features.

These simulators provide complex urban environments with realistic lighting and weather conditions. They also support multiple sensors, including cameras, LiDARs, radars, and semantic segmentation. This allows researchers to simulate various driving scenarios and test their algorithms in a controlled and reproducible environment. Simulators play a crucial role in developing and evaluating autonomous driving systems, enabling researchers to iterate and improve their algorithms before deploying them in real-world settings.

In this section, we have introduced various learning environments that range from simple games to driving simulators. The literature offers a wide variety of choices for these environments. This chapter's conclusion aims to provide a comprehensive overview of the state-of-the-art approaches in autonomous vehicle projects, highlighting their potential impact on society while considering ethical, societal, and technological considerations.

6 Synthesis and Research Trails

In this literature review, we have explored various frameworks for autonomous driving and navigation, which employ supervised learning, reinforcement learning, or a combination of both. It is worth noting that only a few approaches focus on a complete image-based end-to-end solution using reinforcement learning, and the few that do are limited to lane following [Kendall, 2019][Pan, 2017b]. For more complex driving tasks, such as urban driving, the state space is simplified by using segmented bird view images [Agarwal, 2019], pre-training the agent with supervised learning (CIRL [Liang, 2018]]), or pre-training a portion of the network using supervised learning to predict affordances, which are then used as inputs for an RL agent [Toromanoff, 2020]. The agent did not directly learn from images and reinforcement learning in these cases. Therefore, this thesis aims to investigate the feasibility of end-to-end learning using reinforcement learning. We can observe from the literature a widespread utilization of intermediate affordances or representations, both in the context of imitation learning (IL) [Sauer, 2018][Chen, 2015][Al-Qizwini, 2017][Mehta, 2018] and reinforcement learning (RL) [Toromanoff, 2020][Li, 2019]]. These affordances can be categorized as either implicit or explicit. When explicit, affordances are typically driving-specific features (such as distance from the preceding car, center of the road, etc.) that enable more transparent learning by mitigating the black-box effect of neural networks. However, humans generate these affordances, so their selection could bias the agent. For instance, is it necessary to predict that the car ahead is at a distance of 5.5 meters, or is it adequate to know that there is a car ahead and determine whether it is close or far away ? On the other hand, intermediate representations are implicit affordances generated during training for a specific task (such as semantic segmentation or reconstruction of the original image using a VAE). The agent utilizes this intermediate representation for autonomous driving.

The literature shows various approaches for training affordances or intermediate representations, including training them simultaneously with the driving task (known as auxiliary tasks) as in [Mehta, 2018][Hawke, 2020] or training them separately in a supervised way, which is more common [Li, 2018][Li, 2019][Chen, 2019a][Toromanoff, 2020][Sauer, 2018][Chen, 2015][Al-Qizwini, 2017]. In the latter case, the affordances can be used by a PID controller or as input for a driving agent, which helps reduce the state space, especially in RL-based driving[Toromanoff, 2020]. However, decoupling perception and control could negatively impact the agent's performance and robustness since there is no direct mapping between image perception and driving control. To overcome this limitation, end-to-end training using auxiliary tasks has been explored in navigation tasks such as maze-solving [Jaderberg, 2016][Mirowski, 2016]. This approach is promising but underexplored for complex tasks like autonomous driving with RL in an urban environment.

This thesis uses reinforcement learning with localization, raw image input, and continuous actions for conditional urban driving. Continuous actions were chosen to make the system more realistic and not rely on predetermined discretization. However, due to the challenging nature of the driving task, we will investigate methods to speed up learning and improve performance,

including incorporating auxiliary tasks. Given the trial-and-error nature of reinforcement learning and the importance of safety, we will conduct our work in simulation. Specifically, we will use the Carla simulator [Dosovitskiy, 2017] due to its urban driving environment, various sensor options, and availability of evaluation benchmarks and challenges. It should be noted that only some existing approaches use reinforcement learning, and none of them use end-to-end RL with RGB visual input.

Carla provides a suitable environment for urban driving simulations due to its diverse sensors and environmental variability. Additionally, it offers an autopilot mode and two evaluation benchmarks (CoRL and NoCrash) that researchers can use to compare their algorithms. Carla also hosts the Carla Challenge [CARLA Autonomous Driving Challenge 2019], which evaluates the ability of algorithms to handle complex driving situations, such as intersections, overtaking, and multi-lane driving. Furthermore, Carla has a large and growing community ; some have created maps, including interactive traffic scenarios in [Osiński, 2020].

In the following chapter, we will present our initial work, which revolves around attaining precise real-time localization of the kart through the utilization of cost-effective sensors.

Chapitre 4

Localization of Autonomous Vehicle with low-cost sensors

Contents

1 Introduction

The world has experienced many autonomous vehicles (AVs) in recent years. AV will play a significant role in preventing death and injuries caused by car accidents. However, numerous challenges exist before AV can be introduced in the market and sold as a level five fully automated car [Yaqoob, 2019]. AV architecture comprises five functionalities : localization, perception, planning, control, and system management [Jo, 2014]. Nonetheless, localization is essential functionality for an AV [Reid, 2019] since performing a safe trip from point A to point B while recognizing its surroundings is crucial. This will allow the vehicle to avoid static and dynamic obstacles and make different decisions, such as braking, accelerating, stopping, or lane changing. Nowadays, AVs have different exteroceptive sensors (GPS, LiDAR, CAMERA, RADAR, etc.) and proprioceptive sensors (vehicle motion sensor, IMU, etc.). However, none of these sensors can perform a complete and error-free AV localization. For instance, LiDAR sensors achieved promising results, but they are expensive compared to other sensors, the camera does not perform well when the weather is not clear [Liu, 2021], the GPS suffers from signal blockage [Kanhere, 2018], IMU accuracy degrades with time, etc. Furthermore, the computational complexity required by some of these sensors makes them hard to use on an embedded system for real-time localization. In this sense, data fusion of different sensors can be used in AV localization. Multi-sensor data fusion involves integrating data collected by multiple sensor devices to provide reliability and robustness, decrease uncertainty, and increase performance accuracy in real-time [Bounini, 2016].

Sensor networks, robotics, video image processing, and self-driving vehicles are just a few fields where data fusion systems are used today [Khaleghi, 2013]. These features are difficult to eliminate using a single sensor data [Osório, 2019]. Several techniques have been developed to fuse different data modalities in real-time for heterogeneous or homogeneous sensors [Nweke, 2019], such as Kalman Filter (KF), Extended Kalman Filter (EKF), and Error State Extend Kalman Filter (ES-EKF). In addition, methods using probability estimation techniques [Nweke, 2019], machine learning, and neural networks have also been proposed. Although these approaches have been successful, they suffer from many weaknesses : (i) some of them are limited to indoor robotic applications ; (ii) when applied to AVs, the vehicles are equipped with expensive sensors such as cameras and LiDAR that require powerful computational units not suitable for embedded systems [DAlfonso, 2015][Marković, 2022][Shaukat, 2021] ; (iii) when conducted on a simulator such as Carla, sensors' data are always reliable and noise-free [Castillo-Torres, 2021].

To overcome these limitations, the main contributions of this chapter are threefold : 1) we present the absolute localization for autonomous vehicles by integrating two low-cost sensors, namely GPS and IMU. Our approach includes a high-precision calibration method for the IMU sensor, noise reduction through a low-pass filter, and using ES-EKF to integrate IMU data with GPS coordinates ; 2) we have verified that ES-EKF outperformed EKF on the Carla simulator and achieved better localization results ; 3) finally, we performed diverse real-time urban driving simulations utilizing the ICM 20600 IMU and Quectel L80-M39 GPS executed in a FreeRTOS real-time environment, the STM32 Nucleo [1]. These simulations yielded a 92% accuracy within

1. https ://www.st.com/en/evaluation-tools/stm32-nucleo-boards.html/

a confidence level of 0.5m distance threshold. The execution time, with an execution time of approximately 20 μs in real-world industrial applications.

The remainder of this chapter is structured as follows : Section 2 focuses on the chapter's contribution, Section 3 provides a comprehensive overview of all experiments conducted, and lastly, Section 4 concludes and offers perspectives for future considerations.

2 System Overview

The proposed approach for achieving accurate vehicle localization through the fusion of GPS and IMU data is detailed in this section. Fig.4.1 visually represents the pipeline process involved in the localization system. The methodology comprises three key steps, each contributing to the overall accuracy of the localization process.

The first step focuses on IMU calibration, which is crucial for enhancing the precision of angular velocity and linear acceleration measurements obtained from the IMU sensors. Through a rigorous calibration process, offsets and biases inherent in the IMU sensors are carefully measured and compensated for. By rectifying these errors, the accuracy of IMU measurements is significantly improved, laying the foundation for reliable and precise vehicle localization.

In the second step, a low-pass filter is employed to reduce the noise in the IMU data. Given that IMU sensors are prone to noise, which can introduce measurement errors, the low-pass filter effectively smooths the data by averaging old and new values with controlled weighting (α). This noise reduction process further refines the accuracy of IMU measurements, contributing to the overall accuracy of the localization system.

The final step in the pipeline involves the fusion of calibrated IMU and GPS data to estimate the ego vehicle's rotation, velocity, and position. This estimation process is achieved by applying the Error State Extended Kalman Filter (ES-EKF). The ES-EKF is chosen for its robustness in handling the non-linearities associated with vehicle motion. By integrating information from IMU and GPS data, the ES-EKF provides reliable and accurate estimates of the vehicle's motion state, ultimately enhancing the accuracy of the vehicle localization system.

In the forthcoming sections, each part of the proposed method will be elaborated in detail. This comprehensive analysis will illuminate the IMU calibration process's technical complexity, the low-pass filter's noise reduction function, and the mathematical principles underlying the ES-EKF filter for fusion and estimation. By dissecting each component, the analysis aims to offer a deeper understanding of the approach's implementation and its impact on achieving accurate vehicle localization. Through this detailed examination, potential challenges and limitations will also be explored, providing valuable insights for further research and improvements in the localization system.

2.1 IMU calibration

The calibration removes the offsets for the accelerometer and the gyroscope. It is performed stationary on a flat surface where the orientation is precisely known. This known orientation, often referred to as "ground truth".

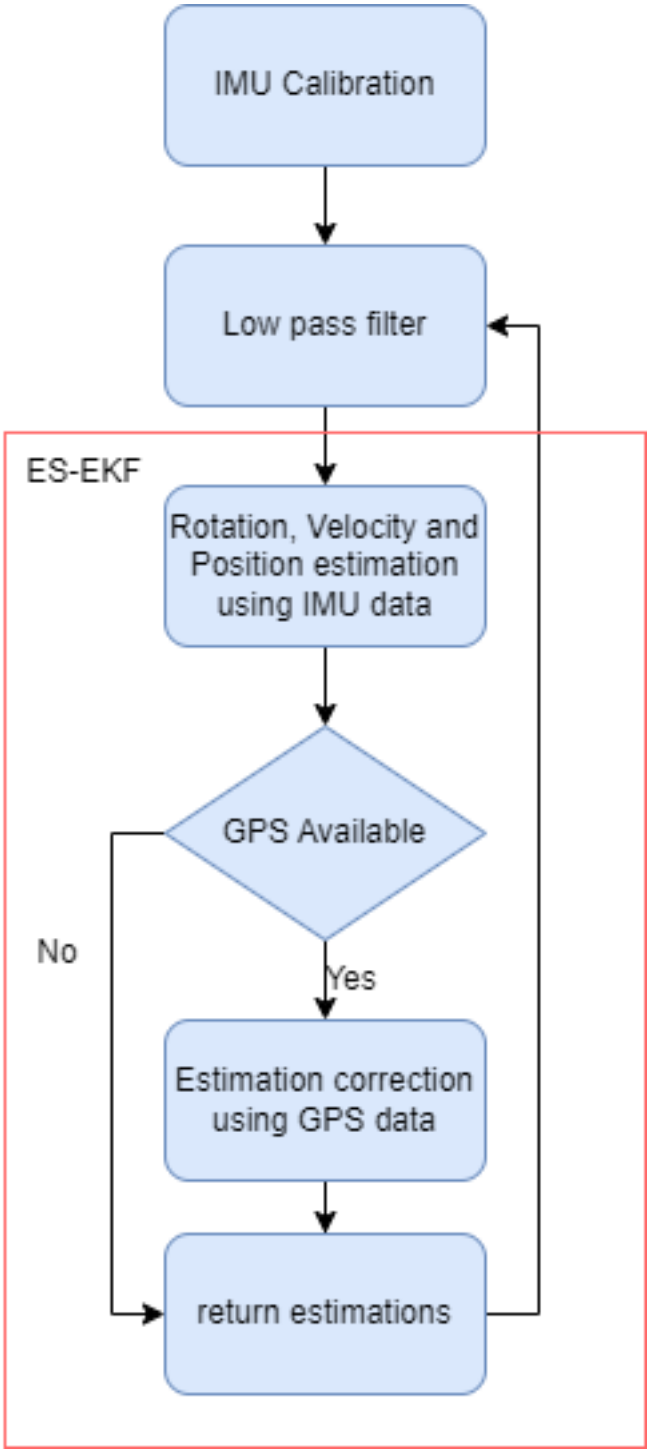


FIGURE 4.1 – System diagram

Ground truth refers to a known and accurate reference point against which the sensor measurements are compared. Since the sensor is placed on a perfectly flat surface (0 degrees of tilt), this established plane serves as the undisputed, true representation of the sensor's orientation. By comparing the sensor readings during calibration with this known ground truth (0 degrees), any biases or offsets in the accelerometer and gyroscope can be identified and corrected. This ensures that the sensors provide more accurate measurements during operation.

2.2 Low pass filter

A $low-pass\ filter$ is an electronic filter that keeps the low-frequency signals unaffected while reducing the amplitude of the signals above the cutoff frequency. The low-pass filter is a series of resistors, a capacitor that absorbs high-frequency signals and blows low frequencies, allowing them to pass through the load in parallel to the capacitor. The cutoff frequency is determined by the time constant $t : t = rc$, where r is the resistance and c the capacitance [Bozic, 2018]. This time constant is directly related to the cut-off frequency $f_C = 1/(2\pi t)$. Generally, a higher time constant t corresponds to a lower cut-off frequency f_C.

We have applied the low pass filter to the accelerometer to minimize the unwanted effect of noise. A simple way to apply such a filter is to take a percentage of the old value with the percentage remainder of the newly measured value :

$$y[n] = \alpha \times y[n-1] + (1-\alpha) \times x[n], \tag{4.1}$$

where y represents the filtered output of the system, and x represents the accelerometer data. α is defined as :

$$\alpha = \frac{t}{t+dt} \tag{4.2}$$

The interrelation between the equation $t = rc$ and the filtering equation is established through the determination of the parameter α. In the filtering equation, α is derived as $\frac{t}{t+dt}$, wherein t represents the time constant of the filter and dt denotes the sample period. By manipulating the equation $t = rc$, we can express t as a function of r and c. Substituting this expression for t into the calculation of α engenders a relationship between the intrinsic properties of the RC circuit, namely resistance and capacitance, and the filtering parameter α. This correlation enables the formulation of the filter design based on the characteristics of the RC circuit and the desired filtering attributes.

2.3 ES-EKF

The ES-EKF is a minimal filter that guarantees an effective linearization at all costs. The Jacobians matrix is easily calculated since it is based on the error state, which is small compared to the nominal state, as in EKF [Madyastha, 2011]. KF corrections are small in ES-EKF compared to EKF or UKF, which results in a lower prediction rate. We have conducted some experiments and proved that ES-EKF outperforms EKF compared to the Carla simulator in a perfect scenario, where the sensor data are noise-free. This experiment will be detailed in the next section.

The architecture of ES-EKF in our setup is loosely coupled, allowing for efficient integration of sensor data from different sources. The IMU sensor installed on the car will produce data at a high rate (*100hz*). This data is collected into a large non-linear nominal-state X without any noise and uncertainties and is used to calculate the measurement matrices. However, the accumulated errors will be collected in a smaller one, the linear error-state δx, and predicts the Gaussian estimate from δx only, since GPS data are slower compared (*10hz*) to IMU. Once the system has captured the GPS sensor data, the correction is completed by ES-EKF, which delivers a posterior Gaussian estimate of δx. Henceforth, δx means are inserted into X. δx is reset to zero, its covariances matrix is updated, and the whole procedure is restated until no further data is gathered.

In our ES-EKF architecture, synchronization between the GPS and IMU sensors is crucial for accurate state estimation. This synchronization is achieved through the alignment of their respective timestamps. The IMU sensor, operating at a high rate of 100 Hz, and the GPS sensor, operating at a slower rate of 10 Hz, each provide data with timestamps that are synchronized to a common reference point.

By aligning the timestamps of both sensors, we ensure that the measurements from each sensor correspond to the same moment in time, facilitating the integration of data from multiple sources into the state estimation process. This synchronized data fusion enables ES-EKF to effectively leverage information from both sensors to produce more accurate and reliable estimates of the vehicle's state, even in dynamic environments with varying rates of sensor data acquisition.

Additionally, the initialization of the covariance matrix is a critical step for accurate state estimation. To ensure an appropriate initialization, we utilize values obtained from the sensors' datasheets. These datasheets provide specifications regarding each sensor's noise characteristics and performance metrics. Furthermore, to handle matrix conditioning due to handling errors rather than values, we have incorporated regularization techniques. Specifically, we add a small amount of diagonal noise to the covariance matrix to prevent it from becoming singular.

By leveraging these values from the sensors' datasheets, we initialize the covariance matrix with appropriate estimates of the uncertainties associated with each sensor's measurements. This initialization process helps establish a reliable foundation for the covariance matrix, which plays a key role in quantifying the uncertainty in the state estimates generated by the ES-EKF.

Moreover, the choice of representing orientation using quaternions in the ES-EKF further enhances its performance. Quaternions offer a compact representation of orientation, avoiding issues such as gimbal lock and ensuring smooth interpolation between orientations. Their numerical stability and convenient algebraic properties simplify calculations involving rotations, contributing to the overall robustness and efficiency of the state estimation process [Sola, 2017].

After reading the IMU data, true-acceleration a_t and true-angular rate w_t are obtained in the form of noisy sensor readings a_m and w_m with $\mathbf{R}_t \triangleq \mathbf{R}\{\mathbf{q}_t\}$. In [Sola, 2017], the readers can find the detailed calculations. In short, the true kinematic equations are :

TABLE 4.1 – Mathematical variables

$x_t, x, \delta x$	true-state, nominal-state, error-state
$p_t, p, \delta p$	true-position, nominal-position, error-position
$v_t, v, \delta v$	true-velocity, nominal-velocity, error-position
$q_t, q, \delta q$	true-quaternion, nominal-quaternion, error-quaternion
$R_t, R, \delta R$	true-rotation matrix, nominal-rotation matrix, error-rotation matrix
$\delta\theta$	error-angles vector
$a_{bt}, a_b, \delta a_b$	true-accelerometer bias, nominal-accelerometer bias, error-accelerometer bias
a_w	noise accelerometer bias
$w_{bt}, w_b, \delta w_b$	true-Gyrometer bias, nominal-Gyrometer bias, error-Gyrometer bias
w_w	noise Gyrometer bias
g_t	true-gravity
a_t, a_m, a_n	true-acceleration, measured acceleration, noise acceleration
w_t, w_m, w_n	true-angular rate, measured angular rate, noise angular rate

$$\dot{\mathbf{p}}_t = \mathbf{v}_t \tag{4.3}$$

$$\dot{\mathbf{v}}_t = \mathbf{R}_t \left(\mathbf{a}_m - \mathbf{a}_{bt} - \mathbf{a}_n\right) + \mathbf{g}_t \tag{4.4}$$

$$\dot{\mathbf{q}}_t = \frac{1}{2}\mathbf{q}_t \otimes \left(\boldsymbol{\omega}_m - \boldsymbol{\omega}_{bt} - \boldsymbol{\omega}_n\right) \tag{4.5}$$

$$\dot{\mathbf{a}}_{bt} = \mathbf{a}_w \tag{4.6}$$

$$\dot{\boldsymbol{\omega}}_{bt} = \boldsymbol{\omega}_w \tag{4.7}$$

$$\dot{\mathrm{g}}_t = 0, \tag{4.8}$$

Where the nominal-state kinematics, without any noise, corresponds to :

$$\dot{\mathbf{p}} = \mathbf{v} \tag{4.9}$$

$$\dot{\mathbf{v}} = \mathbf{R}\left(\mathbf{a}_m - \mathbf{a}_b\right) + \mathbf{g} \tag{4.10}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \left(\boldsymbol{\omega}_m - \omega_b\right) \tag{4.11}$$

$$\dot{\mathbf{a}}_b = 0 \tag{4.12}$$

$$\dot{\boldsymbol{\omega}}_b = 0 \tag{4.13}$$

$$\dot{\mathbf{g}} = 0, \tag{4.14}$$

And the error-state kinematics :

$$\dot{\delta}\mathbf{p} = \delta\mathbf{v} \tag{4.15}$$
$$\dot{\delta}\mathbf{v} = -\mathbf{R}\left[\mathbf{a}_m - \mathbf{a}_b\right]_\times \delta\boldsymbol{\theta} - \mathbf{R}\delta\mathbf{a}_b + \delta\mathbf{g} - \mathbf{R}\mathbf{a}_n \tag{4.16}$$
$$\dot{\delta} = -\left[\boldsymbol{\omega}_m - \boldsymbol{\omega}_b\right]_\times \delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_n \tag{4.17}$$
$$\dot{\delta}\mathbf{a}_b = \mathbf{a}_w \tag{4.18}$$
$$\dot{\boldsymbol{\omega}}_b = \boldsymbol{\omega}_w \tag{4.19}$$
$$\dot{\delta}\mathbf{g} = 0. \tag{4.20}$$

After examining error-state equations, the Jacobian metrics are :

$$\hat{\delta\mathbf{x}} \leftarrow \mathbf{F}_\mathbf{x}\left(\mathbf{x}, \mathbf{u}_m\right) \cdot \hat{\delta\mathbf{x}} \tag{4.21}$$
$$\mathbf{P} \leftarrow \mathbf{F}_\mathbf{x}\mathbf{P}\mathbf{F}_\mathbf{x}^\top + \mathbf{F}_\mathbf{i}\mathbf{Q}_\mathbf{i}\mathbf{F}_\mathbf{i}^\top \tag{4.22}$$

where $\delta\mathbf{x} \sim \mathcal{N}\{\hat{\delta\mathbf{x}}, \mathbf{P}\}; \mathbf{F}_\mathbf{x}$ and $\mathbf{F}_\mathbf{i}$ are the jacobians of $f()$.
When GPS data is received, the ES-EKF is corrected. Consider the GPS equation as such.

$$\mathbf{y} = h\left(\mathbf{x}_t\right) + v \tag{4.23}$$

Where $h()$ is the general nonlinear function of the true state, and v is a white Gaussian noise with covariance V

$$v \sim \mathcal{N}\{0, \mathbf{V}\} \tag{4.24}$$

ES-EKF estimates the error, and the correction equations are as follows :

$$\mathbf{K} = \mathbf{P}\mathbf{H}^\top\left(\mathbf{H}\mathbf{P}\mathbf{H}^\top + \mathbf{V}\right)^{-1} \tag{4.25}$$
$$\hat{\delta\mathbf{x}} \leftarrow \mathbf{K}\left(\mathbf{y} - h\left(\hat{\mathbf{x}}_t\right)\right) \tag{4.26}$$
$$\mathbf{P} \leftarrow \mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P} \tag{4.27}$$

This will lead to the Jacobian matrix H to be defined concerning the error state δ_x :

$$\mathbf{H} \equiv \left.\frac{\partial h}{\partial \delta\mathbf{x}}\right|_\mathbf{x} \tag{4.28}$$

And finally, the error state mean gets resets.

3 Experiments and Results

In this section, we explore the implementation and outcomes of our study. The focal points of our investigation lie in the detailed examination of two components : the Carla Simulation and the Real Test. Each subsection details the distinct environments, methodologies employed,

and the results procured. A central focus lies in the critical evaluation and interpretation of the obtained results, providing valuable insights into the practical implications and effectiveness of the proposed methodologies across simulated and real-world contexts.

3.1 Carla Simulation

3.1.1 Simulation Environment

To compare ES-EKF and EKF, a simulation was conducted on the Carla simulator [Dosovitskiy, 2017], an experimental platform with ego vehicles equipped with GPS and IMU. The GPS provides position measurements at a rate of 10Hz, while the IMU sensor measures accelerometer, gyroscope, and compass at 100Hz. Every 100 updates of IMU navigation, ten GPS difference observations are obtained. The localization estimation algorithm, implemented in Python 3.7, ran on a computer with CPU i7-7700 @ 4.2 GHz and 16 GB RAM. CARLA simulates 3D objects like cars, pedestrians, and infrastructure, using a client-server architecture based on Unreal Engine 4 (UE4) [Sanders, 2016]. CARLA provides pre-defined road networks with specific lane markings and traffic rules, allowing realistic testing of autonomous vehicle behavior.

3.1.2 Simulation Results

Fig. 4.2 and Fig. 4.3 show the impact of GPS and IMU sensor fusion using error state extended Kalman filter (ES-EKF) on enhancing the prediction of the x and y axis of velocity and yaw. The graphs show that the ES-EKF significantly improves the accuracy of the predictions, especially in the case of yaw.

In 4.2 and 4.3, the blue line shows the x and y-axis predictions of velocity and yaw using only the IMU sensor. The red line shows the predictions using the ES-EKF. As can be seen, the ES-EKF predictions are much more accurate, especially in the case of yaw. The accuracy of the predictions improves because the ES-EKF considers the system's non-linearities. The GPS sensor provides linear measurements, but the vehicle's actual motion is non-linear. The ES-EKF linearizes the system around the current state estimate, which allows it to track the vehicle's non-linear motion more accurately. The ES-EKF is a powerful tool for enhancing the accuracy of predicting the x and y axis of velocity and yaw. It is beneficial in applications where the accuracy of the predictions is critical, such as autonomous vehicles. Here are some additional details about the image :

— The graphs show the predictions for 10 seconds.

— The x and y axis of velocity is measured in meters per second.

— The yaw is measured in degrees.

— The error bars show the standard deviation of the predictions.

The first thing to note is that the ES-EKF predictions are much smoother than the GPS predictions. This is because the ES-EKF can filter out the noise in the GPS measurements. Various factors, such as atmospheric interference and the inaccuracy of the GPS satellites, cause noise in the GPS measurements. The ES-EKF can filter out this noise by using a Kalman filter.

The second point to consider is that ES-EKF predictions tend to be more accurate than those solely based on GPS data. This is particularly evident in the prediction of yaw angles. While GPS sensors excel in measuring the vehicle's position, they may not offer the same precision when it comes to yaw angles. On the other hand, IMU sensors are inherently better at measuring yaw angles but may exhibit drift or inaccuracies over prolonged periods. By leveraging the complementary strengths of both sensors, the ES-EKF effectively combines their inputs to generate more precise predictions of the vehicle's yaw angles.

The third thing to note is that the error bars are smaller for the ES-EKF predictions than for GPS predictions. This means that the ES-EKF predictions are more reliable. The error bars show the standard deviation of the predictions. The smaller the error bars, the more confident we can be in the predictions. Overall, the graph analysis shows that the ES-EKF significantly improves the accuracy of the x and y-axis predictions of velocity and yaw. The ES-EKF is a powerful tool for enhancing the accuracy of the prediction of these quantities.

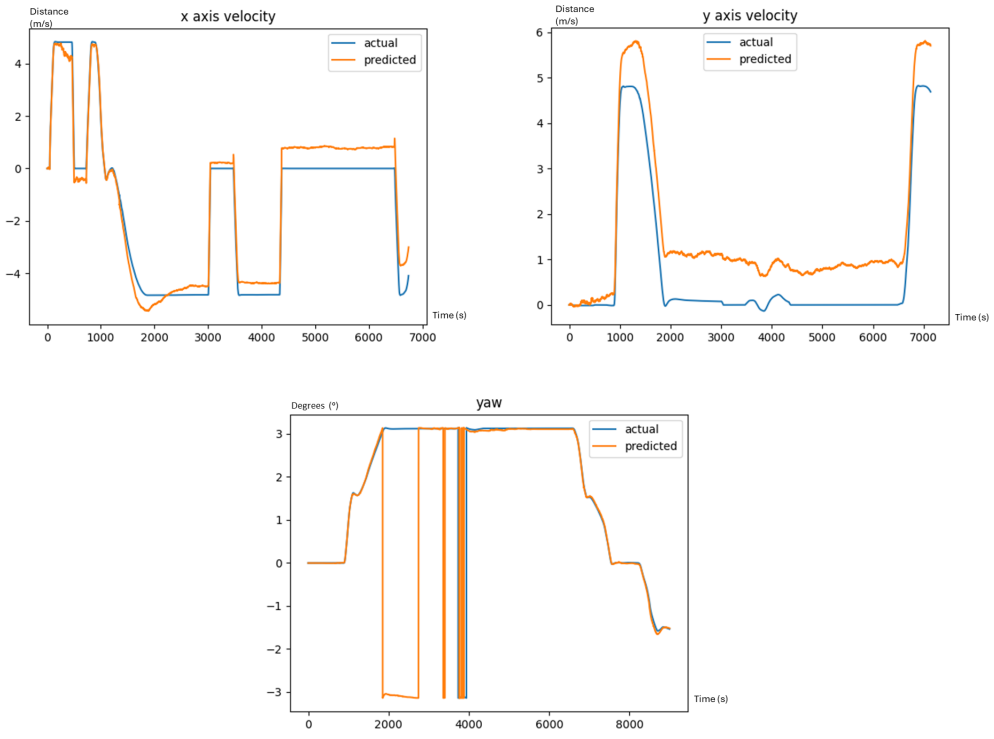


FIGURE 4.2 – IMU predictions comparing to ground truth

The first experiment depicted in Fig.4.4 compares the performance of EKF and the ES-EKF on a simulated dataset. This research was conducted using the CARLA simulator, renowned for its realistic emulation of driving scenarios, rendering it a valuable tool for developing and testing autonomous vehicles.

The experiment outcomes demonstrated that the ES-EKF exhibited marginal enhancements in position estimation accuracy compared to the EKF estimations. This improvement can be attributed to the ES-EKF's adeptness in modeling the system's inherent non-linearities. However, it was observed that the ES-EKF also resulted in a slight increase in orientation error compa-

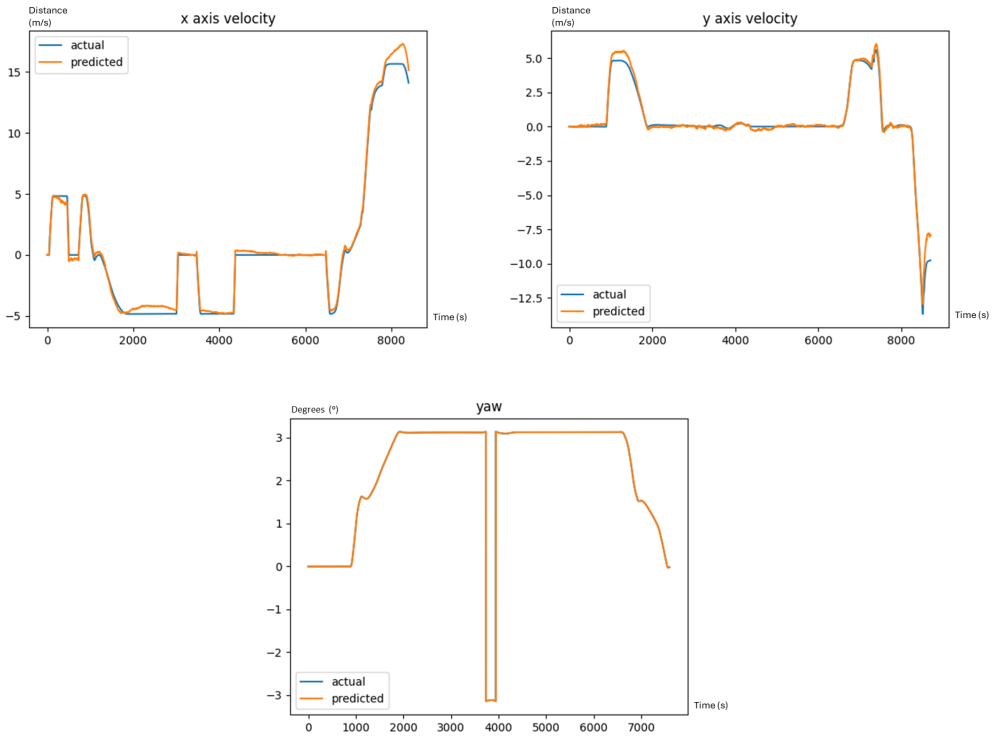


FIGURE 4.3 – ES-EKF predictions comparing to ground truth

red to the EKF estimations. This is potentially due to the ES-EKF's heightened sensitivity to measurement noise.

The marginal increase in orientation error observed in the ES-EKF is unsurprising, considering the experiment utilized noise-free sensor data. Consequently, the ES-EKF's sensitivity to measurement noise became more apparent. Nonetheless, the ES-EKF still exhibited a marginal improvement in orientation estimate accuracy compared to the EKF estimations.

In summation, the findings of this experiment showcase the promising potential of the ES-EKF as a valuable approach to enhance the accuracy of vehicle state estimation. However, it is essential to underscore that further research is warranted to determine the optimal configurations of the ES-EKF for diverse real-world applications.

In addition, using a simulated dataset may limit the direct applicability of these findings to real-world scenarios. Nevertheless, this experiment serves as a valuable foundation for assessing the performance of the ES-EKF in varied practical contexts. Furthermore, it is essential to acknowledge that employing noise-free sensor data in this research may underestimate the ES-EKF error in real-world conditions. Despite this, the experiment effectively offers valuable insights into the potential benefits of the ES-EKF in augmenting vehicle state estimation accuracy.

One must also consider that the ES-EKF, a more complex algorithm than the conventional EKF, demands increased computational resources. However, the potential gains in estimation accuracy may justify the additional computational overhead, substantiating the ES-EKF as an attractive option in specific applications.

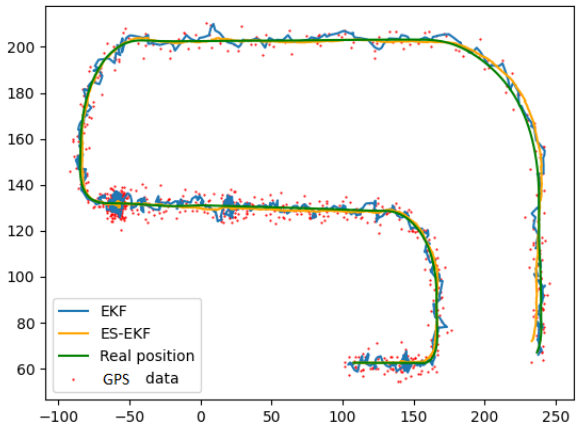


FIGURE 4.4 – Carla simulation results

3.2 Real Test

The application is tested within real-world conditions during the Real Test phase. This segment represents the culmination of theoretical frameworks and simulation endeavors, undertaking the empirical domain to assess the system's viability. The evaluation unfolds in two primary facets : the Real Test Environment and the subsequent calibration and filtering procedures applied to Inertial Measurement Units (IMUs). This validation was conducted in Vallauris, France.

3.2.1 Real Test Environment

The architecture described in Fig.4.6 involves the utilization of FreeRTOS, a real-time operating system, on the Environmental Interface Card (EIC). Its primary function is to recover real-time data sent by the GPS and IMU sensors and inject them into the ES-EKF. The ES-EKF processes this data, generating advanced GPS coordinates, which are then transmitted to the Electronic Control Unit (ECU) Nvidia Jetson TX2 via a Universal asynchronous receiver transmitter (UART). The IMU used in the setup is ICM 20600, and the GPS sensor is Quectel L80-M39, both interfaced on the EIC.

Implementing FreeRTOS in a real-time environment, coded in C language on the STM32, ensures timely data retrieval and processing on the EIC. The NUCLEO-H743ZI kit facilitates data acquisition from the GPS at 10 Hz via UART and from the IMU at 100 Hz via the Serial Peripheral Interface (SPI). The efficient data transmission between the EIC and ECU enables accurate localization and navigation for the autonomous vehicle.

This comprehensive setup showcases a sophisticated and efficient approach to vehicle localization and control, demonstrating the potential for practical implementation in real-world autonomous vehicle systems. It highlights the significance of employing a real-time operating system like FreeRTOS to ensure timely data processing and sensor fusion, enabling precise motion estimation for reliable navigation. The integration of low-cost GPS and IMU sensors and

advanced filtering techniques exemplify the system's robustness and accuracy in various scenarios. This architecture advances modern transportation and mobility technologies, offering promising prospects for future research and applications in autonomous navigation systems.

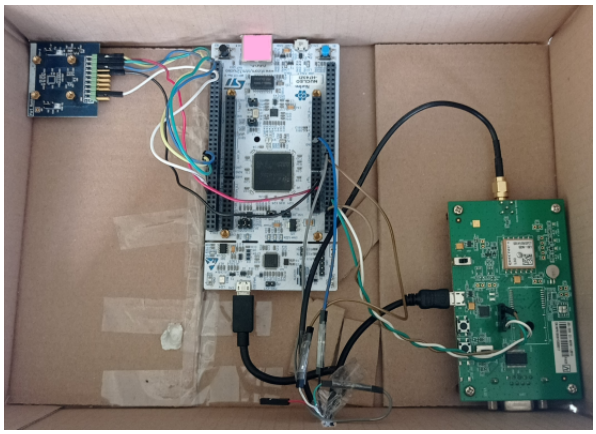


FIGURE 4.5 – EIC and ECU

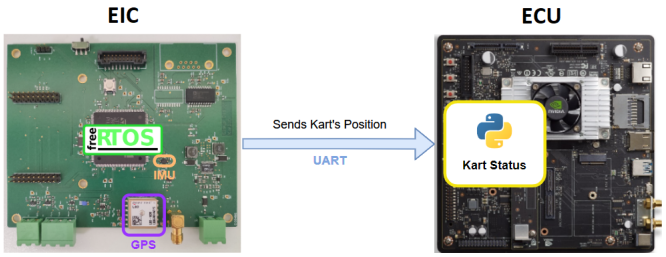


FIGURE 4.6 – EIC and ECU

3.2.2 IMU Calibration and Filtering

Calibration of IMUs is a fundamental process aimed at eliminating sensor offsets in accelerometers and gyroscopes arising from manufacturing tolerances to enhance measurement accuracy. The calibration is conducted under stationary conditions on a known plane with a predetermined degree of inclination, considering that the offsets of the accelerometer and gyroscope are contingent upon the sensor orientation. To achieve calibration, the vehicle is positioned flat, stationary, in neutral gear, and at a red light, and the sensor values are acquired over a designated time frame and subsequently averaged, as shown in Table 4.2 :

The obtained offset values were relatively close to those mentioned above throughout the vehicle's journeys. However, it was evident that these values could exhibit substantial variations contingent upon the IMU's position, mainly attributable to the influences of gravitational and magnetic fields. Consequently, it is imperative to recalculate the offsets at the initiation of each journey to ensure their accuracy concerning the IMU's prevailing position. The calibration's efficacy is validated through a comparative analysis of the accelerometer and gyroscope received signal values before and after calibration, juxtaposed against ideal values that would be

TABLE 4.2 – IMU Calibaration

Accelerometer X	0.266818 m/s^2
Accelerometer Y	0.180452 m/s^2
Accelerometer Z	-0.089558 m/s^2
Gyroscope X	-1.237829 deg/s
Gyroscope Y	-2.018467 deg/s
Gyroscope Z	-0.798545 deg/s

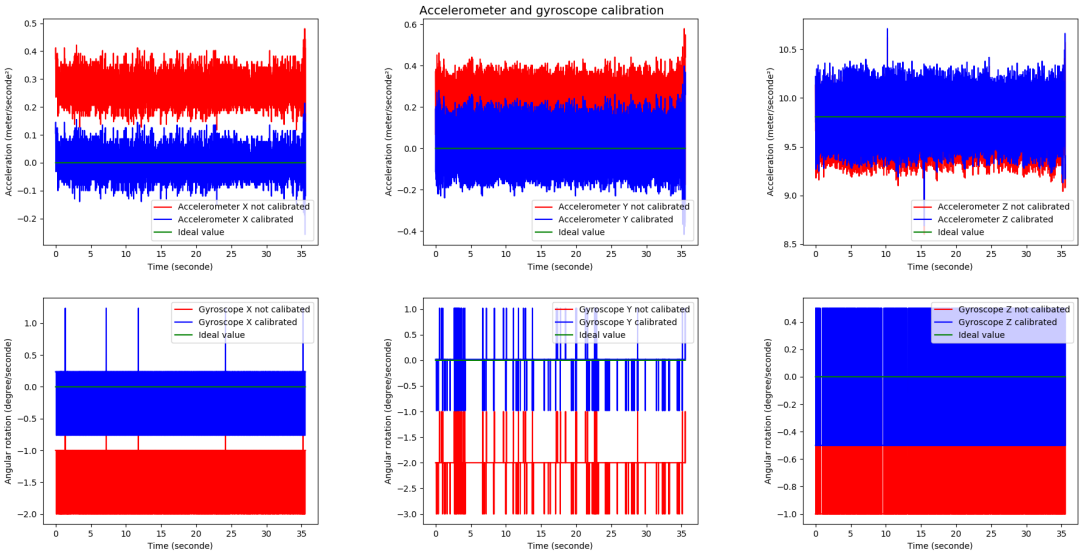


FIGURE 4.7 – Example of IMU before calibration

attained without offsets. Notably, the pre-calibration values deviate significantly from the ideal values, attributable to non-zero offsets. Conversely, post-calibration, the values exhibit remarkable proximity to the ideal values, substantiating the successful removal of offsets through the calibration process. Fig.4.7 shows the accelerometer and gyroscope received signal values before calibration compared to the ideal values.

In summary, the calibration process exemplifies its pivotal role in offset elimination to enhance the reliability and accuracy of IMU measurements. The dependence of offsets on the IMU's position underscores the necessity for periodic recalibration to ensure accurate readings in varying operational conditions. Moreover, the calibration's evident success, as manifested in the improved correspondence with ideal values, bolsters the credibility and effectiveness of this essential calibration procedure for IMUs.

3.2.3 Low-Pass Filter

Fig.4.8 illustrates the outcomes of applying various low-pass filters to accelerometer data, highlighting the filter's efficacy in noise reduction. The low pass filter, a straightforward technique, operates by averaging the previous and current data points, with the α value regulating the weight attributed to the previous value. The image displays multiple attempts to identify

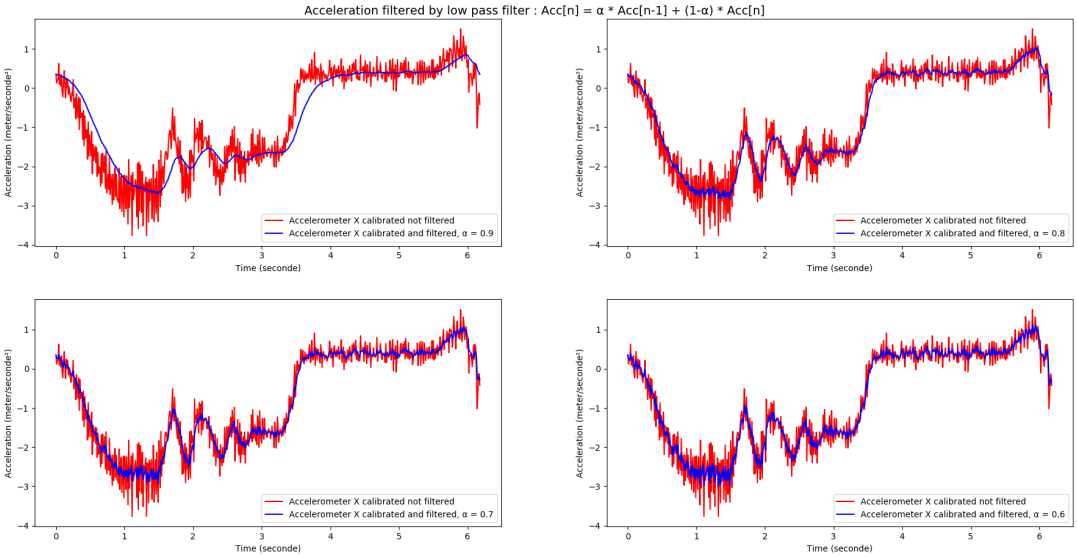


FIGURE 4.8 – Different attempts for α selection

the optimal α values, with the blue line denoting the unfiltered data and other lines representing data subjected to distinct low-pass filters, each associated with a specific α value.

The visual analysis indicates that an α value around 0.6 yields the best fit, signifying that approximately 80% of the weight is ascribed to the previous value. In contrast, 20% is allocated to the current value. This configuration establishes an advantageous equilibrium between noise reduction and signal preservation, offering a reliable outcome.

Undoubtedly, the low-pass filter significantly impacts noise reduction in the accelerometer data. In contrast to the highly noisy unfiltered data, the application of the filter results in smoother data, achieved through the elimination of high-frequency noise components.

However, it is essential to note that the low pass filter slightly affects the signal in the data. While the filtered data may be slightly less accurate than the unfiltered data, the difference is not substantial, corroborating the filter's capability to maintain overall data fidelity.

In conclusion, the low pass filter is a valuable tool for mitigating noise in accelerometer data. The optimal α value, approximately 0.6 in this case, is a practical choice, striking an appropriate balance between noise reduction and signal preservation. It is essential to recognize that the α value can be adjusted depending on the specific application's priorities. Higher α values prioritize noise reduction, while lower values prioritize signal preservation. Moreover, the low pass filter serves the secondary purpose of smoothing out data, enhancing its utility in scenarios where the signal might be less distinct. As a result, the low pass filter emerges as an accessible yet potent solution for noise reduction in accelerometer data analysis, contributing to enhanced data accuracy and reliability across various applications.

Fig.4.9 depicts a comparative analysis of the accelerometer and gyroscope received signal values to the ideal value after applying a low pass filter with α values ranging from 0.6 to 0.7. The low pass filter has yielded noise reduction in the data. The filtered data exhibits a smoother trajectory than the unfiltered data, and the disparity between the filtered data and the ideal value has been substantially minimized.

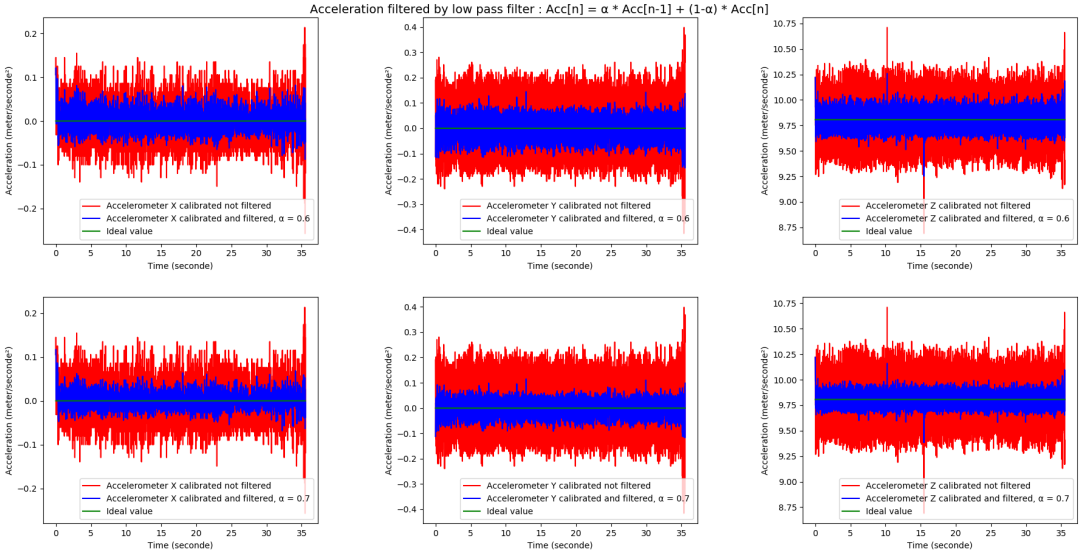


FIGURE 4.9 – IMU data after applying low pass filter

Furthermore, the low pass filter has effectively bridged the discrepancy between the IMU-calibrated and ideal values. This outcome can be attributed to the filter's efficacy in eliminating noise, which previously contributed to the observed difference between the two datasets.

The low-pass filter has significantly enhanced the accuracy of the accelerometer and gyroscope data. When juxtaposed against the unfiltered data, the filtered data showcases superior smoothness and accuracy, and the deviation from the ideal value has been substantially mitigated. This corroborates the filter's potency in promoting the accuracy and reliability of motion measurements from the IMU.

Additionally, it is imperative to underscore the low pass filter's simplicity and efficacy in reducing noise in accelerometer and gyroscope data. The α value in the interval $[0.6\,;\,0.7]$ appears particularly suitable, as it strikes an appropriate balance between noise reduction and signal preservation, optimizing the filter's performance. Moreover, the filter's ability to smoothen out data holds valuable applications, especially in scenarios where the signal may not be distinctly discernible.

The image also illustrates the low pass filter's capacity to bridge the gap between IMU-calibrated and ideal values. This signifies the filter's valuable role in enhancing the calibration process by minimizing discrepancies induced by noise, resulting in more accurate motion measurements.

In conclusion, the low pass filter has proven to be a potent and versatile tool in motion data analysis. Its effectiveness in noise reduction, signal smoothing, and facilitating calibration underscores its significance in enhancing the accuracy and reliability of IMU-derived data, thereby contributing to diverse fields such as navigation, robotics, and motion tracking.

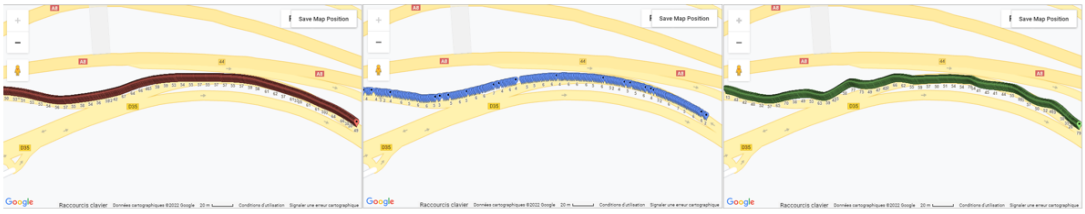


FIGURE 4.10 – Data fusion without IMU calibration

3.2.4 Data Fusion Results

In the southeastern region of France, specifically in Vallauris, Alpes-Maritimes department, Provence-Alpes-Côte d'Azur region, we conducted a series of experiments to evaluate the proposed system's performance. The system was mounted on a vehicle, and various scenarios were tested to assess the reliability and effectiveness of the proposed algorithm.

Throughout all the tests, we utilized a color-coded representation for the data visualization : red dots represented IMU data, blue dots represented GPS data, and green dots represented the data fusion results.

We aimed to demonstrate the data fusion results without IMU calibration in the initial attempt. The scenario involved a simple straight line without curves, which ideally yields accurate results for the system. However, upon analyzing Fig.4.10, it is evident that the data fusion results deviated from the road track, and the accuracy compared to the real-world ground truth was below 60%. This percentage represents the proportion of data points where the estimated position (X, Y) from the ES-EKF falls within a confidence level of a 0.5m distance threshold of the corresponding ground truth position.

The discrepancy in the results can be attributed to the absence of IMU calibration, which is crucial for enhancing the accuracy of the fusion process. The uncalibrated IMU data introduces errors and inconsistencies, leading to inaccurate fusion outcomes. In a straightforward straight-line scenario, one would expect more accurate results. Future iterations of the system should incorporate IMU calibration techniques to unlock its full potential and enhance its applicability in various practical settings.

This study examines the effects of applying a low pass filter to the IMU accelerometer and gyroscope data. The objective is to assess the impact of calibration on the system's performance, as depicted in Fig.4.11. Notably, the system's ability to achieve high accuracy in various scenarios, including straight lines and roundabouts, is evaluated by comparing the data fusion results to the actual trajectory traced by the vehicle.

A significant improvement in accuracy is observed upon applying the low-pass filter. The system exhibits remarkable performance in straight-line trajectories and complex roundabout maneuvers, achieving accuracy levels of up to 96% calculated by measuring the proportion of data points where the estimated position (X, Y) from the ES-EKF falls within a confidence level of a 0.5m distance threshold of the corresponding ground truth position.

The effectiveness of the low pass filter can be attributed to its capability to reduce noise and enhance data smoothness. The filter successfully attenuates high-frequency noise by averaging

FIGURE 4.11 – Data fusion with IMU calibration

the old and new values with controlled weighting (α), resulting in a more refined and reliable data fusion process.

Moreover, implementing IMU calibration plays a pivotal role in enhancing accuracy. By removing sensor biases and inaccuracies, calibration ensures that the IMU data is aligned with the actual motion of the vehicle. Consequently, the fusion results demonstrate higher fidelity to the actual trajectory.

The obtained accuracy levels indicate that the proposed system, equipped with the low pass filter and calibrated IMU, can provide highly accurate motion estimation. Such precision is crucial in various applications, including autonomous navigation, vehicle tracking, and robotics, where accurate trajectory information is essential for safe and effective operation.

It is worth noting that while the presented results showcase the system's remarkable performance, further investigations may be warranted to explore the impact of different filter settings (α values) and calibration techniques on performance under diverse environmental conditions and driving scenarios.

Overall, the findings of this study underscore the significance of employing low-pass filters and IMU calibration in motion estimation systems. By mitigating noise and sensor inaccuracies, these techniques pave the way for more reliable and accurate trajectory predictions, enhancing the overall efficacy and applicability of the system in real-world settings. Future research may explore integrating additional sensor data and advanced filtering algorithms to improve the accuracy and robustness of motion estimation systems.

In this research, we sought to assess the robustness of our filter by subjecting it to a more challenging test scenario. To complicate the test, we deliberately reduced the frequency of the GPS updates to 0.2 Hz, simulating an environment with intermittent or blocked GPS signals. Despite this reduction in GPS updates, the system demonstrated remarkable performance, achieving an accuracy level of 92% as evidenced in Fig.4.12 within the confidence level of 0.5m distance threshold.

The ability of our filter to maintain high accuracy under these conditions highlights its

FIGURE 4.12 – GPS blocked for 5 seconds

resilience in handling adverse situations with limited or interrupted GPS data. By leveraging the low-pass filter and IMU calibration, the system successfully compensated for the reduced GPS frequency, preserving the accuracy of the data fusion process.

During the test, the reduced GPS updates presented a challenge in maintaining accurate trajectory information, as the system had to rely more heavily on IMU data. The low pass filter played a pivotal role in attenuating noise and extracting valuable information from the IMU sensors, which, when combined with the calibrated data, led to the accuracy achieved.

The 92% accuracy level indicates the filter's capacity to make reliable predictions, even when GPS signals are sparse or temporarily blocked. Such robustness is highly desirable in real-world applications, especially in urban environments with tall buildings, tunnels, or other obstructions that can obstruct GPS signals.

It is essential to highlight the potential implications of these findings for autonomous navigation and other safety-critical applications. A filter with high robustness, such as the one demonstrated in this study, can instill confidence in the system's ability to handle challenging environments effectively.

Future research could explore a broader range of scenarios to validate further and generalize our findings, varying the duration and frequency of GPS blockages and testing the filter's performance under different driving conditions and terrains. Additionally, assessing the impact of varying filter parameters and calibration settings would offer valuable insights into optimizing the system's robustness and accuracy in various operational contexts.

In conclusion, the results obtained in this study underscore the robustness of our filter in handling situations with reduced GPS frequency. The high accuracy achieved, even under challenging conditions, reflects the efficacy of the low pass filter and IMU calibration in preserving the system's accuracy and reliability. These findings have significant implications for autonomous navigation and related fields, where robust motion estimation is essential for safe and efficient operation in real-world environments.

In the concluding phase of our experimentation, we designed the final test to assess the robustness of our filter under the most challenging conditions. The primary objective was to subject the system to an extensive evaluation encompassing diverse scenarios. As anticipated, the system exhibited remarkable resilience, achieving an accuracy of 90% while maintaining an

efficient execution time of approximately 20 µs in the same confidence level of 0.5m. The test was conducted with a notably low GPS update frequency of 0.05 Hz, and the road configuration comprised a complex combination of a roundabout, a straight line segment, and an intersection, as illustrated in Fig.4.13.

The system's robustness in maintaining high accuracy despite the infrequent GPS updates signifies the effectiveness of our low pass filter and IMU calibration techniques. By judiciously fusing the IMU data with the limited GPS information, the filter mitigated the adverse effects of reduced GPS frequency and generated reliable motion estimations.

The test's mix of various road scenarios, including a roundabout, a straight line, and an intersection, introduced complexities that could challenge the accuracy and consistency of the system. However, the filter's performance remained steady, delivering accurate results across these diverse scenarios.

The execution time of approximately 20 μs indicates our filter's computational efficiency. The swift execution time is essential, particularly in real-time applications, where rapid data processing is crucial for timely decisions.

The high accuracy and efficient execution time demonstrated by our filter in this rigorous test scenario have significant implications for autonomous navigation and other safety-critical applications. The system's robustness in handling diverse road configurations and low GPS update rates enhance its applicability in real-world environments with challenging conditions, such as urban landscapes with signal obstructions and complex road layouts.

The 90% accuracy achieved in this comprehensive evaluation bolsters confidence in the filter's ability to deliver reliable motion estimations, even in scenarios with limited GPS data. The successful performance under these demanding conditions underscores the practicality and efficacy of our filter for real-world deployment within the confidence level of 0.5m distance threshold.

In conclusion, the final test comprehensively assessed our filter's robustness, accuracy, and computational efficiency. The system showcased remarkable resilience, achieving high accuracy and rapid execution time despite reduced GPS updates and a diverse set of road scenarios. These findings reinforce the potential of our filter for various applications, particularly in autonomous navigation and other safety-critical domains where precise and efficient motion estimation is of utmost importance. Future research could explore further optimizations and validate the filter's performance in large-scale real-world scenarios to strengthen its application in autonomous systems and related fields.

FIGURE 4.13 – GPS blocked for 20 seconds

4 Conclusion

In conclusion, vehicle localization is a crucial aspect of autonomous navigation, demanding precise and reliable position estimation for safe and efficient operation. In this chapter, we presented a hybrid solution that capitalizes on the strengths of low-cost GPS and IMU sensors through a Kalman filter-based fusion approach. The proposed hardware design encompassed a low-cost GPS module, an IMU sensor, and a microcontroller. In contrast, the software design incorporated a Kalman filter algorithm within a real-time operating system. Comparative evaluations on the Carla simulator demonstrated the proposed method's superior accuracy in diverse scenarios compared to the traditional Extended Kalman Filter (EKF). Real-time executions in an urban driving context confirmed the method's performance, achieving over 92% accuracy across various configurations and driving maneuvers with the confidence level of 0.5m distance threshold. The proposed solution's real-world deployment in an autonomous vehicle further validated its reliability and applicability in industrial settings. The method exhibited robustness in different environments, reaffirming its potential for practical implementation. The contribution of this chapter lies in providing a promising approach for vehicle localization that fuses GPS and IMU measurements through a Kalman filter-based fusion scheme. The accuracy, reliability, and implementation feasibility on low-cost hardware make this solution particularly appealing for autonomous vehicle systems. Future research directions could explore further optimizations to enhance the system's performance in challenging conditions and expand its applicability in large-scale urban environments. Additionally, investigations into incorporating additional sensor data, such as LiDAR or camera inputs, could further enrich the proposed method's localization capabilities.

In conclusion, the presented solution is valuable to vehicle localization, offering a robust, accurate, and cost-effective approach suitable for practical deployment in autonomous navigation systems. With continued advancements and refinements, this hybrid solution holds promise for addressing the demands of real-world autonomous vehicles, driving this critical technology's advancement in modern transportation and mobility systems. In the next chapter, we will explore decision-making for automated parking systems (APS) and propose an advanced autonomous driving system that takes complete control over a vehicle to perform parking maneuvers with

full autonomy, ensuring user security, time efficiency, and optimized parking space utilization.

Chapitre 5

Dynamic Adjustment of Reward Function for PPO with IL : Application to Automated Parking Systems

Contents

1 Introduction

The global automotive industry has witnessed a substantial increase in vehicles compared to the population in recent years, resulting in an augmented risk of road accidents. Advanced Driver Assistance Systems (ADAS) for autonomous driving have been introduced to prevent fatalities and injuries caused by car accidents [Farag, 2019]. These ADAS systems, equipped with advanced sensors like radar, cameras, and LiDAR, offer various applications, including lane keep and correction, pedestrian detection and avoidance, emergency braking, and Automated Parking Systems (APS).

This chapter centers explicitly on APS, an advanced autonomous driving system that takes complete control over a vehicle to perform parking maneuvers with complete autonomy, ensuring user comfort, security, time efficiency, and optimized parking space utilization. APS's architecture comprises three fundamental methods : parking spot exploration, path planning, and path tracking. While APS is not a novel concept, various path planning methods have been explored, such as geometric, sampling, and numerical optimization techniques [Fraichard, 2004 ; Gómez-Bravo, 2001 ; Lini, 2011]. Path tracking methods, like the Ackerman steering model and simple kinematics, have also been employed [Weinstein, 2010]. However, these conventional approaches often do not account for the European standard BS ISO 16787-2017 Intelligent Transport Systems, which mandates APS to have an inclination angle restricted to $\pm 3°$ and a deviation of less than 0.2 meters [Standardization, 2017b].

This study explores the potential of leveraging recent advancements in machine learning, particularly deep reinforcement learning (DRL), to enhance APS performance and overcome the limitations of traditional methods. DRL is an actively researched area where an agent learns to behave intelligently by exploring its environment, taking actions, and receiving rewards [Sutton, 2018]. In this context, DRL trains an agent to perform the auto parking task. Recent developments in artificial neural networks and computational processing capabilities have resulted in significant breakthroughs in various domains, such as image classification and natural language processing. DRL has achieved human-level control in diverse fields, including Atari video games [Mnih, 2015], Go [Schrittwieser, 2020], and robot manipulation [Schulman, 2017].

While previous works have demonstrated promising results in auto parking tasks using DRL [Folkers, 2019 ; Schulman, 2017 ; Fehér, 2019 ; Fehér, 2020], most of them only considered static obstacles in the environment, neglecting dynamic obstacles and the braking constraint was not addressed. Furthermore, despite the success of DRL in simple domains like Atari video games, it still faces challenges in adapting to complex real-world environments, such as APS, due to the difficulty in formulating a suitable reward function [Codevilla, 2019].

To address these challenges, this chapter presents our contributions to enhance APS performance : (i) Utilizing a DRL algorithm, PPO, to define a Markov decision process that enables the agent to explore empty parking spots effectively, plan optimal paths, and park vehicles safely while avoiding both static obstacles like parked cars and traffic signs and dynamic obstacles like moving cars and pedestrians, with the ability to accelerate or brake when necessary ; (ii) Introducing a dynamic adjustment of the reward function, leveraging intrinsic reward signals to

facilitate exploration during training and improve the DRL algorithm ; (iii) Reducing training time by integrating IL with DRL, enabling the agent to learn from expert demonstrations ; (iv) Developing a task-specific curriculum learning approach to train the agent in highly complex environments effectively ; (v) Conducting extensive experiments to demonstrate the effectiveness of the proposed model, achieving a remarkable 90% success rate in executing sophisticated control commands, with 97% of them accurately aligned with parking spots having an inclination angle $> \pm 0{,}2°$ and a deviation > 0.1 meter.

Through these contributions, we aim to demonstrate the potential of DRL in APS capabilities and advancing the field of autonomous parking systems, paving the way toward safer and more efficient parking solutions.

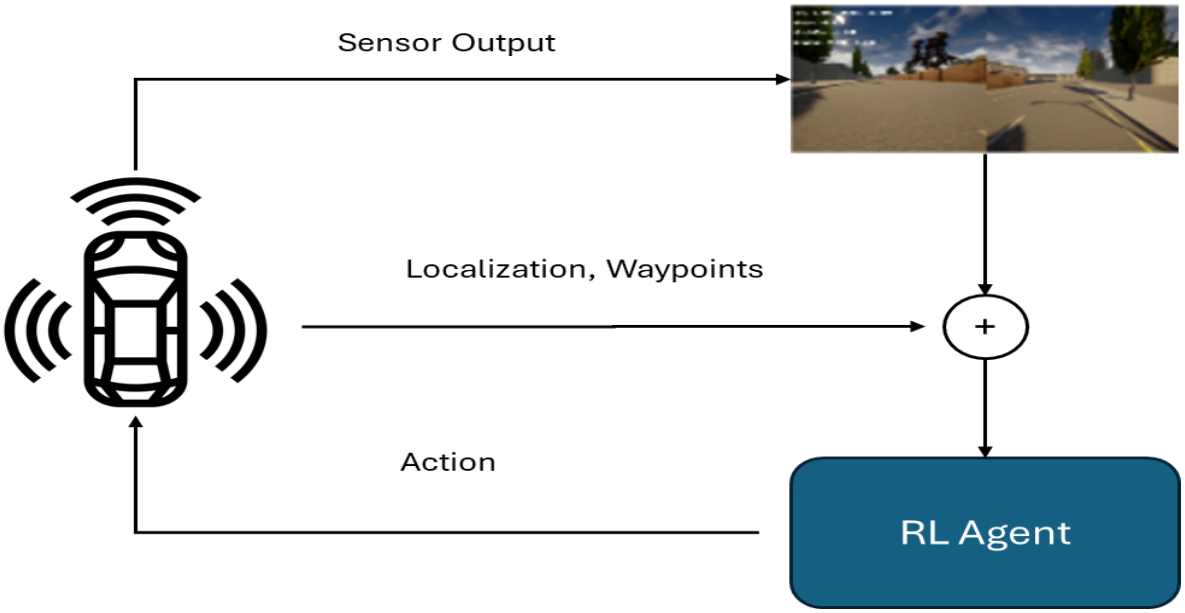


FIGURE 5.1 – DPPO-IL architecture

2 Statment

This research presents a complex and challenging autonomous parking scenario for an agent. The task involves the agent beginning from a randomly positioned and oriented location within the blue zone (Fig. 5.2). The parking space is situated randomly in the red zone. The agent aims to park the vehicle within a minute, ensuring proper alignment while adhering to all the test standards specified in ISO 16787 :2017. The maximum allowable distance of 70 meters between the vehicle's starting position and the targeted parking spot is established to streamline the parking process and optimize efficiency. While this distance is not standardized, it serves to focus the scenario specifically on finding and parking a car rather than navigating lengthy distances. By imposing this limit, the self-driving car system prioritizes the parking task itself, ensuring that the vehicle dedicates its resources to locating an appropriate parking spot within reasonable proximity and efficiently maneuvering into it.

Completing this scenario demands the agent to identify an empty parking spot amidst a complex scene cluttered with various obstacles. Subsequently, it must plan a safe and efficient path to reach the designated parking spot, avoiding collisions with obstacles. Once at the parking space, the agent must execute the parking maneuver with precise alignment, requiring adept control of the vehicle's speed and steering. Remarkably, the agent must learn all these essential skills in a single shot without prior knowledge of the environment or the parking task.

This scenario poses several challenges for autonomous parking algorithms :

— The agent must exhibit robust capability in spot identification within a cluttered scene.

— It must demonstrate effective path planning while avoiding obstacles, ensuring safety and efficiency.

— The agent must master the art of precise parking with the specified alignments, which demands precise control over the vehicle.

— The agent must achieve all these skills in an integrated and seamless manner without any prior experience with the environment or the specific parking task.

This challenging scenario is a valuable benchmark for evaluating autonomous parking algorithms' performance. The experimental results from our study will provide valuable insights into autonomous parking and the potential of machine learning approaches in addressing these complex challenges.

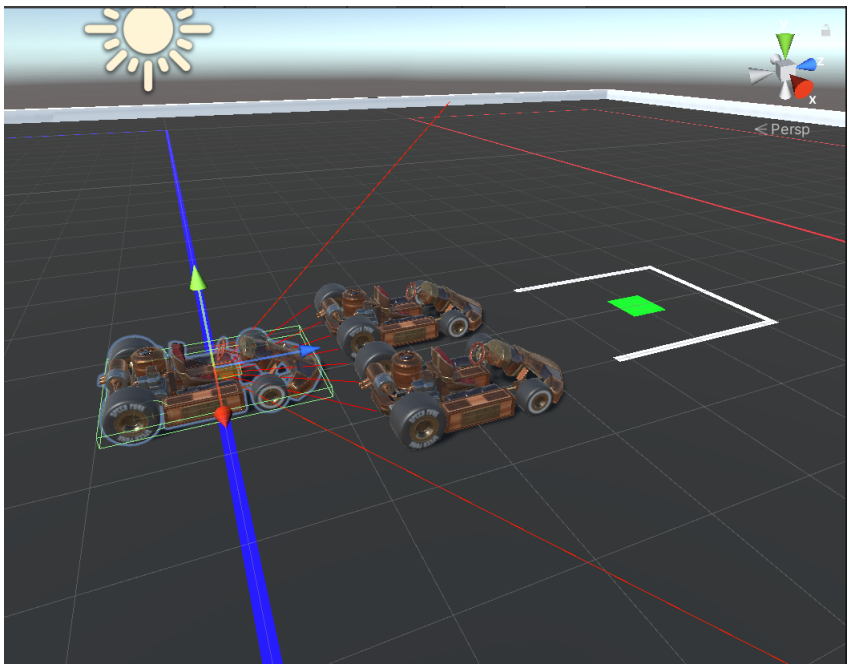


FIGURE 5.2 – Vehicle modeling

3 DPPO-IL, a Dynamic Adjustment of Reward Function for PPO

As an initial step towards proposing a framework that aligns with our primary objective, we conducted a comparative study of several widely used DRL algorithms on a simple parking scenario. In this scenario, the parking space was situated less than 30 meters from the agent, with no brake system in place and no control over the vehicle's arrival speed at the parking spot. The agent and the parking space were randomly initialized within the parking area, and the episode concluded either when the agent moved more than 30 meters from the parking spot, exceeded the time limit, or successfully parked the vehicle (with a distance of less than 0.5 meters from the parking space borders) [Codevilla, 2019].

For this scenario, we selected three algorithms for comparison : PPO [Schulman, 2017], Soft Actor-Critic (SAC) [Haarnoja, 2018], and MultiAgent POsthumous Credit Assignment (MA-POCA) from the Unity ML-agent library. SAC, an off-policy DRL algorithm, outperformed DDPG and aimed to maximize the long-term reward to find the optimal policy. Among the three algorithms tested on the simple scenario, PPO demonstrated the best convergence time, achieving an impressive 98% success rate. Additionally, PPO proved to be the most straightforward algorithm to implement compared to the others [Schulman, 2017].

Based on these results, we proceeded with PPO and evaluated its performance on a more complex scenario, as presented in Section 2. However, in the complex scenario, the success rate dropped to 74.8%. It became evident that while PPO achieved a perfect success rate in a simple scenario with a well-defined reward signal, its performance diminished in a real-world domain, such as driving a car in an environment with dynamic and static obstacles [Codevilla, 2019]. The extensive state space made it challenging for the agent to determine the optimal actions for each state and initialize the neural network weights effectively. Furthermore, formulating a sound mathematical reward function that precisely distinguishes between good and bad policies presented a formidable challenge. The absence of a suitable reward function rendered DRL ill-posed for addressing the complexities of Automated Parking Systems (APS) [Codevilla, 2019].

To address these limitations, we propose three improvements to the PPO algorithm : (1) a dynamic adjustment of the reward function to encourage the agent to explore more using intrinsic curiosity, a self-supervised prediction approach that addresses the exploration problem ; (2) utilizing IL, whereby a good policy can be learned by imitating an expert's driving and parking maneuvers, facilitating the teaching of complex tasks to the agent through expert demonstrations ; and (3) incorporating curriculum learning, which involves breaking down the learning task into sub-tasks to aid the agent in training effectively in complex environments [Codevilla, 2019].

Before delving into the proposed algorithm, we provide an overview of the key concepts utilized in this study : curiosity reward, IL, and curriculum learning.

3.1 Curiosity Reward

In the context of the Intrinsic Curiosity Module (ICM) [Pathak, 2017] used for reward learning, the parameter ϕ_{s_t} represents the feature representation of the current state s_t. This feature

representation is obtained by encoding the essential information from the observation s_t into a lower-dimensional feature space. The process of transforming the observation into a feature vector helps extract relevant information required for the agent's actions while discarding redundant or less informative details.

The parameter η in the equation for the intrinsic reward r_t^i is a hyperparameter that controls the scale of the reward signal. It determines the weighting of the prediction error in influencing the agent's behavior. A higher value of η amplifies the impact of the prediction error, leading to more significant changes in the agent's exploration behavior based on curiosity.

In the equation for r_t^i, the term $\hat{\phi}(s_{t+1})$ represents the predicted feature vector of the following state s_{t+1}, obtained from the forward dynamics model. On the other hand, $\phi(s_{t+1})$ denotes the real feature vector of the following state s_{t+1}. The intrinsic reward r_t^i is computed as half of the squared Euclidean distance between the predicted and real feature vectors, scaled by the parameter η. This reward serves to incentivize the agent to explore states that are less predictable based on its current knowledge, thereby promoting curiosity-driven learning.

$$r_t^i = \frac{\eta}{2} \left\| \hat{\phi}(s_{t+1}) - \phi(s_{t+1}) \right\|_2^2 . \tag{5.1}$$

3.2 Imitation Learning

To foster the agent's exploration, the ICM is employed as an algorithm for reward learning. ICM transforms observations into a feature vector, capturing only the essential information necessary for the agent's actions.

The agent's action $\hat{a}_t$ is predicted using a self-supervised neural network on a proxy inverse dynamics task, leveraging the agent's current and subsequent states (s_t and s_{t+1}). Subsequently, a forward dynamics model is trained with the feature space to predict the future representation of the subsequent state $\phi_{s_{t+1}}$, based on the feature representation of the current state ϕ_{s_t} and the action $\hat{a}_t$. Finally, the prediction error of the forward dynamics model is utilized as an intrinsic reward r_t^i to enhance the agent's curiosity. Curiosity is the difference between the predicted feature vector of the subsequent state and the actual feature vector of the subsequent state.

3.2.1 Behavior Cloning

Behavior Cloning (BC) is an effective IL algorithm that treats the problem as a supervised learning task. Given a dataset of expert demonstrations, BC aims to learn a policy that maps states to actions like the expert. In the BC approach, the agent does not interact with the environment ; instead, it learns the policy by minimizing the negative log-likelihood of the expert's trajectories under the agent's policy. This negative log-likelihood quantifies how probable the agent's policy generated the expert's trajectories. A small negative log-likelihood indicates a good match between the agent's and the expert's policies. The supervised training loss for BC is formulated as follows :

$$\min_{\pi} J_{\mathrm{BC}}(\pi) := -\frac{1}{N} \sum_{k=1}^{N} \log \pi(a_k \mid s_k) \tag{5.2}$$

where N is the number of expert trajectories, s_k represents the kth state in the trajectory, a_k is the kth action in the trajectory, and $\pi(a_k \mid s_k)$ denotes the probability of the agent taking action a_k in state s_k according to its policy.

The goal is to minimize this loss function by finding a policy π that maximizes the likelihood of the agent taking actions that align with those taken by the expert in the dataset.

BC presents a straightforward and efficient approach to IL. However, it may need more brittleness and limited generalization to new environments. The reason is that BC solely focuses on imitating the expert's actions and does not learn a model of the environment.

Despite its limitations, BC is a valuable starting point for IL and can be highly effective in simple environments. It provides a practical means to learn a competent policy by leveraging the expertise demonstrated in the expert demonstrations.

3.2.2 Generative Adversarial Imitation Learning

GAIL consists of two neural networks : the generator G_θ and the discriminator D_φ. The generator produces a distribution of state-action pairs, while the discriminator, acting as a secondary neural network, generates a distribution from expert demonstration samples φ. GAIL aims to learn near-optimal behaviors directly from expert demonstrations and self-exploration without requiring task-specific reward functions by encouraging the generator to confuse the discriminator. This approach allows the agent to mimic a policy that exhibits states and actions similar to those demonstrated by experts, resulting in human-like behavior [Blondé, 2019].

The GAIL optimization process involves minimizing the min-max cross-entropy objective, as presented in Equation (5.3) :

$$\min_{\theta} \max_{\varphi} V(\theta, \varphi) \triangleq \mathbb{E}_{\pi_\theta} \left[\log\left(1 - D_\varphi(s, a)\right)\right] + \mathbb{E}_{\pi_e} \left[\log D_\varphi(s, a)\right] \tag{5.3}$$

Here, the solution involves taking a gradient step concerning φ to increase $V(\theta, \varphi)$ and performing a PPO optimization step on θ to decrease $V(\theta, \varphi)$. The discriminator D_φ is trained to distinguish between the policy π_θ generated by the generator and the expert policy π_e. On the other hand, the objective of the generator G_θ is to learn $\mathit{\Pi}_\theta$ using the output of the discriminator as a reward. This adversarial process enables GAIL to effectively imitate expert behavior and learn robust policies for complex tasks.

3.3 Dynamic Proximal Policy Optimization with Imitation Learning Algorithm

The DPPO-IL algorithm is designed in two distinct phases. In the first phase, BC is employed for a predetermined number of steps denoted by H_{BC}, to find an initial imitation policy from a set of expert trajectory demonstrations $\tau_E \sim \pi_E$ (see Algorithm 9, Line 1).

These expert trajectories are derived from pre-recorded demonstrations of an expert attempting to navigate towards an empty parking spot and park the car. These demonstrations contain information on the observations, actions, and rewards for a given agent during the recording session.

The process begins by generating a set of agent transition states from the demonstrated expert trajectory sequence states, denoted as $\mathcal{T}_{\mathrm{E}}^{a}$ (Algorithm 9, Line 5), which is then utilized to learn the agent's inverse dynamic model. For each transition $(s_t, s_{t+1}) \in \tau_e^a$, the agent-specific inverse model $\mathcal{M}_\theta(s_t, s_{t+1})$ is computed in the InverseModelLearning step, where the agent learns its unique inverse dynamics model [Torabi, 2018] (Algorithm 9, Line 6). This is formulated as a maximum-likelihood estimation problem given by :

$$\theta^* = \arg\max_{\theta} \prod_{i=0}^{N} p_\theta \left(a_i \mid s_i^a, s_{i+1}^a\right), \tag{5.4}$$

where p_θ is a conditional distribution over the action created at a specific state-transition in $\mathcal{M}_\theta$. The neural network is trained using an Adam optimizer [Kingma, 2014] to find θ^* in Equation (5.4), with state-transitions as input and the individual means and standard deviations for each action dimension as output. The maximum likelihood action from $\mathcal{M}_\theta$ is then utilized to construct a state-action set $\{(s_i, a_i)\}$ (Algorithm 9, Line 7). BC is then applied to find the parameter θ in π_θ that matches $\{(s_i, a_i)\}$ to learn the imitation policy. θ^* can be found using the maximum likelihood estimation in Equation (5.5) :

$$\theta^* = \arg\max_{\phi} \prod_{i=0}^{N} \pi_\theta \left(\tilde{a}_i \mid s_i\right), \tag{5.5}$$

Moving on to the second phase, the discriminator parameter is updated and trained as a binary classifier to distinguish between the agent's policy and the expert's policy (Algorithm 9, Line 12). Subsequently, the policy π is updated via a PPO gradient method, and the policy is optimized with the intrinsic reward r_t^i (Algorithm 9, Line 13). The DPPO-IL algorithm extends PPO, the BC algorithm [Torabi, 2018], and GAIL [Ho, 2016].

3.4 Curriculum Learning

Transfer learning is a significant concept in DRL, where the agent does not solely focus on learning the final target task. However, instead, it learns a sequence of subtasks that lead up to the ultimate target task. Each sub-task is treated as a Markov Decision Process (MDP) from which knowledge can be transferred to other sub-tasks, thereby accelerating the learning process. This knowledge transfer can take different forms, including value functions [Taylor, 2005], options [Soni, 2006], samples [Lazaric, 2008], policies [Fernandez, 2010], or models [Fachantidis, 2013].

The sequence of sub-tasks is organized as a curriculum, representing a set of prior experiences obtained through training on various tasks. By accumulating these experiences, the agent can improve performance and accelerate training on more challenging tasks. In our approach, we adopt a task-level curriculum.

The curriculum is constructed as a directed acyclic graph denoted as C=($\mathcal{V}$, $\mathcal{E}$, g, $\mathcal{T}$), where $\mathcal{T}$ is the set of tasks, and $\mathcal{D}^{\mathcal{T}}$ represents the set of all possible transitions for each task in $\mathcal{T}$. For each $m_i \in \mathcal{T}$, $\mathcal{V}$ is the set of vertices associated with a sample for a single task, $\mathcal{E}$ is the set of directed edges and $g : \mathcal{V} \to \left\{\mathcal{D}_i^{\mathcal{T}} \mid m_i \in \mathcal{T}\right\}$ is the mapping function.

Figure 5.3 illustrates a curriculum learning approach employed to train an automated parking

Algorithm 9 DPPO-IL algorithm

Input Expert trajectories $\tau_e \sim \pi_e$ where $\tau_e = (s_1, a_1.s_2, a_2, ..)$
params θ of policy π_θ, params w of discriminatorD_φ, H_{BC}
Output policy π_θ
while $H_{BC} \leq \tau_e$ **do**
    Generate set of agent state transition $\mathcal{T}_{\mathrm{E}}^{a}$ from the demonstrated Expert trajectories τ_E
    $\mathcal{M}_\theta = \mathrm{InverseModelLearning}(\mathcal{T}_{\mathrm{E}}^{a})$
    Deduce stat-actions pair $\{(s_i, a_i)\}$ from $\mathcal{M}_\theta$
    $\pi_\theta = \mathrm{BC}(\{(s_i, a_i)\})$
end while
for $i = 0, 1, 2, \ldots$ **do**
    Sample trajectories $\tau_i \sim \pi_\theta$
    Update the discriminator parameters φ_i to φ_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\varphi \log\left(1 - D_\varphi(s,a)\right)\right] + \hat{\mathbb{E}}_{\tau_E}\left[\nabla_\varphi \log\left(D_\varphi(s,a)\right)\right]$$

    $\pi_\theta =$ Update policy via policy gradient method of PPO on the intrinsic reward r_t^i inferred by

$$r_t^i = \frac{\eta}{2}\left\|\hat{\phi}\left(s_{t+1}\right) - \phi\left(s_{t+1}\right)\right\|_2^2$$

end for

system agent. Curriculum learning entails progressively increasing the complexity of training tasks as the agent becomes more proficient. The breakdown of the curriculum depicted in the image is as follows :

Task 01 : Initially, the agent learns to plan a trajectory within an unobstructed space. This task typically involves exploring various paths and assessing their smoothness or efficiency.

Task 02 : Once the agent demonstrates effective trajectory planning, it advances to a more demanding scenario involving obstacle avoidance. This task may require integrating obstacle detection capabilities and adapting trajectory planning based on the surrounding environment.

Task 03 : Lastly, the agent undertakes the comprehensive parking task, which encompasses trajectory planning, obstacle avoidance, and precise maneuvers to successfully park the vehicle.

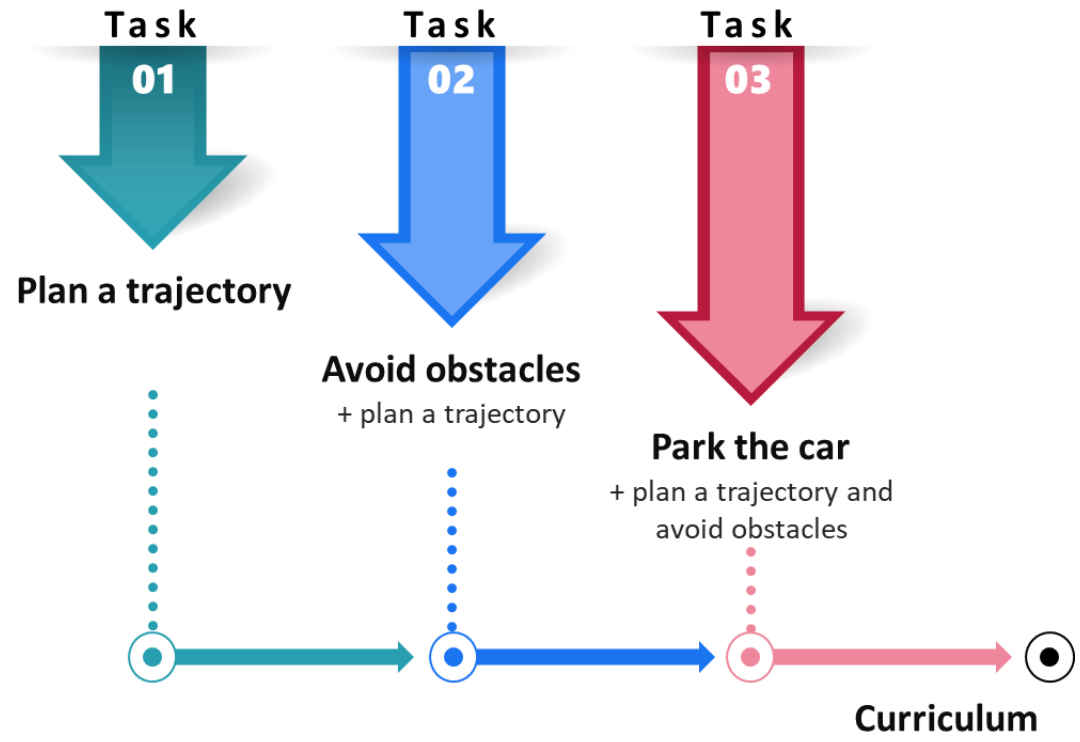


FIGURE 5.3 – Curriculum Learning

4 Experiments and Results

4.1 Test Setup

The system takes the agent's state space, representing its observations, as input. The agent is continuously trained to associate the observed state with the most appropriate action to progress toward its goal and maximize the expected average return in a single task.

The DRL script specifies that three float values are produced as output, each ranging between $[-1, 1]$, representing percentages. These values are learned by the agent and subsequently scaled by the Maximum motor torque, Maximum steering angle, and Maximum brake torque parameters in the Unity vehicle model [1], developed by Unity Technologies. Consequently, the agent receives three action variables, namely "motorTorque," "steering angle," and "brakeTorque."

The first experiment tests and compares the performance of three algorithms, PPO, SAC, and POCA, in a simple scenario. The results are exported in a ".onnx" open-source file format [Bai, 2019], ensuring compatibility between AI models and various associated frameworks like PyTorch and TensorFlow. After the training is completed and the policy is optimized, the obtained model (.onnx) can be associated with an agent for inference in the developed Unity environment, allowing the model's accuracy to be tested.

During the training phase, training statistics are saved in a Tensorboard, a tool developed by TensorFlow [Singh, 2020], which facilitates the visualization of the evolution of hyperparameters

1. https ://unity.com/

and other training parameters.

4.2 Agent

The vehicle in this study is accurately modeled to resemble the real-world Sodivehicle RSX 04 in terms of dimensions, weight, size, radius, wheel size, and other relevant aspects.

Special attention was given to ensuring a realistic simulation of a practical scenario. TThe vehicle is designated as the agent„ and its physics are meticulously represented using Unity's "WheelColliders" documentation [2]. The agent possesses a chassis modeled with a "Box Collider" and four "Wheel Colliders" that simulate the wheels. The front wheels are capable of changing direction and braking. To facilitate the vehicle's forward movement, a "Torque" is applied. The agent is equipped with an autonomous parking script to learn from and interact with its environment.

Additionally, the vehicle is equipped with a radar system that provides observations of the azimuth plane within a range of ±60°, with a resolution of 15° between two objects. The maximum range of the radar is set at 80 meters. These radar-related parameters are conveniently managed using Unity's "Raycast Observations" component, which is integrated into the vehicle's setup. A "ray cast" is essentially a radar originating from the vehicle's center that detects the distance at which it encounters an object and identifies the type of the detected object.

4.3 Environment

The agent's environment consists of a parking area that includes the previously described vehicle and a designated parking space, which serves as the agent's objective. The parking space is visually represented by white lines forming a rectangular shape with a green square positioned in the center. Figure 5.4 illustrates the parking scenario, where the white blocks represent obstacles, and a humanoid figure signifies the presence of a pedestrian.

4.4 Experiments

In the preliminary agent training sessions, we aimed to identify the optimal state space and reward function for training the DPPO-IL algorithm. During the initial trials on the simple scenario, the DPPO-IL algorithm achieved a satisfactory success rate with the following configuration :

— State space : The agent's heading angle H_a (1 value) ; its position P_a (3 values) ; its velocity components in the plane V (2 values) ; the position of the parking space P (3 values).

— Reward function : The agent receives a reward based on the following conditions :

$$\begin{cases} -1 & \text{if } d(P_a, P) > 30 \text{ meters OR } t_p \geq 1 \text{ minute.} \\ +1 & \text{if } d(P_a, P) < 0.5 \text{ meters} \end{cases}$$

2. https ://docs.unity3d.com/Manual/class-WheelCollider.html

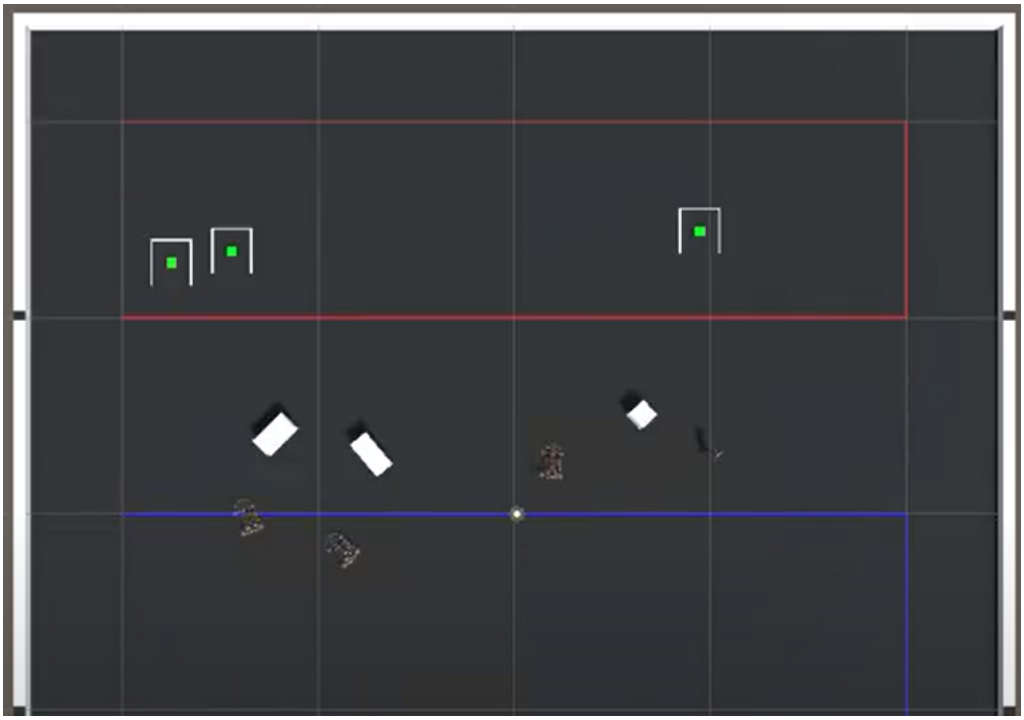


FIGURE 5.4 – 3 karts on the scene scenario

Here, t_p denotes the allocated parking time, and $d(P_a, P)$ represents the distance between the agent and the parking space.

However, despite achieving this policy within one hour of training, it was observed that the agent needed to exhibit proper alignment with the parking space and tended to park in reverse circles, indicating that further refinement and adjustments are needed in the training process.

To further optimize the system, several recommendations were followed during the training of the DRL model :

— Increasing Parallel Agents : Increasing the number of agents working in parallel allows for collecting a more extensive and more diverse set of experiences, enabling the exploration of a broader range of states and accelerating the training process. For instance, by increasing the number of agents from 8 to 256, the learning time decreased from 4 hours and 30 minutes to 1 hour and 30 minutes.

— Adjusting Penalty for Bad Alignment : Applying a higher penalty for lousy alignment between the agent and the parking space improved the agent's performance. This adjustment helped the agent better align himself during the parking task.

Fig. 5.5 compares the performances of the three DRL algorithms (PPO, POCA [Cohen, 2021], and SAC [Haarnoja, 2018]). The pink curve represents PPO, the blue curve corresponds to POCA, and the red curve represents SAC. The comparison demonstrates that PPO and POCA outperform SAC. Specifically, the reward for SAC was negative after 600,000 steps, whereas PPO was thoroughly trained in less than 300,000 steps, and POCA took an additional 100,000 steps to reach full convergence after 400,000 steps.

Based on this initial experiment, it can be confirmed that PPO, in this particular scenario, outperforms both SAC and POCA. PPO achieved a remarkable success rate of 98% (2,165 out of 2,209 trials), of which 97.5% (2,154 out of 2,209 trials) were aligned and respected the ISO-16787 :2017 norms for parking.

The results of this experiment suggest that PPO shows promise as a practical algorithm for training DRL agents in autonomous parking tasks. Nevertheless, further experiments are required to validate these findings and evaluate PPO's performance in various scenarios.

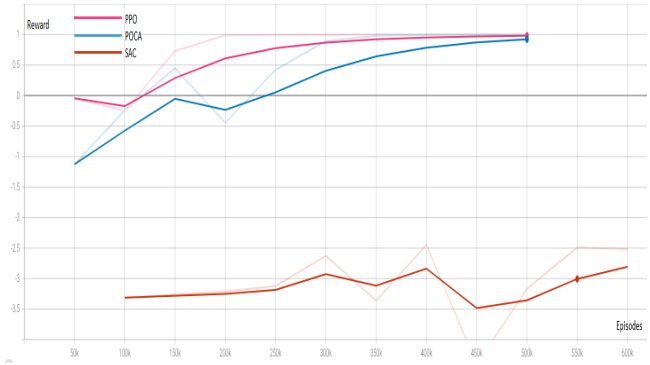


FIGURE 5.5 – DRL algorithm comparison

The second experiment was conducted by employing the PPO algorithm with the following configuration :

— State space : The agent's heading angle (H_a), its position (P_a), its velocity components in the plane (V), the position of the parking space (P), and the 3D components of the remaining distance between the parking space and the agent. All observations are normalized.

— Reward function : The agent receives a reward based on the following conditions :

$$\begin{cases} -1 & \text{if } t_p > 1 \text{ minute OR } \Gamma < 0 \text{ OR } |\delta_{sw}| > \pm 15^\circ \text{ OR } |e_y| = \pm 10^\circ. \\ +1 & \text{if } d(P_a, P) < 0.5 \text{ meters AND } t_p \geq 1 \text{ second.} \end{cases}$$

Where t_p denotes the allocated parking time, Γ is the action torque, δ_{sw} is the wheel alignment with the parking spot, e_y is the alignment error with the parking space, and $d(P_a, P)$ denotes the distance between the agent and the parking space.

As shown in Fig. 5.6, PPO took 8 million steps to converge and did not reach the maximum reward of 1. Due to the random occurrence of various static and dynamic obstacles, the success rate decreased to 74.8%. As a result, the scenario was retested by incorporating IL and curiosity reward into PPO.

These experiments' results demonstrated that adding IL and curiosity reward to PPO significantly improved the agent's performance. The agent could converge in less than 5 million steps and achieve a success rate of 94.2% in the presence of obstacles.

These results suggest that the combination of IL and curiosity reward shows promise as a practical approach for training DRL agents for autonomous parking. However, further experiments are needed to confirm these findings and evaluate this approach's performance in different

scenarios.

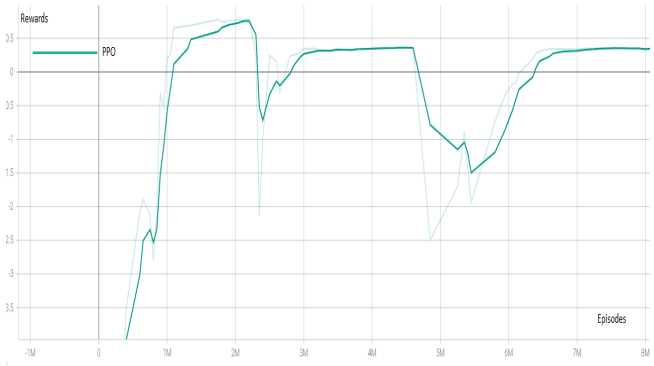


FIGURE 5.6 – PPO main scenario

Fig. 5.7 illustrates the experiments' results with different combinations of IL, Curiosity Reward, and PPO. The orange curve represents DPPO, the red curve represents DPPO with GAIL, and the blue curve represents DPPO-IL. The graph shows that adding Curiosity Reward to DPPO significantly improved the agent's performance. The agent achieved a cumulative reward of 1 after 1.5 million steps, compared to 8 million without Curiosity Reward. This result suggests that Curiosity Reward encouraged the agent to explore and learn new things. The inclusion of GAIL in DPPO-IL further enhanced the agent's performance. The agent could converge in 750,000 steps, compared to 3.5 million without GAIL. This finding indicates that GAIL assisted the agent in learning a more robust policy to handle dynamic obstacles. The final configuration, which combines Behavioral Cloning (BC), GAIL, and Curiosity Reward with PPO, achieved the best performance. The agent learned in less than 350,000 steps, attaining a 92% success rate. This outcome highlights the promising potential of combining these three techniques for training DRL agents for autonomous parking in the presence of dynamic obstacles.

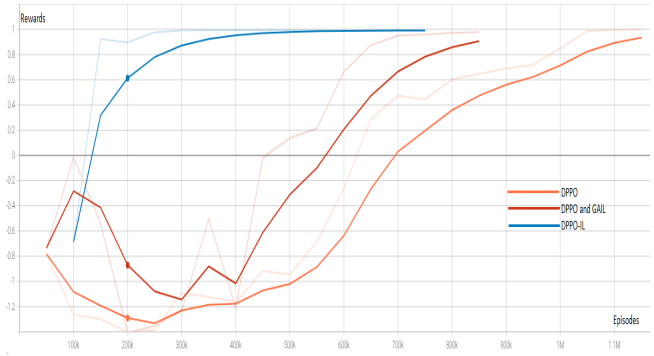


FIGURE 5.7 – PPO with IL and curiosity

The third experiment extends the previous scenario in Fig. 5.4 by introducing three vehicles in the scene, each with three different parking spots that do not belong to anyone. The experiments use the same state space as in the previous scenario. Each vehicle searches for the closest parking spot and parks the car properly, ensuring that the selected spot is unoccupied.

The results of the experiments with DPPO-IL are presented in Fig. 5.8. As observed, the agent was able to converge in 60 million steps, but the reward exhibits significant fluctuations due to the presence of highly dynamic objects in the environment. The success rate reached 81%, where 90% were aligned with the parking spot.

This outcome indicates that adding more vehicles to the scene makes the environment more challenging for the agent. The agent must be more reactive, frequently re-exploring the available parking spots and re-planning its trajectory. Moreover, this scenario imposes additional stress on the acceleration and braking actions.

Although PPO is one of the best DRL algorithms, the highly dynamic environment proves to be demanding, even with the inclusion of intrinsic rewards and IL. This suggests that further research is required to develop DRL algorithms that effectively handle such challenging environments.

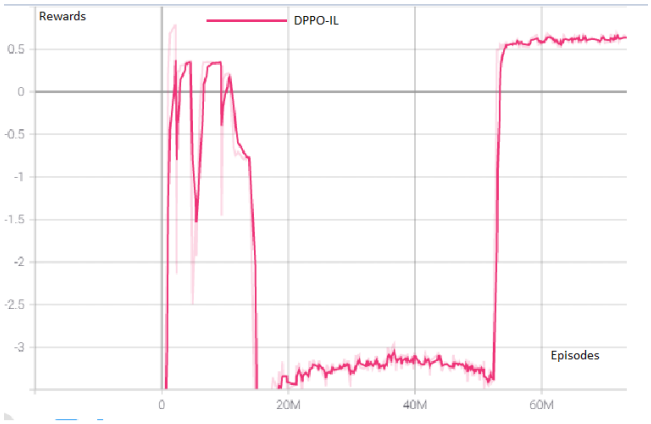


FIGURE 5.8 – 3 karts on the scene DPPO-IL results

To address the challenges of training a DRL agent in a highly dynamic environment, we have adopted curriculum learning [Narvekar, 2020]. This approach enables the agent to learn concepts before exploiting the complex environment's reward signal. The primary goal of these concepts is to reduce random explorations and foster the adoption of more reliable actions. Instead of conducting the training process as a single shot, we have decomposed the parking task into sub-tasks. The first sub-task involves identifying and planning a trajectory toward an unoccupied parking space. The second sub-task is to control the acceleration and braking actions, avoiding any types of obstacles. Lastly, the third sub-task focuses on parking the car with the required alignments. Each sub-task is deemed successfully learned when the agent manages to double its accumulated reward for four million episodes, as depicted in Figure 5.9. The non-linear learning curve of the rewards in Figure 5.9 is expected, as rewards are anticipated to improve continuously throughout the training. Curriculum learning for DPPO-IL has proven effective, delivering superior results and a faster learning rate than DPPO-IL from scratch, as shown in Figure 5.9. The agent achieved a 90% success rate, where 97% of them aligned with the parking spot with an inclination angle $> \pm 0{,}2°$ and a deviation > 0.1m.

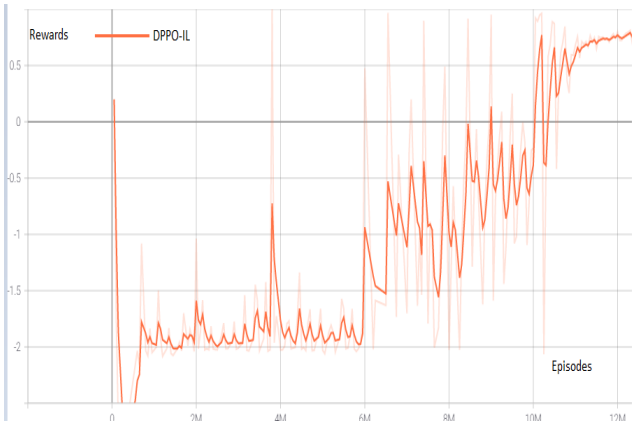


FIGURE 5.9 – 3 karts on the scene with curriculum learning

5 Conclusion

In this chapter, we have introduced DPPO-IL, a DRL framework tailored for end-to-end automated parking systems, utilizing the Proximal Policy Optimization algorithm. DPPO-IL enables the agent to explore empty parking spaces, learn optimal acceleration, braking, and rotation actions, and successfully park the vehicle in random parking spots while avoiding static and dynamic obstacles.

We comprehensively studied various DRL algorithms, including PPO, SAC, and POCA. Subsequently, we trained the agent in two distinct scenarios, refining the DRL algorithm by implementing dynamic reward function adjustments through intrinsic reward signals. Furthermore, we optimized training time by incorporating BC and Generative Adversarial IL.

The agent achieved a 90% success rate through these improvements in a highly complex environment, facilitated by adopting a task-specific curriculum learning strategy.

In the following chapter, we will address the design of a safe decision-making system for end-to-end urban driving, aiming to further enhance the capabilities of autonomous vehicles in complex and dynamic urban environments.

Chapitre 6

Guided Hierarchical Reinforcement Learning for Safe Urban Driving

Contents

1 Introduction

Today, with the availability of affordable sensors (Camera, LiDAR, IMU, GPS) and the introduction of cutting-edge software technologies (artificial intelligence, machine learning, computer vision), the prospect of fully autonomous vehicles (AV) has never been closer. However, cars must still attain level five of the Society of Automation Engineers (SAE). Most autonomous vehicle (AV) systems use a hand-crafted modular architecture [Xu, 2017]. However, the modular architecture is criticized for showing poor accuracy in highly interactive environments, such as urban driving. These models are tightly interconnected, which makes them expensive to scale and maintain. These limitations are bypassed by adopting end-to-end architectures, in which a driving policy is learned and generalized without human intervention [Xu, 2017]. The learned driving policy can also be continuously tuned with each driving attempt to achieve human-level performance. A safe driving policy remains an open challenge where the complexity surpasses a few well-defined tasks (e.g., moving box robot). The three main categories of end-to-end AV driving policy are : rule-based methods [Furda, 2011], imitation learning (IL) [Bansal, 2018][Xu, 2017][Chen, 2019a] and reinforcement learning (RL) [Wolf, 2017]. The rule-based methods are human-designed predetermined rules structured to achieve the best driving policy by selecting maneuvers and then planning the trajectory [Furda, 2011]. Despite the popularity of rule-based systems, manual rule encoding can put a strain on system engineers as they must anticipate all the crucial and possible rules for each driving scenario [Wolf, 2017]. IL-based methods are an effective alternative where the driving policy is learned directly and supervised by mimicking expert demonstrations as training sets. However, these methods require large amounts of labeled training data. To limit the time-consuming hand-labeled data, solutions that use deep RL (DRL) for end-to-end driver policy learning have been applied to simple driving scenarios, like lane keeping, steering control, and managing the acceleration [Kendall, 2019]. However, they suffer from a cold start and require extensive training before converging into a sound policy. In the case of end-to-end urban driving, the state space is typically vast, encompassing details about the road, surrounding vehicles, pedestrians, and traffic. Moreover, the exploration can be dangerous in urban driving scenarios and may result in collisions.

Recent studies have shown that Hierarchical Reinforcement Learning (HRL) is more suitable for urban driving [Bronstein, 2022]. HRL helps by breaking the task into smaller sub-tasks with more straightforward state space, thus reducing the required exploration [Guo, 2021]. On the other hand, IL can help with the cold start issue by providing a pre-trained expert policy that helps guide the agent's actions. Instead of providing the expert's demonstrations as action recommendations, modeling them as rules increases the information per interaction and does not limit the intake to the current state. This more informationally-rich interaction method improves the agent's performance compared to existing methods. Combining HRL and IL in urban driving can be a powerful approach to improve the efficiency of the driving task. However, HRL may suffer from the same safety issues as RL, such as exploration of unsafe actions and failure to generalize to new situations [Chen, 2019a]. Integrating a rule-based system as a safety mechanism can be a potential solution to address safety concerns in critical situations when using an HRL

approach for urban driving. Guided by the rules, the agent can learn to follow specific guidelines or constraints during decision-making, reducing the risk of accidents or dangerous maneuvers. Moreover, the rule-based system can provide a fallback option to ensure the safety of the driver, passengers, and other road users. However, it is essential to note that incorporating a rule-based system may limit the agent's adaptability to new situations and may require additional engineering effort to define the rules accurately. Therefore, it is crucial to balance the trade-offs between safety and adaptability when designing an HRL-based autonomous driving system.

This chapter proposes a Guided Hierarchical Reinforcement Learning (GHRL) approach based on vision and localization for end-to-end urban driving. We developed an approach to handle navigation and affordance prediction in autonomous driving scenarios with a two-phase training strategy. In the first phase, we employed a Autoencoder (AE) to learn a lower-dimensional representation of the visual sensor's input. This representation was further enriched with localization and waypoints as navigation input. Moreover, the AE predicted various affordance features, such as distance to impact or light traffic status. These features were inputs for the HRL agent instead of the raw sensor data.

Our approach involves training a high-level master policy and several low-level sub-policies, each with a specific goal, to handle the driving task. Initially, we divided the task into four maneuvers - driving in the lane, right and left lane change, and braking - before training each action's sub-policy. The master policy selects the appropriate sub-policy based on the current conditions. We created different state spaces and reward functions for each sub-policy.

To accelerate the training process, we injected expert demonstrations expressed in Answer Set Programming (ASP) [Brewka, 2011] formalism into the learning process, which guides the Proximal Policy Optimization (PPO) exploration policy. These rules are treated as on-policy, meaning that the agent generates them. When making decisions, the agent considers the injected rules and its learned policies, combining the information from both sources.

In dangerous situations, exploration by the agent to learn the best policy might not be feasible and could even result in catastrophic consequences. Therefore, the system relies solely on ASP rules to ensure safety. This approach guarantees the system adheres to predefined safety constraints, preventing potential pedestrian harm. By incorporating a rule-based system in the agent's decision-making process, the agent can benefit from both the exploration of the learned policy and the safety of the rule-based system. Specific situations or events, such as detecting an obstacle or predicting a dangerous situation, trigger the switching between the two systems. We evaluated our method on urban driving scenarios using Carla's simulator [Dosovitskiy, 2017] and demonstrated its effectiveness in handling various challenges, such as traffic lights and static and dynamic obstacles.

Hence, the main contributions of this chapter can be summarized as follows :

— We proposed GHRL, a model-free on-policy RL algorithm. The algorithm employs PPO and is guided by expert demonstration rules expressed in ASP.

— In situations where safety is of utmost importance, the system automatically switches to ASP rules integrated into the agent's decision-making process to take the appropriate action in critical situations.

— We have studied extensive parameters and performed ablation studies on reward shaping.

— The agents can learn efficient driving policies in the CARLA simulator that exhibits a wide range of urban behaviors like lane-following, handling intersections or traffic lights, and avoiding static or dynamic obstacles. The framework is adequately justified using the Carla NoCrash benchmark.

The rest of this chapter is organized as follows : Section 2 is dedicated to the contribution of this chapter, Section 3 details all the experiments, and finally, Section 4 concludes and gives some perspectives.

2 Guided Hierarchical Reinforcement Learning

In the GHRL paradigm, the agent endeavors to tackle the intricate hierarchy of subtasks, a structure where higher-level policies dictate the allocation of focus to specific subtasks, while lower-level policies dictate the precise methodologies to execute each subtask. This approach is grounded in the observation that human drivers inherently disassemble complex driving tasks into a hierarchy of subtasks, encompassing actions like route planning, lane changes, and braking. By cultivating an ability to resolve each subtask independently, the agent can glean a driving policy that balances safety and efficiency, ultimately facilitating its capacity to navigate real-world driving scenarios. The framework utilized within GHRL for learning hierarchical policies is the option-critic (OC) framework [Hutsebaut-Buysse, 2022]. In OC, the agent acquires a repertoire of options, where each option denotes a sequence of actions to achieve a distinct goal. These options are assimilated through a reinforcement learning mechanism, often Q-learning, while a critic, functioning as a value function, assesses the anticipated returns associated with each option. The critic's feedback is instrumental in directing the refinement of the options by supplying information about their respective values. In the context of GHRL, we have harnessed the OC framework to engender a hierarchy of driving options. The high-level options encapsulate various driving maneuvers, like lane changes and braking, that the agent can execute. Correspondingly, low-level options delineate the specific actions essential for the execution of each maneuver. The master policy governs the selection of the most suitable high-level option, predicated on the prevailing environment state. Subsequently, the agent systematically executes the sub-policies, orchestrating a hierarchy of actions. This OC framework has several advantages over alternative HRL strategies. Primarily, it entails a comparatively straightforward learning process, easing the burden of acquiring effective hierarchical policies. Moreover, the OC framework demonstrates its efficacy in mastering tasks that may resist conventional reinforcement learning methods. Furthermore, its adaptive nature lends itself well to scenarios marked by environmental fluctuations, allowing policies to remain robust in the face of dynamic changes. In essence, the GHRL approach, founded on the principle of hierarchical task decomposition, employs the OC framework to empower agents with the capacity to navigate a complex array of driving tasks efficiently. Through the synthesis of high-level and low-level policies, this approach provides a flexible yet structured mechanism for autonomous agents to interact with and maneuver within their environment, fostering a delicate equilibrium between safety, efficiency, and

adaptability.

2.1 State

We aim to make the agent's training more efficient and effective in completing the navigation task with dynamic actors by defining the state space S to include sufficient environmental knowledge as features. In our approach, the inputs fed to the AE [Wang, 2020b] consist of all objects detected by Carla's semantic segmentation (including static and dynamic obstacles, traffic lights, sidewalks, and roads) and traffic light states. The AE utilizes an encoder and a decoder to extract latent variables from images and generate new ones by sampling these variables. Essentially, the encoder and decoder are neural networks trained on unlabeled images [Doersch, 2016]. The AE encoder's CNN includes 4x4 kernels, four filters (32, 64, 128, 256) convolutions, and a stride of 2, with ReLU activation functions. The last convolution output is flattened and fed into two fully connected layers of size Z_{dim}, which creates a vector z from a Gaussian distribution. To augment the vector z, we have added external state variables such as waypoint features w. w is a set of predefined points from CARLA that the vehicle must pass through to reach its destination. We have also included accurate localization by fusing GPS and IMU [Albilani, 2022], speed, and orientation as a matrix m, which predicts additional features to aid training, such as the distance to impact and distance to an incoming event (such as entering an intersection or stopping at a traffic light). Furthermore, augmenting the data variety includes lateral distance and angle with the optimal trajectory. This approach results in a more disentangled agent training process.

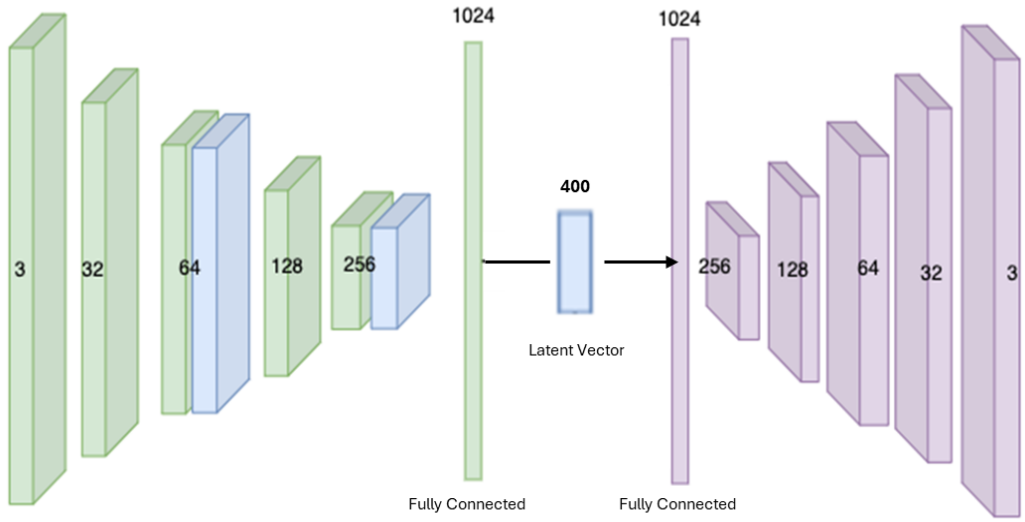


FIGURE 6.1 – Autoencoder architecture [Wang, 2020b]

2.2 Data Variety Augmentation

In pursuit of a comprehensive dataset for training and evaluating our proposed approach, we meticulously gathered a collection of 10,000 images, each with dimensions of 160x80 pixels,

encompassing both RGB and semantic segmentation information. This dataset was curated within the confines of Carla 0.9.8, employing an autonomous vehicle equipped with an autopilot system. The data collection process spanned diverse urban environments, encompassing city streets, highways, and intersections, to ensure the dataset encapsulated a broad spectrum of driving scenarios.

Initially, relying solely on autopilot for data collection yielded a dataset without errors. With its calibrated precision, the autopilot consistently maintained a centered lane position and adeptly evaded collisions with dynamic obstacles. However, this pristine dataset posed a challenge when training a reinforcement learning (RL) agent. The inherent predictability of the autopilot's behavior neglected the necessary exposure to unpredictable scenarios that an RL agent should adeptly handle.

To address this limitation, we followed a methodology inspired by the approach delineated in [Toromanoff, 2020]. By deliberately introducing perturbations during data collection, we emulated the effects of varying real-world conditions, thus injecting a degree of realism into our dataset. This procedure involved perturbing the positioning of the camera relative to the autopilot during image capture. This deliberate variation in camera position engendered a layer of noise within the images, rendering the dataset more representative of the inherent uncertainties and fluctuations experienced during actual driving scenarios.

In addition to manipulating camera positions, we supplemented the dataset by incorporating random perturbations to the steering and throttle inputs of the autopilot. This strategic inclusion of noise intentionally disrupted the autopilot's otherwise flawless trajectory and behavior. The rationale behind this introduction of noise was to cultivate an environment in which the RL agent could be trained to navigate and react effectively when confronted with unforeseen circumstances.

The amplitude of the introduced noise underwent careful calibration, encompassing values ranging from -0.1 to 0.1. This selection was a product of deliberate consideration aimed at achieving equilibrium. The intention was to maintain a level of magnitude that would allow the autopilot to carry out its driving maneuvers safely and with precision yet possess a substantial magnitude that could give rise to noticeable deviations. This calibration sought to emulate the inherent variability encountered in real-world driving scenarios, capturing the essence of unanticipated perturbations while upholding a fundamental sense of safety.

Our dataset construction approach meticulously intertwined the deliberate introduction of camera position variations and controlled noise within steering and throttle inputs. This interplay was orchestrated to construct a dataset that not only encapsulated a wide array of urban driving scenarios but also instilled the necessary elements of unpredictability, essential for nurturing the development of robust and adaptable RL-based driving policies.

2.3 Reward Shaping

We devised a reward function following a quadratic structure, characterized by a quadratic decrement for each sub-policy. This formulation of rewards serves as a driving incentive for the agent, motivating actions that align more closely with the optimal path while concurrently

imposing penalties for actions that veer further from this path. Additionally, we amalgamated factors encompassing speed, deviation, and angle within the reward function, contributing to a comprehensive assessment of the agent's actions.

Our experimental findings highlight reward multiplication's advantages over mere addition. This preference for multiplication is grounded in its ability to foster more effective learning within the agent. Consider a scenario where the agent governs a vehicle : the reward function could emphasize higher speeds and penalize significant deviations from the intended trajectory. Likewise, actions that facilitate the vehicle's alignment with a specific angle or trajectory could be subject to rewards.

Its impetus on maintaining specific performance parameters is central to the reward function's design. This encompasses sustaining a predefined speed, adhering to the lane center, and aligning harmoniously with the road configuration. However, the reward function judiciously bestows elevated rewards only when a set of prerequisites is partially met. This design aims to mold the agent's learning trajectory towards actions that concurrently optimize safety and efficiency. These rewards were inspired from [Vergara, 2019] and defined as follows :

$$R = f(r_v) * f(r_d) * f(r_\alpha) \tag{6.1}$$

where f is a quadratic function, r_v is the velocity reward function defined by :

$$r_v = \begin{cases} -10 & \text{on infraction} \\ \frac{v}{v_{\min}} & v < v_{\min} \\ 1 & v_{\min} \leq v < v_{\text{target}} \\ \left(1 - \frac{v - v_{\text{target}}}{v_{\max} - v_{\text{target}}}\right) & v \geq v_{\text{target}} \end{cases} \tag{6.2}$$

where $v_{\min}$ and $v_{\max}$ are the minimum and maximum allowed velocity (speed) according to the law, and v_{target} is the identified target velocity, r_d is the deviation distance reward function defined by :

$$r_d = \begin{cases} 0 & d >= 3 \\ 1 - \frac{d}{d_{\max}} & else \end{cases} \tag{6.3}$$

where d is the route deviation distance and $d_{\max}$ is the maximum threshold set to 3m. It indicates that r_d decreases with the increase of d. If d is larger than the maximum allowed value, then the agent will get a minimum reward of 0, and the deviation degree reward r_α is calculated as follows :

$$r_\alpha = \begin{cases} 1 - \left|\frac{\alpha_{\text{diff}}}{\alpha_{\max}}\right| & |\alpha_{\text{diff}}| < \alpha_{\max} \\ 0 & \text{else} \end{cases} \tag{6.4}$$

where $\alpha_{\max}$is the maximum threshold set to 90° and α_{diff} is the angle difference between the vehicle's forward vector and the current way-points forward vector.

2.4 GHRL Learning

Designing a single end-to-end policy for urban driving with numerous behaviors can be challenging and may lead to poor performance in completing the driving task. To address this issue, we propose an OC framework where high-level and low-level policies are trained synchronously. This framework allows for the incorporation of expert demonstrations by injecting rules into the learning process, which guide the agent's behavior

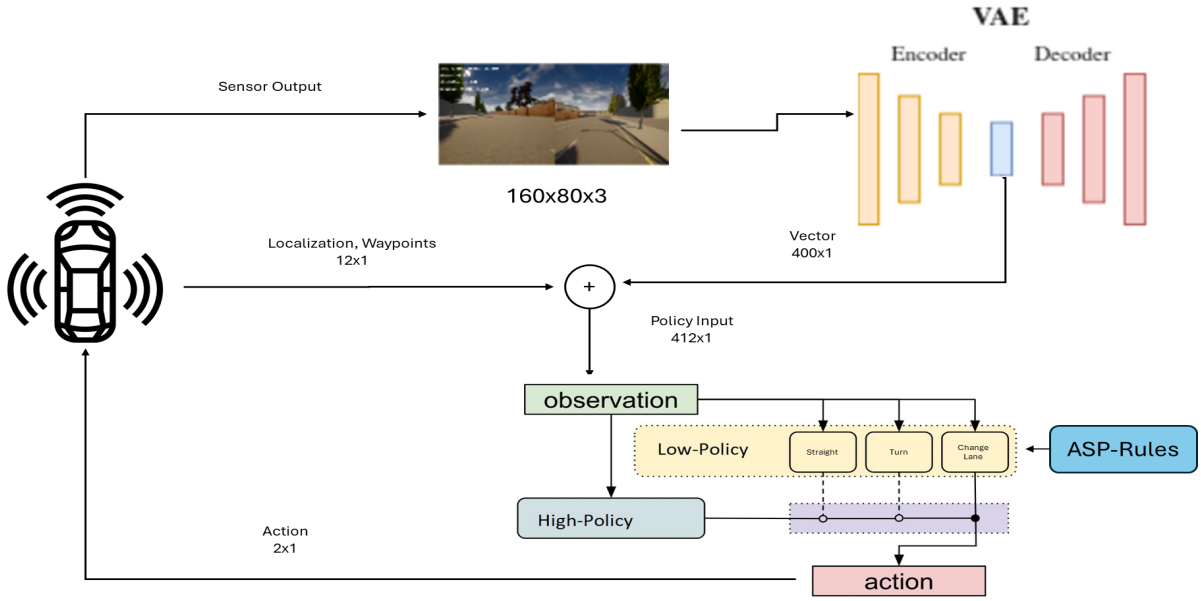


FIGURE 6.2 – High-level representation of GHRL

Fig.6.2 illustrates the pipeline to train GHRL. It integrates visual input from cameras, sensor data, and predefined rules to facilitate both high-level and low-level control of the vehicle. The system comprises several interconnected components : car images are processed by an Autoencoder (AE) to generate a compressed latent representation capturing essential visual features ; sensor fusion data, including localization and waypoints, complements the visual information. The AE acts as an encoder, providing the HRL agent with a compressed latent vector and sensor fusion data, enabling the agent to make high-level decisions regarding the vehicle's overall goal and direction. A separate low-level policy translates these decisions into concrete actions guided by predefined rules, ultimately outputting control signals such as steering angle and acceleration.

2.4.1 Low-Level Policies

We train low-level policies using PPO while incorporating expert demonstrations through ASP rules injected by a well-defined hyperparameter p, whose optimal value is determined through empirical study. This empirical investigation ensures that the hyperparameter p is carefully tuned to achieve the best performance and efficacy in training the low-level policies. These rules are considered "on-policy," indicating that the agent generates them during training. By integrating these expert demonstrations into the learning process, the agent can converge faster, reducing the time spent on exploration. Also, this integration allows the agent to benefit from

the expert's demonstrations and valuable knowledge for handling various driving scenarios effectively. We represent the set E of expert trajectories as $\tau = (s_0, \mathrm{R}_0, a_0, s_1, \mathrm{R}_1, a_1, ...)$, where each state s_i has a corresponding ASP rule R_i that determines the appropriate action a_i (explained in the following sub-section). Hence, the agent can effectively incorporate the expert's knowledge into decision-making as a pair (s_i,a_i) while having a single source of reward. The new modified policy π_θ^ϕ is used in the clipping function defined in[Schulman, 2017] as follows :

$$\pi_\theta^\phi = \begin{cases} \pi_\theta & \text{sampled from Environment with probability } \textit{1-p} \\ \pi_E & \text{sampled from Expert E with probability } p \end{cases} \tag{6.5}$$

In Algorithm 10, we have modified the PPO by including expert demonstrations in the training data and updating the policy to maximize the probability of taking actions guided by the expert (lines 6-9). The hyperparameter p (line 6) controls the probability of selecting an expert action rather than relying solely on the policy's output, and it can be gradually decreased as the policy improves during training (line 8). PPO optimizes the objective function L^{GPPO} (line 19), representing the expected return of the policy π_θ with respect to its parameters θ. This function estimates the cumulative reward anticipated from following the policy in the environment.

The optimization process in PPO employs gradient descent to iteratively update the policy parameters θ for enhanced performance. Over multiple epochs (K), the policy is optimized using minibatches of experiences sampled from the environment. The size of these minibatches (M) is usually smaller than or equal to the dimensionality of the observation space (N) multiplied by the episode length (T) (line 20).

During each epoch, the policy parameters θ are adjusted to minimize the objective function via gradient descent. The gradient of this function with respect to the policy parameters is computed through backpropagation within the neural network representing the policy (line 21).

Furthermore, PPO maintains an empty storage E to retain experiences sampled during optimization, essential for updating policy parameters in subsequent iterations and epochs. This ensures that the policy learns from a diverse array of experiences (line 22).

By integrating expert demonstrations, we provide the learning algorithm with constraints that steer it toward the desired behavior while allowing for exploring and discovering new approaches. This hybridization enables the algorithm to learn from the expert's knowledge and experiences, leading to more effective decision-making and better handling of various tasks.

2.4.2 High-Level Policies

The high-level master policy π_{high} is trained after completing all low-policies training. Algorithm 11 is an OC algorithm responsible for learning the different high-level intra-option policies $\pi_{o_t}(a_t \mid s_t, o_t)$ and the termination condition $\beta_{o_t}(s_t, o_t)$ for the option o_t at state s_t. In dangerous situations, φ, the agent will rely solely on the ASP rule set to make safe decisions. This rule set will guide the AV in executing appropriate actions to address the short-term goal, detailed in the upcoming sub-section. Dangerous situations are recognized by applying

Algorithm 10 GPPO Low-Level Policies

Initialize parameters θ, p
Initialize storage $\varepsilon \leftarrow \{\}$
for every update **do**
  for actor 1,2,...,N **do**
    Sample τ from expert trajectories E
    if p **then**
      for steps $1, 2, \ldots, T$ **do**
        $s_t, \mathrm{R}_t, a_t, r_t, s_{t+1} \sim \pi_E\left(a_t \mid s_t\right)$
        Store transition $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_t, a_t, r_t)\}$
      end for
    else
      for steps $1, 2, \ldots, T$ **do**
        Execute an action in the environment
        $s_t, a_t, r_t, s_{t+1} \sim \pi_\theta\left(a_t \mid s_t\right)$
        Store transition $\mathcal{E} \leftarrow \mathcal{E} \cup \{(s_t, a_t, r_t)\}$
      end for
    end if
  end for
  Optimize L^{GPPO} wrt θ, with K epochs and
  minibatches size $M \leq NT$
  $\theta \leftarrow \theta - \eta \nabla_\theta L^{GPPO}$
  Empty storage $\mathcal{E} \leftarrow \{\}$
end for

the Responsibility-Sensitive Safety (RSS) framework [Shalev-Shwartz, 2017]. RSS aims to establish a common understanding of safe driving behavior for AVs by providing a set of rules and guidelines governing AV behavior in various driving scenarios. The overarching goal of RSS is to ensure the safety of passengers, pedestrians, and other road users by promoting responsible driving practices in AVs.

The high-level policy over options is a ϵ-greedy form on approximating the option-value function $Q_\Omega(S_t, O_t)$, where Ω is the specific value function to a particular option o_t at state s_t. Further elaboration on dangerous situations will be provided in section 2.4.4.

OC trains the intra-option policies as follows (see [Guo, 2021] for more details) :

$$\frac{\partial L(\theta)}{\partial \theta_\pi} = \mathbb{E}\left[\frac{\partial \log \pi\left(a_t \mid s_t, o_t\right)}{\partial \theta_\pi} Q_U\left(s_t, o_t, a_t\right)\right], \tag{6.6}$$

where θ_π is the parameter of low-policies, and $Q_U\left(s_t, o_t, a_t\right)$ is the the option-value function. The gradient is calculated as follows :

$$\frac{\partial L(\theta)}{\partial \theta_\beta} = \mathbb{E}\left[-\frac{\partial \beta\left(s_t, o_t\right)}{\partial \theta_\beta}\left(A_\Omega\left(s_t, o_t\right) + \eta\right)\right], \tag{6.7}$$

where θ_β is the high-level policy termination parameter, and η is the deliberation cost. $A_\Omega(s_t, o_t)$ is the termination advantage. To update the option-value function is as follows :

$$Q_\Omega^{k+1}(s, o) = Q_\Omega^k(s, o) + \alpha\Omega \tag{6.8}$$

Algorithm 11 Learning High-Level Policies

Initialize external options o_i from low-policies
Initialize master policy π_Ω, option library Ω
Initialize the facts F
Initialize dangerous situations φ
add all pre-trained low policies o_i into Ω
for every update **do**
  if φ **then**
    Execute the logical program (P,F)
  else
    Choose o_t according to s and π_Ω
    Execute o_t according to low-policy π_{o_t} and β_{o_t}
    get s' and R_{t+1}, add (s, o_t, s', R_{t+1}) into buffer
    Update with SMDP Q-Learning
    $Q_\Omega^{k+1}(s,o) = Q_\Omega^k(s,o) + \alpha\Omega$
  end if
end for

2.4.3 Safety Specification

Pre-defined rules are designed to make safe decisions in longitudinal and lateral critical situations φ (explained in the following sub-section). A critical time interval in self-driving cars refers to a situation where the car's autonomous system fails to perceive or appropriately respond to a potential hazard in the surrounding environment, such as a pedestrian crossing the street or another vehicle suddenly changing lanes. During this interval, the ego vehicle c_{ego} will solely apply safety ASP rule-based policy to guarantee safe longitudinal decision-making.

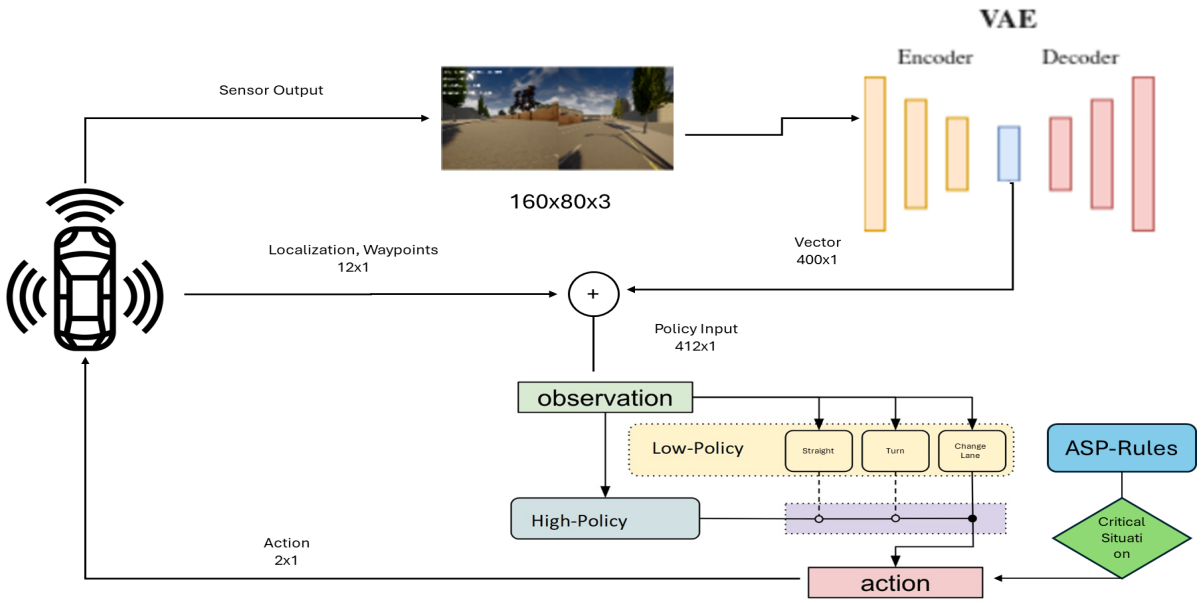


FIGURE 6.3 – Safety in critical situation

Fig.6.3 illustrates the pipeline during critical situations. The ASP rule engine assumes control over signals like steering angle and acceleration, which are provided by the HRL agent.

The environment mapping in the scene is transformed into predicates describing the objects present and their positions. These predicates are represented as facts F in ASP that form the input for the driving decision-making process. F contains, among other information, the

speed, the lane, the relative distance, the AV predicted trajectory, lane structure, intersection information, visible traffic signs, lights, and other detected objects. Depending on the facts F, the rules R are applied to produce decisions Y such as accelerating, braking, cruising, changing lanes, and turning left or right. In the following example, the sensors detect a pedestrian while the ego vehicle initiates a right turn. The system should identify this situation as critical, requiring longitudinal and lateral safety measures. Let's consider the following logical program P and $F = \{ego\_path(30.215, 3:05), obj\_path(Pedst1, 30.215, 3:05\}$ a set of atoms :

$$\begin{cases} r1 \text{ :abort\_select\_action(turn\_right, T) :-} \\ \text{ego\_path(EPath, T),} \\ \text{obj\_path(Oid, OPath, T),} \\ \text{intent(turn\_right, T),} \\ \text{path\_intersects(EPath, Oid).} \\ r2 \text{ :path\_intersects(EPath, Oid) :- ego\_path(EPath, T),} \\ \text{obj\_path(Oid,OPath, T'), T=T', EPath = OPath.} \\ r3 \text{ :brake\_conditions(T) :- intent(turn\_right, T),} \\ \text{abort\_select\_action(turn\_right, T).} \end{cases} \tag{6.9}$$

In this scenario, the logical program P defines a set of rules describing the behavior detected by the system of an AV intending to merge into the right lane "$intent(turn\_right, T)$" at time "T=3 :05". These rules consider the presence of pedestrians and intersections to make appropriate decisions. Rule 1 aborts the action "turn_right" at time "T" under certain conditions. The rule checks explicitly if the ego vehicle's path and the pedestrian's path intersect at time "T" using the 'path_intersects(EPath, Oid) predicate. If an intersection is detected, the AV avoids selecting the "turn_right" action to ensure pedestrian safety. Rule 2 defines the predicate 'path_intersects(EPath, Oid)' responsible for checking whether the ego vehicle's path ('EPath') and the pedestrian's path ('Opath') intersect at time "3 :05". The rule unifies 'ego_path(EPath, T)' and 'obj_path(Oid, OPath, T')' to determine if the paths intersect, enabling the AV to make informed decisions based on spatial relationships. Lastly, Rule 3 specifies the "brake_conditions" that are met at a time "T" if there is an "intent(turn_right, T)" and the ego vehicle's path and the pedestrian's path intersect at that time. This rule ensures that the AV applies the brakes when necessary to avoid collisions and adhere to the intent to merge into the right lane while ensuring pedestrian safety.

2.4.4 Longitudinal Critical Situations

Let c_{ego} and c_{fwd} be two cars such as c_{fwd} is followed by c_{ego} with a distance d. A longitudinal critical situation occurs when c_{fwd} brakes with action $a_{max,brake}$ whilst c_{ego} accelerates with $a_{max,acc}$ then it brakes with $a_{min,brake}$ until a collide with c_{fwd} during a response time τ_{res}. Otherwise, the longitudinal situation is safe. Let t_d be a period during which the situation is critical and $d \leq d_{min}$. The interval $[t_d, t_d + \tau_{res}]$ becomes a critical interval where the ASP safety

rules are applied. The mathematical proof of d_{min} computation can be found in

The minimal distance is computed with the following equation :

$$d_{\text{min}} = \left[v_{ego}\tau_{res} + \frac{1}{2}a_{\text{max, accel}}\,\tau_{res}^2 + \frac{(v_{ego} + \tau_{res}a_{\text{max, accel}})^2}{2a_{\text{min, brake}}} - \frac{v_{fwd}^2}{2a_{\text{max , brake}}} \right]_+, \tag{6.10}$$

where $[x]_+ := \max\{x, 0\}$ and v is the velocity of the cars. Let t_d be a period during which the situation is critical and $d \leq d_{min}$. The interval $[t_d, t_d + \tau_{res}]$ becomes a critical interval where the common sense safety rules are applied. The mathematical proof of d_{min} computation can be found in [Shalev-Shwartz, 2017].

2.4.5 Lateral Critical Situations

Let c_{ego} be a car driving at a velocity v_{ego} and c a sideways moving car with a velocity v_c during a time interval $[0, \tau_res]$ distant from each other with a distance d. A lateral critical situation occurs when both cars (or one of them) apply a lateral acceleration $a_{max,acc}^{lat}$ and then brake $a_{min,brake}^{lat}$ until colliding laterally. Otherwise, the lateral situation is safe. The interval $[t_{lat}^d, t]$ is a critical lateral interval time, where t_{lat}^d is a lateral danger threshold time. The minimal distance is computed by :

$$d_{\text{min}} = \mu + \left[\frac{v_{ego} + v_{ego,\tau}}{2}\tau + \frac{v_{ego,\tau}^2}{2a_{\text{min , brake}}^{\text{lat}}} - \left(\frac{v_c + v_{c,\tau}}{2}\tau - \frac{v_{c,\tau}^2}{2a_{\text{min , brake}}^{\text{lat}}} \right) \right]_+ . \tag{6.11}$$

The interval $[t_{lat}^d, t]$ is a critical lateral interval time, where t_{lat}^d is a lateral danger threshold time. The mathematical proof of d_{min} can be found in [Shalev-Shwartz, 2017].

3 Experiments

3.1 Environment Setup and Evaluation Metrics

All the experiments were conducted on the Carla simulator. The environment includes criteria such as high traffic density or complex intersections (i.e., intersections with multiple lanes, merging traffic, and pedestrians crossing). It also contains narrow streets, construction zones, inclement weather, and pedestrian/cyclist interactions, which provide a realistic urban driving experience.

Training Procedure. We evaluated the training phase and the testing outcomes in Town10. The goal of an agent is to complete a trip from point A to point B with no infractions. Points A and B are randomly chosen from a list of 120 points manually placed on the map, with 7140 possibilities. Using the search algorithm A^* [Hart, 1968], a planner calculates the route between

TABLE 6.1 – Training parameters

Parameter	Value
Number of input frames	4
AE model type	CNN
AE z_{dim}	400
Action history	false
Action smoothing	0
Batch size	32
Discount factor	0.99
Entropy scale	0.01
Value scale	1
Eval interval	20
Synchronous	true
FPS	30
GAE lambda	0.95
Horizon	128
Initial std	1
Learning rate	$1e^{-4}$
LR decay	0.9995
Num epochs	3
PPO epsilon	0.2
Seed	0

both points. Instead of scoring the completion rate of a route, an episode is considered successful when the agent travels a distance of 2500 m, as in [Zhao, 2022]. Counting the completion rate of a simple 200-meter straight route is not an indicator of generalization, whereas traveling 2,500 m with random routes does.

To save time, we defined three termination criteria : 1) the agent drives at a speed of $1km/h$ for 5 seconds, 2) the agent deviates from the center for more than 2.5m, and 3) an agent travels a distance of 2500 m. We used five metrics to compare our results : the total rewards, the total distance traveled, the center lane deviation, the angle deviation, and the average speed. [Codevilla, 2018] suggest using the NoCrash benchmark to evaluate the independent driving policy in various urban settings. This benchmark has three different traffic situations with varying degrees of difficulty : empty, regular (mid numbers of people and cars), and dense (no moving items) (a large number of pedestrians and vehicles). Besides, it specifies 25 routes in Town01 for training and 25 in Town02 for testing, along with six different types of weather. Our autonomous agents are tested in the testing town and the testing weather to see how well they operate. Table 6.1 represents the training hyperparameters.

Obstacles Avoidance Scenarios. NoCrash benchmark does not consider how the appearance of various cars (such as small cars and big trucks) might affect the agent's behavior. NoCrash benchmark tests the agent over a lengthy path. Each scenario is created using the unpredictable actions of cars and people, for example, pedestrians crossing the road. Furthermore, there are 26 types of pedestrians and 27 types of cars in CARLA 0.9.8. We thus created twenty-six different obstacle avoidance scenarios in Town01 and Town02. Each scenario is a set

of short courses to lessen the unpredictability and assess the obstacle avoidance performance and inertia problem [Codevilla, 2018]. It is worth noting that all the courses are just for testing. We determine the success rates along all paths, just like in the NoCrash benchmark.

Vehicles Avoidance Scenarios. Similarly, we created twenty-seven different obstacle avoidance scenarios. A parked vehicle is produced at a 30m distance to block the ego vehicle once it reaches the trigger location. The parked vehicle disappears, and the ego must stop in time and resume its motion.

Pedestrians Avoidance Scenarios. A person appears at a distance of 30 m on the sidewalk to cross the road. The ego vehicle must halt in time and resume motion after the pedestrian has crossed the street.

In the first stage, we gathered a vast and diverse dataset for training our system using Carla's autopilot with additional random noise and 25 training routes under the three criteria specified in the NoCrash benchmark from CARLA. A 60k 160x80x3 monocular RGB image, a waypoint sequence, GPS, and IMU measurement will be fused using a Kalman filter for accurate localization. We moved the camera around the autopilot during data collection. Besides, we introduced a random noise to the steering and throttle between -0.1 to 0.1, leading to a more realistic dataset for training the RL agent.

3.2 Results On NoCrach Benchmark

We compared our framework with state-of-the-art methods such as CADRE [Zhao, 2022], IARL [Toromanoff, 2020], and LBC [Chen, 2020] (described in 3.2). Results of IARL and LBC are taken from CADRE. Besides, IARL does not provide test results on testing weather, though we have only the results on the training weather. Table 6.2 shows the success rate results on the NoCrash benchmark.

In the empty scenario, GHRL achieves the highest score of 97 compared to the other methods. LBC and CADRE also perform reasonably well, with scores of 89 and 95, respectively. However, IARL lags behind with a score of 85, indicating slightly lower performance compared to GHRL and CADRE.

Moving to the regular scenario, GHRL demonstrates performance with a perfect score of 100, showcasing its robustness and reliability in regular driving conditions. IARL, LBC, and CADRE follow with scores of 86, 87, and 92, respectively.

In the dense scenario, GHRL maintains its superiority with a score of 96, indicating its effectiveness even in dense traffic situations. IARL lags behind significantly with a score of 63, suggesting challenges in handling dense traffic scenarios. LBC and CADRE perform better than IARL but still fall short of GHRL, with scores of 75 and 82, respectively.

Overall, GHRL consistently outperforms the other methods across all scenarios, showcasing its effectiveness in ensuring safe driving conditions. While IARL, LBC, and CADRE demonstrate reasonable performance, they show limitations compared to GHRL, especially in challenging scenarios such as dense traffic.

Table 6.2 – NoCrash Benchmark

Task	Town	Weather	IARL	LBC	CADRE	GHRL
Empty			85	89	95	**97**
Regular	Train	Train	86	87	92	**100**
Dense			63	75	82	**96**
Empty				60	94	**95**
Regular	Train	Test		60	86	**90**
Dense				54	76	**85**
Empty			77	85	92	92
Regular	Test	Train	66	79	78	**88**
Dense			33	53	61	**72**
Empty				36	78	78
Regular	Test	Test		36	72	**85**
Dense				12	52	**65**

Table 6.3 – Obstacle Avoidance Benchmark

	Vehicle avoidance	Pedestrian avoidance
LBC	55 / 81	73 / 78
IARL	69 / 81	57 / 78
CADRE	81 / 81	76 / 78
GHRL	81 / 81	77 / 78

3.3 Results on Obstacle Avoidance

Table 6.3 shows the results of the NoCrash benchmark obstacle avoidance scenarios. We have executed the evaluation 81 times for vehicle avoidance and 78 times for pedestrian avoidance to align with the test scenarios performed by CADRE, LBC, and IARL. Our framework achieved 81 over 81 in vehicle avoidance, the same as CADRE, and slightly outperformed it in pedestrian avoidance by achieving 77 over 78.

3.4 Training Sub-Policies

Figure 6.4 illustrates the performance evaluation of distinct iterations of GHRL agents within the traffic light management sub-policy context. These agent variants diverge based on the proportion of expert demonstrations integrated into ASP, which are employed to steer the training of the PPO policies. The findings underscore that GHRL agents infused with a higher ratio of ASP rules tend to exhibit superior performance compared to their counterparts lacking ASP rules (PPO). This discrepancy in performance is attributed to the fact that ASP rules encapsulate fundamental driving principles, such as refraining from acceleration when facing a red traffic light. These encoded rules contribute to developing a traffic light management policy that balances safety and efficiency within the GHRL agent. Furthermore, the results illuminate a delicate equilibrium between the extent of ASP rule integration and the agent's overarching learning and generalization capacity. Notably, the GHRL agent, incorporating 20% ASP rules, demonstrates optimal performance within the traffic light management sub-policy. However, this

agent variant falls short compared to other GHRL agents in different sub-policies. This phenomenon is likely rooted in the specificity of the 20% ASP rules, which may be overly tailored to the traffic light management sub-policy, thus hindering their applicability to broader scenarios. The outcomes underscore that the strategic incorporation of ASP rules during GHRL agent training can significantly enhance their competence in traffic light management. These findings emphasize the importance of striking a nuanced balance between integrating domain-specific rules and the agent's broader capacity for learning and adaptation.

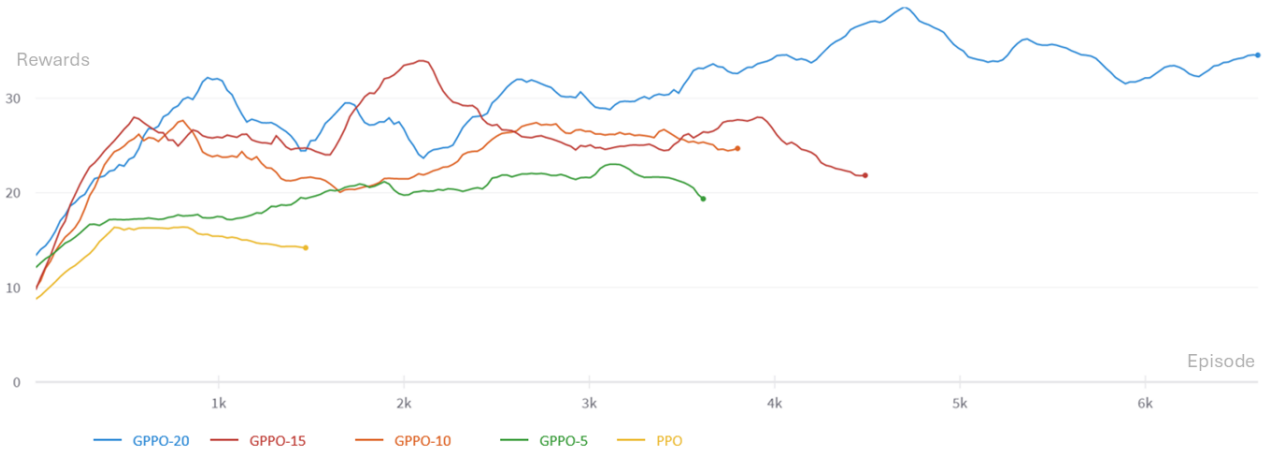


FIGURE 6.4 – Learning sub-policy

3.5 Safe High-Policies Experiments

Figure 6.5 is dedicated to meticulously evaluating urban driving safety through the lens of distinct reinforcement learning (RL) algorithms. Our focus extends to diverse scenarios encompassing potential hazards and the unpredictable pre-crash scenarios stipulated by the National Highway Traffic Safety Administration (NHTSA). The provided visual representation offers a comparative analysis of three RL algorithms within a simulated urban driving environment :

— GHRL with safety rules (GHRL-R) : This variant integrates safety rules into the GHRL agent's framework, a strategic augmentation to enhance its safety performance.

— GHRL : This configuration adheres to the conventional GHRL algorithm without any embedded safety rules.

— HRL : In this instance, a traditional hierarchical reinforcement learning algorithm is deployed without any safety rule incorporation.

The outcomes of our comparative analysis underscore the superior performance of GHRL-R in critical situations, eclipsing both GHRL and HRL counterparts. Notably, GHRL-R not only outperforms its counterparts in dangerous scenarios but also accumulates higher rewards across the entirety of the evaluation. This observed trend alludes to the effectiveness of safety features in GHRL-R, rendering it capable of making informed decisions in precarious scenarios while still achieving commendable performance across other contexts.

Conversely, HRL emerges as the weakest performer, trailing behind GHRL and GHRL-R in overall rewards and significantly struggling in critical situations. This deficiency in performance can be attributed to the need for built-in safety mechanisms within the HRL framework. The absence of such safety mechanisms renders HRL more susceptible to committing errors in potentially hazardous situations, thus affecting its overall efficacy.

The outcomes derived from this investigation distinctly advocate for introducing safety rules into RL algorithms to elevate their safety performance in urban driving scenarios. This strategic incorporation, as evidenced by the superiority of GHRL-R, can substantially enhance an agent's ability to navigate dangerous circumstances. However, it is imperative to exercise meticulousness in the design of these safety rules. Striking a balance between safety and learning potential is pivotal to ensure that the agent's ability to learn and generalize is not compromised in pursuing enhanced safety measures.

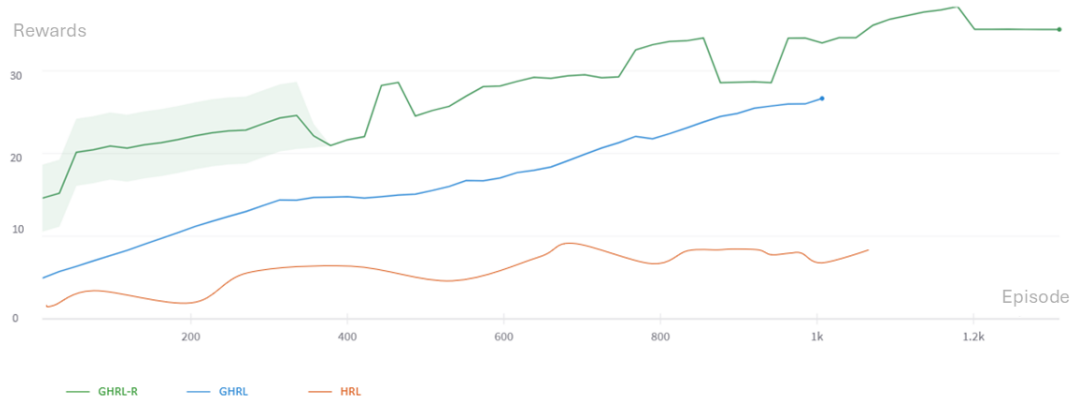


FIGURE 6.5 – Safety comparison

4 Conclusion

This chapter presents GHRL, an innovative framework for vision-based control of autonomous vehicles within complex urban environments. GHRL capitalizes on the capabilities of a convolutional neural network (CNN) to extract pertinent visual features from camera imagery, supplemented by localization and waypoints serving as navigation inputs. The reinforcement learning (RL) paradigm is harnessed to foster the acquisition of high-level policies through the option-critic (OC) framework while simultaneously enabling the acquisition of low-level policies via expert demonstrations encoded within Answer Set Programming (ASP) rules. Notably, the decision-making process is fortified by including safety rules strategically invoked during precarious scenarios to ensure the agent's decisions align with the principles of safety and responsibility.

The efficacy of GHRL is substantiated through its evaluation on the Carla NoCrash benchmark. Additionally, an ablation study is conducted to discern the impact of diverse network architectures and RL hyperparameters on the performance of the proposed framework. The

empirical outcomes from these evaluations reveal GHRL's superiority over state-of-the-art methodologies, manifesting in 87% overall success rate on the Carla NoCrash benchmark and a fourfold improvement compared to conventional RL methods. The results effectively underscore the potential and viability of harnessing hierarchical reinforcement learning (HRL) for the vision-based control of autonomous vehicles operating within urban landscapes.

In the following chapter, we will explore the thesis conclusion, summarizing the key contributions and insights of this research, and discuss potential future directions in the field of autonomous driving.

Conclusion and Future Directions

1 Summary

This thesis has made contributions to the field of autonomous driving, with a particular focus on vehicle localization, automated parking, and safety control.

Chapter 4 presents a solution for vehicle localization, a crucial aspect of autonomous navigation. Leveraging a Kalman filter-based approach, the proposed method adeptly fuses measurements from low-cost Global Positioning System (GPS) and Inertial Measurement Unit (IMU) sensors. By capitalizing on the complementary strengths of these sensors, the solution aims to achieve precise and reliable position estimation, thereby facilitating safe and efficient autonomous navigation. The efficacy of the proposed method is rigorously evaluated through a series of comprehensive experiments involving both simulated and real-world datasets. These evaluations include diverse scenarios, thereby encompassing a wide range of environmental conditions and driving maneuvers. The empirical analysis reveals that the approach consistently outperforms alternative localization methods, particularly in challenging urban environments characterized by signal obstructions and high noise levels. Notably, the empirical evidence shows a high accuracy of the solution. The method's performance superiority compared to conventional localization techniques signifies a promising advancement in autonomous vehicle localization. This promising outcome paves the way for its consideration as a viable and effective solution to address the critical challenges in autonomous vehicle navigation. The method's ability to maintain superior performance across various environmental settings and driving scenarios makes it particularly promising for industrial applications where precise and reliable vehicle localization is paramount.

Chapter 5 introduces the DPPO-IL framework, tailored explicitly for end-to-end automated parking. The framework leverages the Proximal Policy Optimization (PPO) algorithm to acquire optimal actions for parking maneuvers while effectively circumventing both static and dynamic obstacles. The research design incorporates a task-specific curriculum learning strategy alongside integrating dynamic reward function adjustments, augmenting the learning process. The empirical evaluation of DPPO-IL is performed within a simulated parking environment, whereby the agent is observed to accomplish a remarkable 90% success rate in a highly complex and challenging setting.

The DPPO-IL framework exhibits its versatility by successfully learning to park in various parking configurations, including parallel, perpendicular, and angled spots. Moreover, the agent

demonstrates its adaptability to environmental changes, notably manifesting the capability to navigate safely in the presence of other vehicles or pedestrians. The simulation-based assessments affirm the efficacy of DPPO-IL in accomplishing the parking task and provide a comprehensive understanding of its performance characteristics.

Overall, introducing the DPPO-IL framework represents a substantial advancement in automated parking systems. Its integration of advanced reinforcement learning techniques, task-specific curriculum, and dynamic reward functions contribute to its high success rate and adaptive capabilities in a complex parking environment. The empirical validation of the framework across various parking scenarios establishes its potential for real-world deployment, thus contributing valuable insights to autonomous parking and reinforcement learning in the context of vehicle navigation.

Chapter 6 introduces the Guided Hierarchy of Reinforcement Learning (GHRL) framework, a seminal contribution in the realm of vision-based control for autonomous vehicles within the challenging context of urban environments. The GHRL framework presents an innovative combination of advanced machine-learning techniques to enable efficient and safe navigation for autonomous vehicles.

At the core of the GHRL framework lies a Convolutional Neural Network (CNN) employed to extract highly discriminative visual features from raw camera images, combined with precise localization and waypoints, forming a comprehensive and informative navigation input. Subsequently, the framework harnesses a Hierarchical Reinforcement Learning (HRL) algorithm, a powerful paradigm for decision-making in dynamic environments, to learn both high-level policies, orchestrated through the Option-critic (OC) framework, and low-level policies that are effectively guided by expert demonstrations encoded as Answer Set Programming (ASP) rules. This hierarchical structure endows the GHRL framework with a multi-level control system that effectively balances the high-level strategic decisions and the fine-grained control actions, leading to proficient and contextually aware navigation.

Furthermore, the GHRL framework incorporates safety rules that augment the agent's learning process to ensure responsible and safe decision-making. These safety rules act as a safeguard, enabling the agent to make prudent and cautious decisions even in complex and ambiguous situations, thus mitigating potential risks and ensuring the safety of passengers, pedestrians, and other road users.

The GHRL framework's prowess is robustly demonstrated through rigorous empirical evaluations on the Carla NoCrash benchmark, where it exhibits superior performance compared to state-of-the-art methods. This notable achievement signifies GHRL's potential as an effective vision-based control solution for autonomous vehicles navigating complex and dynamic urban environments.

This thesis contributes to the burgeoning field of autonomous driving, specifically in vehicle localization, automated parking, and vision-based control. Empirical evaluations confirm their potential for real-world deployment, providing clear responses to the research questions.

2 Perspectives

This thesis introduces solutions in autonomous driving ; it is imperative to recognize that certain inherent limitations and challenges still need to be exhaustively addressed, thus warranting more in-depth scientific scrutiny. These limitations stem from various sources, including the complexities of real-world environments, algorithmic constraints, and hardware limitations.

Vehicle localization. The proposed solution showcases the absolute localization for autonomous vehicles by integrating two low-cost sensors, namely GPS and IMU. The approach includes a high-precision calibration method for the IMU sensor, noise reduction through a low-pass filter, and using ES-EKF to integrate IMU data with GPS coordinates. However, the real world presents diverse and dynamic conditions, from varying weather and lighting conditions to unexpected obstacles and road deviations. Ensuring the robustness and adaptability of the localization system under such conditions is a critical challenge. Addressing this challenge requires an in-depth exploration of sensor fusion techniques, machine learning algorithms, and data collection strategies to enhance the system's precision, reliability, and scalability.

Real-time modeling of vehicle dynamics is critical for autonomous vehicles operating at the limits of handling. Precise knowledge of the current vehicle dynamics is essential, with tire-road contact, characterized by the friction value, playing a pivotal role. This friction value undergoes significant changes influenced by aerodynamics (downforce), road conditions (tarmac), weather conditions (rain, snow), and the vehicle's ongoing maneuver (load shift during braking, acceleration). The inherent high level of model uncertainty, attributed to external factors coupled with the nonlinear effects within tire and vehicle dynamics, poses a formidable challenge for motion planning and control algorithms. While existing models provide certain approximations of vehicle dynamics, their computational demands remain significant, particularly concerning tire models. Ongoing research endeavors aim to leverage artificial neural networks to compute the vehicle's dynamic behavior, offering computational efficiency surpassing traditional physical models.

Cumulative rewards optimization poses an additional challenge in reinforcement learning, stemming from the inherent tension between the imperative to explore learning optimal policies and the paramount importance of ensuring safety. This challenge manifests itself in distinct ways across various approaches. In the case of hierarchical agents, suboptimal policies distort rewards intractably, impeding the improvement of policies. On the other hand, end-to-end methods need to be improved in exploring rewards due to the gradual expansion of the safe region during training, thereby presenting obstacles to effective learning.

Domain adaptation, transferring knowledge gained in a simulated environment to a real-world setting, poses a significant challenge in the autonomous vehicle (AV) domain. The data received by sensors in the real world often belongs to distributions vastly different from those encountered during training in simulated environments. Additionally, a crucial related concern is inference time in the real-world environment.

Prior research has explored the sim-to-real transfer approach to address this challenge, particularly in localized AV tasks such as optimal parking assignment or navigation in controlled

environments like indoor areas. Notably, meta-reinforcement learning (meta-RL) techniques have successfully achieved zero-shot sim-to-real domain adaptation in robotics.

Exploring meta-RL-based approaches becomes a promising avenue to further enhance sim-to-real transfer efficiency in the AV domain. These techniques, which have succeeded in robotics, could be adapted for efficient knowledge transfer in the AV context.

Furthermore, active learning emerges as another potential research area in this domain. Active learning involves agents consulting human experts in real-time to enhance the model's decision-making abilities. This avenue holds promise for improving AV systems' adaptability and real-world performance.

Generating new Answer Set Programming (ASP) rules directly from the environment without relying on manual, handcrafted engineering. This issue arises in autonomous systems or intelligent agents that utilize ASP for decision-making or rule-based reasoning.

Traditional approaches often involve human experts crafting ASP rules based on their understanding of the environment and task requirements. However, this manual engineering process can be labor-intensive and time-consuming, and it may need to fully capture the complexity of dynamic or evolving environments.

The research problem seeks innovative solutions that enable autonomous systems to autonomously learn and generate ASP rules by interacting with and observing their environment. The goal is to reduce the dependence on manual rule creation, allowing systems to adapt flexibly to changes and uncertainties in their operating environment.

In conclusion, the thesis catalyzes future research in the autonomous driving domain, providing solutions and performance benchmarks and inspiring comprehensive scientific exploration. The unfolding research landscape is poised to unlock new horizons, offering the potential to refine existing techniques, develop advanced safety and regulatory frameworks, and enhance user acceptance, ultimately propelling the field of autonomous driving into a new era of innovation and practical implementation.

List Of Publications

[P.1]

Mohamad Albilani, Amel Bouzeghoub. "*Guided Hierarchical Reinforcement Learning for Safe Urban Driving*". **2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)** .

[P.2]

Mohamad Albilani, Amel Bouzeghoub. "*Dynamic Adjustment of Reward Function for Proximal Policy Optimization with Imitation Learning : Application to Automated Parking Systems.*". **2022 IEEE Intelligent Vehicles Symposium (IV)** .

[P.3]

Mohamad Albilani, Amel Bouzeghoub. "*Localization of Autonomous Vehicle with low cost sensors.*". **2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)** .

Bibliographie

[Achiam, 2017] Joshua Achiam, David Held, Aviv Tamar et Pieter Abbeel. « Constrained policy optimization ». *International conference on machine learning*. PMLR. 2017, p. 22-31 (cf. p. 5, 13).

[Agarwal, 2019] Tanmay Agarwal, Hitesh Arora, Tanvir Parhar, Shubhankar Deshpande et Jeff Schneider. « Learning to drive using waypoints ». *Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information ProcessingSystems (NeurIPS 2019)*. 2019 (cf. p. 71, 78).

[Ahmed, 2019] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi et Dale Schuurmans. « Understanding the impact of entropy on policy optimization ». *International conference on machine learning*. PMLR. 2019, p. 151-160 (cf. p. 41).

[Akermi, 2020] Kada Akermi, Samira Chouraqui et Boudjemaa Boudaa. « Novel SMC control design for path following of autonomous vehicles with uncertainties and mismatched disturbances ». *International Journal of Dynamics and Control* 8.1 (2020), p. 254-268 (cf. p. 54).

[Albilani, 2022] Mohamad Albilani et Amel Bouzeghoub. « Localization of Autonomous Vehicle with low cost sensors ». *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE. 2022, p. 339-345 (cf. p. 122).

[Alcantarilla, 2018] Pablo F Alcantarilla, Simon Stent, German Ros, Roberto Arroyo et Riccardo Gherardi. « Street-view change detection with deconvolutional networks ». *Autonomous Robots* 42 (2018), p. 1301-1322 (cf. p. 54).

[Alshiekh, 2018] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum et Ufuk Topcu. « Safe reinforcement learning via shielding ». *Proceedings of the AAAI conference on artificial intelligence*. T. 32. 1. 2018 (cf. p. 5, 13, 74).

[Alsuwian, 2022] Turki Alsuwian, Rana Basharat Saeed et Arslan Ahmed Amin. « Autonomous Vehicle with Emergency Braking Algorithm Based on Multi-Sensor Fusion and Super Twisting Speed Controller ». *Applied Sciences* 12.17 (2022), p. 8458 (cf. p. 3).

[Amini, 2022] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han et al. « Vista 2.0 : An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles ». *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, p. 2419-2426 (cf. p. 77).

[Andrychowicz, 2017] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder et al. « Hindsight experience replay ». *Advances in neural information processing systems* 30 (2017) (cf. p. 30).

[Atia, 2017] Mohamed Maher Atia, Allaa R Hilal, Clive Stellings, Eric Hartwell, Jason Toonstra, William B Miners et al. « A low-cost lane-determination system using GNSS/IMU fusion and HMM-based multistage map matching ». *IEEE Transactions on Intelligent Transportation Systems* 18.11 (2017), p. 3027-3037 (cf. p. 46).

[Bach, 2017] Martin Bach, Stephan Reuter et Klaus Dietmayer. « Multi-camera traffic light recognition using a classifying Labeled Multi-Bernoulli filter ». *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, p. 1045-1051 (cf. p. 54).

[Bai, 2019] J. Bai, F. Lu, K. Zhang et Others Onnx. *Open neural network exchange*. 2019 (cf. p. 111).

[Banos, 2012] O. Banos, M. Damas, H. Pomares et I. Rojas. « On the use of sensor fusion to reduce the impact of rotational and additive noise in human activity recognition ». *Sensors* 12 (2012), p. 8039-8054 (cf. p. 55).

[Bansal, 2018] Mayank Bansal, Alex Krizhevsky et Abhijit Ogale. « Chauffeurnet : Learning to drive by imitating the best and synthesizing the worst ». *arXiv preprint arXiv :1812.03079* (2018) (cf. p. 13, 65, 119).

[Barth-Maron, 2018] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva Tb et al. « Distributed distributional deterministic policy gradients ». *arXiv preprint arXiv :1804.08617* (2018) (cf. p. 36).

[Beattie, 2016] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler et al. « Deepmind lab ». *arXiv preprint arXiv :1612.03801* (2016) (cf. p. 75).

[Behrendt, 2017] Karsten Behrendt, Libor Novak et Rami Botros. « A deep learning approach to traffic lights : Detection, tracking, and classification ». *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, p. 1370-1377 (cf. p. 54).

[Bejar, 2019] E. Bejar et A. Morán. « Reverse parking a car-like mobile robot with deep reinforcement learning and preview control ». *2019 IEEE 9th Annual Computing And Communication Workshop And Conference (CCWC)* (2019), p. 377-0383 (cf. p. 72).

[Bellemare, 2017] Marc G Bellemare, Will Dabney et Rémi Munos. « A distributional perspective on reinforcement learning ». *International conference on machine learning*. PMLR. 2017, p. 449-458 (cf. p. 36).

[Bellemare, 2013] Marc G Bellemare, Yavar Naddaf, Joel Veness et Michael Bowling. « The arcade learning environment : An evaluation platform for general agents ». *Journal of Artificial Intelligence Research* 47 (2013), p. 253-279 (cf. p. 75).

[Belousov, 2021] Boris Belousov, Hany Abdulsamad, Pascal Klink, Simone Parisi et Jan Peters. *Reinforcement learning algorithms : analysis and applications*. Springer, 2021 (cf. p. 31).

[Bewley, 2019] Alex Bewley, Jessica Rigley, Yuxuan Liu, Jeffrey Hawke, Richard Shen, Vinh-Dieu Lam et al. « Learning to drive from simulation without real world labels ». *2019 International conference on robotics and automation (ICRA)*. IEEE. 2019, p. 4818-4824 (cf. p. 61, 62).

[Blondé, 2019] L. Blondé et A. Kalousis. « Sample-efficient imitation learning via generative adversarial nets ». *The 22nd International Conference On Artificial Intelligence And Statistics* (2019), p. 3138-3148 (cf. p. 108).

[Bojarski, 2016] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal et al. « End to end learning for self-driving cars ». *arXiv preprint arXiv :1604.07316* (2016) (cf. p. 54, 58, 65).

[Bojarski, 2017] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel et al. « Explaining how a deep neural network trained with end-to-end learning steers a car ». *arXiv preprint arXiv :1704.07911* (2017) (cf. p. 54).

[Bounini, 2016] F. Bounini, D. Gingras, H. Pollart et D. Gruyer. « Real time cooperative localization for autonomous vehicles ». *2016 IEEE 19th International Conference On Intelligent Transportation Systems (ITSC)* (2016), p. 1186-1191 (cf. p. 11, 81).

[Bozic, 2018] Svetozar Mile Bozic. *Digital and Kalman filtering*. Courier Dover Publications, 2018 (cf. p. 84).

[Brewka, 2011] Gerhard Brewka, Thomas Eiter et Mirosław Truszczyński. « Answer set programming at a glance ». *Communications of the ACM* 54.12 (2011), p. 92-103 (cf. p. 14, 120).

[Brodeur, 2017] Simon Brodeur, Ethan Perez, Ankesh Anand, Florian Golemo, Luca Celotti, Florian Strub et al. « Home : A household multimodal environment ». *arXiv preprint arXiv :1711.11017* (2017) (cf. p. 75).

[Bronstein, 2022] Eli Bronstein, Mark Palatucci, Dominik Notz, Brandyn White, Alex Kuefler, Yiren Lu et al. « Hierarchical Model-Based Imitation Learning for Planning in Autonomous Driving ». *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, p. 8652-8659 (cf. p. 13, 119).

[Brostow, 2009] Gabriel J Brostow, Julien Fauqueur et Roberto Cipolla. « Semantic object classes in video : A high-definition ground truth database ». *Pattern Recognition Letters* 30.2 (2009), p. 88-97 (cf. p. 76).

[Buehler, 2007] Martin Buehler, Karl Iagnemma et Sanjiv Singh. *The 2005 DARPA grand challenge : the great robot race*. T. 36. springer, 2007 (cf. p. 53).

[Buehler, 2009] Martin Buehler, Karl Iagnemma et Sanjiv Singh. *The DARPA urban challenge : autonomous vehicles in city traffic*. T. 56. springer, 2009 (cf. p. 53).

[Burton, 2017] Simon Burton, Lydia Gauerhof et Christian Heinzemann. « Making the case for safety of machine learning in highly automated driving ». *Computer Safety, Reliability, and Security : SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS, Trento, Italy, September 12, 2017, Proceedings 36*. Springer. 2017, p. 5-16 (cf. p. 5).

[Cabon, 2020] Yohann Cabon, Naila Murray et Martin Humenberger. « Virtual kitti 2 ». *arXiv preprint arXiv :2001.10773* (2020) (cf. p. 76).

[Castillo-Torres, 2021] R. Castillo-Torres. « Embedded Implementation of a Kalman Filter for the Fusion of Automotive Inertial Sensors Using CARLA Simulator. (ITESO ». 2021 (2021) (cf. p. 11, 81).

[Cattaneo, 2020] D. Cattaneo, M. Vaghi, S. Fontana, A. Ballardini et D. Sorrenti. « Global visual localization in LiDAR-maps through shared 2D-3D embedding space ». *2020 IEEE International Conference On Robotics And Automation (ICRA)* (2020), p. 4365-4371 (cf. p. 56).

[Chen, 2015] Chenyi Chen, Ari Seff, Alain Kornhauser et Jianxiong Xiao. « Deepdriving : Learning affordance for direct perception in autonomous driving ». *Proceedings of the IEEE international conference on computer vision*. 2015, p. 2722-2730 (cf. p. 62, 78).

[Chen, 2020] Dian Chen, Brady Zhou, Vladlen Koltun et Philipp Krähenbühl. « Learning by cheating ». *Conference on Robot Learning*. PMLR. 2020, p. 66-75 (cf. p. 132).

[Chen, 2018] Jianyu Chen, Zining Wang et Masayoshi Tomizuka. « Deep hierarchical reinforcement learning for autonomous driving with distinct behaviors ». *2018 IEEE intelligent vehicles symposium (IV)*. IEEE. 2018, p. 1239-1244 (cf. p. 67, 68).

[Chen, 2019a] Jianyu Chen, Bodi Yuan et Masayoshi Tomizuka. « Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety ». *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, p. 2884-2890 (cf. p. 13, 67, 78, 119).

[Chen, 2019b] Jianyu Chen, Bodi Yuan et Masayoshi Tomizuka. « Model-free deep reinforcement learning for urban autonomous driving ». *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE. 2019, p. 2765-2771 (cf. p. 71).

[Chen, 2017a] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy et Alan L Yuille. « Deeplab : Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs ». *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), p. 834-848 (cf. p. 63).

[Chen, 2017b] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Huimin Ma, Sanja Fidler et Raquel Urtasun. « 3d object proposals using stereo imagery for accurate object class detection ». *IEEE transactions on pattern analysis and machine intelligence* 40.5 (2017), p. 1259-1272 (cf. p. 54).

[Chopra, 2019] Kriti Chopra, Kunal Gupta et Annu Lambora. « Future internet : The internet of things-a literature review ». *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*. IEEE. 2019, p. 135-139 (cf. p. 54).

[Cina, 2023] Mehdi Cina et Ahmad B Rad. « Categorized review of drive simulators and driver behavior analysis focusing on ACT-R architecture in autonomous vehicles ». *Sustainable Energy Technologies and Assessments* 56 (2023), p. 103044 (cf. p. 3).

[Clements, 2017] Lewis M Clements et Kara M Kockelman. « Economic effects of automated vehicles ». *Transportation Research Record* 2606.1 (2017), p. 106-114 (cf. p. 53).

[Codevilla, 2018] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun et Alexey Dosovitskiy. « End-to-end driving via conditional imitation learning ». *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, p. 4693-4700 (cf. p. 59, 60, 65, 131, 132).

[Codevilla, 2019] Felipe Codevilla, Eder Santana, Antonio M López et Adrien Gaidon. « Exploring the limitations of behavior cloning for autonomous driving ». *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, p. 9329-9338 (cf. p. 10, 64, 103, 106).

[Cohen, 2021] Andrew Cohen, Ervin Teng, Vincent-Pierre Berges, Ruo-Ping Dong, Hunter Henry, Marwan Mattar et al. « On the use and misuse of absorbing states in multi-agent reinforcement learning ». *arXiv preprint arXiv :2111.05992* (2021) (cf. p. 113).

[Collin, 2020] Anne Collin, Artur Bilka, Scott Pendleton et Radboud Duintjer Tebbens. « Safety of the intended driving behavior using rulebooks ». *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2020, p. 136-143 (cf. p. 73, 74).

[Cordts, 2016] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson et al. « The cityscapes dataset for semantic urban scene understanding ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 3213-3223 (cf. p. 63, 76).

[Cropper, 2022] Andrew Cropper, Sebastijan Dumančić, Richard Evans et Stephen H Muggleton. « Inductive logic programming at 30 ». *Machine Learning* (2022), p. 1-26 (cf. p. 13).

[Cultrera, 2020] Luca Cultrera, Lorenzo Seidenari, Federico Becattini, Pietro Pala et Alberto Del Bimbo. « Explaining autonomous driving by learning end-to-end visual attention ». *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, p. 340-341 (cf. p. 59).

[DAlfonso, 2015] L. D'Alfonso, W. Lucia, P. Muraca et P. Pugliese. « Mobile robot localization via EKF and UKF : A comparison based on real data ». *Robotics And Autonomous Systems* 74 (2015), p. 122-127 (cf. p. 11, 81).

[De Giacomo, 2019] Giuseppe De Giacomo, Luca Iocchi, Marco Favorito et Fabio Patrizi. « Foundations for restraining bolts : Reinforcement learning with LTLf/LDLf restraining specifications ». *Proceedings of the international conference on automated planning and scheduling*. T. 29. 2019, p. 128-136 (cf. p. 5).

[Delhi, 2016] Springer India-New Delhi. « Automotive revolution & perspective towards 2030 ». *Auto Tech Review* 4.5 (2016), p. 20-25 (cf. p. 53).

[Den Hengst, 2022] Floris Den Hengst, Vincent François-Lavet, Mark Hoogendoorn et Frank van Harmelen. « Planning for potential : efficient safe reinforcement learning ». *Machine Learning* 111.6 (2022), p. 2255-2274 (cf. p. 5).

[Di, 2021] Xuan Di et Rongye Shi. « A survey on autonomous vehicle control in the era of mixed-autonomy : From physics-based to AI-guided driving policy learning ». *Transportation research part C : emerging technologies* 125 (2021), p. 103008 (cf. p. 54).

[Dimopoulos, 1997] Yannis Dimopoulos, Bernhard Nebel et Jana Koehler. « Encoding planning problems in nonmonotonic logic programs ». *European Conference on Planning*. Springer. 1997, p. 169-181 (cf. p. 47).

[Doersch, 2016] Carl Doersch. « Tutorial on variational autoencoders ». *arXiv preprint arXiv :1606.05908* (2016) (cf. p. 122).

[Dosovitskiy, 2017] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez et Vladlen Koltun. « CARLA : An open urban driving simulator ». *Conference on robot learning*. PMLR. 2017, p. 1-16 (cf. p. 59, 61, 68, 69, 71, 77, 79, 88, 120).

[Du, 2020] Z. Du, Q. Miao et C. Zong. « Trajectory Planning for Automated Parking Systems Using Deep Reinforcement Learning ». *International Journal Of Automotive Technology* 21 (2020), p. 881-887 (cf. p. 72).

[Duan, 2016] Y. Duan, X. Chen, R. Houthooft, J. Schulman et P. Abbeel. « Benchmarking deep reinforcement learning for continuous control ». *International Conference On Machine Learning* (2016), p. 1329-1338 (cf. p. 35).

[Espeholt, 2018] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward et al. « Impala : Scalable distributed deep-rl with importance weighted actor-learner architectures ». *International conference on machine learning*. PMLR. 2018, p. 1407-1416 (cf. p. 37).

[Everingham, 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn et Andrew Zisserman. « The pascal visual object classes (voc) challenge ». *International journal of computer vision* 88 (2010), p. 303-338 (cf. p. 54).

[Fachantidis, 2013] A. Fachantidis, I. Partalas, G. Tsoumakas et I. Vlahavas. « Transferring task models in reinforcement learning agents ». *Neurocomputing* 107 (2013), p. 23-32 (cf. p. 109).

[Fang, 2016] Jie Fang, Yu Zhou, Yao Yu et Sidan Du. « Fine-grained vehicle model recognition using a coarse-to-fine convolutional neural network architecture ». *IEEE Transactions on Intelligent Transportation Systems* 18.7 (2016), p. 1782-1792 (cf. p. 54).

[Farag, 2019] Wael Farag et Zakaria Saleh. « An advanced vehicle detection and tracking scheme for self-driving cars ». *2nd Smart Cities Symposium (SCS 2019)*. IET. 2019, p. 1-6 (cf. p. 103).

[Farrell, 2017] Jay A Farrell et Paul F Roysdon. « Advanced vehicle state estimation : A tutorial and comparative study ». *IFAC-PapersOnLine* 50.1 (2017), p. 15971-15976 (cf. p. 3).

[Favarò, 2017] Francesca M Favarò, Nazanin Nader, Sky O Eurich, Michelle Tripp et Naresh Varadaraju. « Examining accident reports involving autonomous vehicles in California ». *PLoS one* 12.9 (2017), e0184952 (cf. p. 54).

[Fehér, 2020] Árpád Fehér, Szilárd Aradi, Tamás Bécsi, Péter Gáspár et Zsolt Szalay. « Proving ground test of a ddpg-based vehicle trajectory planner ». *2020 European Control Conference (ECC)*. IEEE. 2020, p. 332-337 (cf. p. 72, 103).

[Fehér, 2019] Árpád Fehér, Szilárd Aradi, Ferenc Hegedüs, Tamás Bécsi et Péter Gáspár. « Hybrid DDPG approach for vehicle motion planning » (2019) (cf. p. 72, 103).

[Feraco, 2021] S. Feraco, S. Favelli, A. Tonoli, A. Bonfitto et N. Amati. « Localization Method for Autonomous Vehicles with Sensor Fusion Using Extended and Unscented Kalman Filters. (SAE Technical Paper ». 2021 (2021) (cf. p. 56).

[Fernandez, 2010] F. Fernandez, J. Garcia et M. Veloso. « Probabilistic policy reuse for inter-task transfer learning ». *Robotics And Autonomous Systems* 58 (2010), p. 866-871 (cf. p. 109).

[Fernando, 2017] Tharindu Fernando, Simon Denman, Sridha Sridharan et Clinton Fookes. « Going deeper : Autonomous steering with neural memory networks ». *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017, p. 214-221 (cf. p. 58).

[Ferraris, 2005] Paolo Ferraris. « Answer sets for propositional theories ». *Logic Programming and Nonmonotonic Reasoning : 8th International Conference, LPNMR 2005, Diamante, Italy, September 5-8, 2005. Proceedings 8*. Springer. 2005, p. 119-131 (cf. p. 48).

[Filos, 2020] Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine et Yarin Gal. « Can autonomous vehicles identify, recover from, and adapt to distribution shifts ? » *International Conference on Machine Learning*. PMLR. 2020, p. 3145-3153 (cf. p. 74).

[Folkers, 2019] A. Folkers, M. Rick et C. B"uskens. « Controlling an autonomous vehicle with deep reinforcement learning ». *2019 IEEE Intelligent Vehicles Symposium (IV)* (2019), p. 2025-2031 (cf. p. 72, 103).

[Forechi, 2018] A. Forechi, T. Oliveira-Santos, C. Badue et A. De Souza. « Visual global localization with a hybrid WNN-CNN approach ». *2018 International Joint Conference On Neural Networks (IJCNN)* (2018), p. 1-9 (cf. p. 56).

[Fraichard, 2004] T. Fraichard et A. Scheuer. « From Reeds and Shepp's to continuous-curvature paths ». *IEEE Transactions On Robotics* 20 (2004), p. 1025-1035 (cf. p. 12, 71, 103).

[Fritsch, 2013] Jannik Fritsch, Tobias Kuehnl et Andreas Geiger. « A new performance measure and evaluation benchmark for road detection algorithms ». *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE. 2013, p. 1693-1700 (cf. p. 76).

[Furda, 2011] Andrei Furda et Ljubo Vlacic. « Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making ». *IEEE Intelligent Transportation Systems Magazine* 3.1 (2011), p. 4-17 (cf. p. 13, 73, 119).

[Gandhi, 2017] Dhiraj Gandhi, Lerrel Pinto et Abhinav Gupta. « Learning to fly by crashing ». *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, p. 3948-3955 (cf. p. 65).

[Garcıa, 2015] Javier Garcia et Fernando Fernández. « A comprehensive survey on safe reinforcement learning ». *Journal of Machine Learning Research* 16.1 (2015), p. 1437-1480 (cf. p. 5).

[Gebser, 2007] Martin Gebser, Torsten Schaub et Sven Thiele. « Gringo : A new grounder for answer set programming ». *Logic Programming and Nonmonotonic Reasoning : 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007. Proceedings 9*. Springer. 2007, p. 266-271 (cf. p. 47).

[Geiger, 2013] Andreas Geiger, Philip Lenz, Christoph Stiller et Raquel Urtasun. « Vision meets robotics : The kitti dataset ». *The International Journal of Robotics Research* 32.11 (2013), p. 1231-1237 (cf. p. 62, 63, 76).

[Gelb, 1974] Arthur Gelb et al. *Applied optimal estimation*. MIT press, 1974 (cf. p. 4, 55).

[Gelfond, 1988] Michael Gelfond et Vladimir Lifschitz. « The stable model semantics for logic programming. » *ICLP/SLP*. T. 88. Cambridge, MA. 1988, p. 1070-1080 (cf. p. 48).

[Geyer, 2020] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung et al. « A2d2 : Audi autonomous driving dataset ». *arXiv preprint arXiv :2004.06320* (2020) (cf. p. 77).

[Giacalone, 2019] Jean-Pierre Giacalone, Luc Bourgeois et Andrea Ancora. « Challenges in aggregation of heterogeneous sensors for Autonomous Driving Systems ». *2019 IEEE sensors applications symposium (SAS)*. IEEE. 2019, p. 1-5 (cf. p. 3).

[Gomes, 2008] Carla P Gomes, Henry Kautz, Ashish Sabharwal et Bart Selman. « Satisfiability solvers ». *Foundations of Artificial Intelligence* 3 (2008), p. 89-134 (cf. p. 47).

[Gómez-Bravo, 2001] F. Gómez-Bravo, F. Cuesta et A. Ollero. « Parallel and diagonal parking in nonholonomic autonomous vehicles ». *Engineering Applications Of Artificial Intelligence* 14 (2001), p. 419-434 (cf. p. 12, 71, 103).

[Gordon, 2012] J Gordon. « JavaScript Racer ». *JavaScript racing game we modified* (2012) (cf. p. 77).

[Gu, 2022] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang et al. « A review of safe reinforcement learning : Methods, theory and applications ». *arXiv preprint arXiv :2205.10330* (2022) (cf. p. 5, 13).

[Gu, 2023] Ziqing Gu, Lingping Gao, Haitong Ma, Shengbo Eben Li, Sifa Zheng, Wei Jing et al. « Safe-state enhancement method for autonomous driving via direct hierarchical reinforcement learning ». *IEEE Transactions on Intelligent Transportation Systems* (2023) (cf. p. 67).

[Guo, 2021] Youtian Guo, Qichao Zhang, Junjie Wang et Shasha Liu. « Hierarchical reinforcement learning-based policy switching towards multi-scenarios autonomous driving ». *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, p. 1-8 (cf. p. 13, 119, 127).

[Haarnoja, 2017] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel et Sergey Levine. « Reinforcement learning with deep energy-based policies ». *International conference on machine learning*. PMLR. 2017, p. 1352-1361 (cf. p. 40).

[Haarnoja, 2018] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan et al. « Soft actor-critic algorithms and applications ». *arXiv preprint arXiv :1812.05905* (2018) (cf. p. 41, 106, 113).

[Han, 2011] L. Han, Q. Do et S. Mita. « Unified path planner for parking an autonomous vehicle based on RRT ». *2011 IEEE International Conference On Robotics And Automation* (2011), p. 5622-5627 (cf. p. 12, 71).

[Hart, 1968] Peter E Hart, Nils J Nilsson et Bertram Raphael. « A formal basis for the heuristic determination of minimum cost paths ». *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), p. 100-107 (cf. p. 130).

[Hasanbeig, 2019] Mohammadhosein Hasanbeig, Yiannis Kantaros, Alessandro Abate, Daniel Kroening, George J Pappas et Insup Lee. « Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees ». *2019 IEEE 58th conference on decision and control (CDC)*. IEEE. 2019, p. 5338-5343 (cf. p. 5).

[Hasselt, 2010] Hado Hasselt. « Double Q-learning ». *Advances in neural information processing systems* 23 (2010) (cf. p. 29, 31).

[Hasselt, 2016] Van Hasselt. « Deep reinforcement learning with double Q-learning ». *Assoc. Adv. Artif. Intell.* 2 (2016), p. 5 (cf. p. 29, 30).

[Hataba, 2022] Muhammad Hataba, Ahmed Sherif, Mohamed Mahmoud, Mohamed Abdallah et Waleed Alasmary. « Security and Privacy Issues in Autonomous Vehicles : A Layer-Based Survey ». *IEEE Open Journal of the Communications Society* 3 (2022), p. 811-829 (cf. p. 3).

[Hawke, 2020] Jeffrey Hawke, Richard Shen, Corina Gurau, Siddharth Sharma, Daniele Reda, Nikolay Nikolov et al. « Urban driving with conditional imitation learning ». *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, p. 251-257 (cf. p. 63, 78).

[He, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren et Jian Sun. « Deep residual learning for image recognition ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770-778 (cf. p. 21).

[He, 2019] X. He, Y. Liu, C. Lv, X. Ji et Y. Liu. « Emergency steering control of autonomous vehicle for collision avoidance and stabilisation ». *Vehicle System Dynamics* 57 (2019), p. 1163-1187 (cf. p. 12, 72).

[Heess, 2015] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap et David Silver. « Memory-based control with recurrent neural networks ». *arXiv preprint arXiv :1512.04455* (2015) (cf. p. 35).

[Henderson, 2018] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup et David Meger. « Deep reinforcement learning that matters ». *Proceedings of the AAAI conference on artificial intelligence*. T. 32. 1. 2018 (cf. p. 35).

[Hessel, 2018] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney et al. « Rainbow : Combining improvements in deep reinforcement learning ». *Proceedings of the AAAI conference on artificial intelligence*. T. 32. 1. 2018 (cf. p. 31).

[Ho, 2016] J. Ho et S. Ermon. « Generative adversarial imitation learning ». *Advances In Neural Information Processing Systems* 29 (2016), p. 4565-4573 (cf. p. 109).

[Hoshiya, 1984] M. Hoshiya et E. Saito. « Structural identification by extended Kalman filter ». *Journal Of Engineering Mechanics* 110 (1984), p. 1757-1770 (cf. p. 4, 56).

[Hu, 2020] Jin-wen Hu, Bo-yin Zheng, Ce Wang, Chun-hui Zhao, Xiao-lei Hou, Quan Pan et al. « A survey on multi-sensor fusion based obstacle detection for intelligent ground vehicles in off-road environments ». *Frontiers of Information Technology & Electronic Engineering* 21.5 (2020), p. 675-692 (cf. p. 44).

[Huang, 2019] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng et Ruigang Yang. « The apolloscape open dataset for autonomous driving and its application ». *IEEE transactions on pattern analysis and machine intelligence* 42.10 (2019), p. 2702-2719 (cf. p. 77).

[Hutsebaut-Buysse, 2022] Matthias Hutsebaut-Buysse, Kevin Mets et Steven Latré. « Hierarchical reinforcement learning : A survey and open research challenges ». *Machine Learning and Knowledge Extraction* 4.1 (2022), p. 172-221 (cf. p. 121).

[International, 2018] Sae International. « Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles ». *SAE international* 4970.724 (2018), p. 1-5 (cf. p. 54).

[Jaderberg, 2016] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver et al. « Reinforcement learning with unsupervised auxiliary tasks ». *arXiv preprint arXiv :1611.05397* (2016) (cf. p. 78).

[Jamgochian, 2022] Arec Jamgochian, Etienne Buehrle, Johannes Fischer et Mykel J Kochenderfer. « SHAIL : Safety-Aware Hierarchical Adversarial Imitation Learning for Autonomous Driving in Urban Environments ». *arXiv preprint arXiv :2204.01922* (2022) (cf. p. 5, 74).

[Jansen, 2020] Nils Jansen, Bettina Könighofer, JSL Junges, AC Serban et Roderick Bloem. « Safe reinforcement learning using probabilistic shields » (2020) (cf. p. 74).

[Jaritz, 2018] Maximilian Jaritz, Raoul De Charette, Marin Toromanoff, Etienne Perot et Fawzi Nashashibi. « End-to-end race driving with deep reinforcement learning ». *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, p. 2070-2075 (cf. p. 66).

[Jeong, 2010] SH Jeong, CG Choi, JN Oh, PJ Yoon, BS Kim, M Kim et al. « Low cost design of parallel parking assist system based on an ultrasonic sensor ». *International Journal of Automotive Technology* 11.3 (2010), p. 409-416 (cf. p. 45).

[Jiang, 2021] Yuqian Jiang, Suda Bharadwaj, Bo Wu, Rishi Shah, Ufuk Topcu et Peter Stone. « Temporal-logic-based reward shaping for continuing reinforcement learning tasks ». *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 35. 9. 2021, p. 7995-8003 (cf. p. 5).

[Jo, 2014] K. Jo, J. Kim, D. Kim, C. Jang et M. Sunwoo. « Development of autonomous car—Part I : Distributed system architecture and development process ». *IEEE Transactions On Industrial Electronics* 61 (2014), p. 7131-7140 (cf. p. 81).

[Kalman, 1960] R. A Kalman. new approach to linear filtering et prediction problems, 1960 (cf. p. 4, 55).

[Kang, 2019] Yue Kang, Hang Yin et Christian Berger. « Test your self-driving algorithm : An overview of publicly available driving datasets and virtual testing environments ». *IEEE Transactions on Intelligent Vehicles* 4.2 (2019), p. 171-185 (cf. p. 63).

[Kanhere, 2018] A. Kanhere et G. Gao. « Integrity for GPS/LiDAR fusion utilizing a RAIM framework ». *Proceedings Of The 31st International Technical Meeting Of The Satellite Division Of The Institute Of Navigation (ION GNSS+ 2018)* (2018), p. 3145-3155 (cf. p. 81).

[Kendall, 2019] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen et al. « Learning to drive in a day ». *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, p. 8248-8254 (cf. p. 5, 13, 70, 78, 119).

[Khaleghi, 2013] B. Khaleghi, A. Khamis, F. Karray et S. Multisensor data fusion : A Razavi. « review of the state-of-the-art ». *Information Fusion* 14 (2013), p. 28-44 (cf. p. 11, 81).

[Khankalantary, 2020] S. Khankalantary, S. Rafatnia et H. Mohammadkhani. « An adaptive constrained type-2 fuzzy Hammerstein neural network data fusion scheme for low-cost SINS/GNSS navigation system ». *Applied Soft Computing* 86 (2020) (cf. p. 56).

[Kimura, 2021] Daiki Kimura, Subhajit Chaudhury, Akifumi Wachi, Ryosuke Kohita, Asim Munawar, Michiaki Tatsubori et al. « Reinforcement learning with external knowledge by using logical neural networks ». *arXiv preprint arXiv :2103.02363* (2021) (cf. p. 74).

[Kingma, 2014] Diederik P Kingma et Jimmy Ba. « Adam : A method for stochastic optimization ». *arXiv preprint arXiv :1412.6980* (2014) (cf. p. 21, 29, 109).

[Kiran, 2021] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani et al. « Deep reinforcement learning for autonomous driving : A survey ». *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2021), p. 4909-4926 (cf. p. 5).

[Kolve, 2017] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti et al. « Ai2-thor : An interactive 3d environment for visual ai ». *arXiv preprint arXiv :1712.05474* (2017) (cf. p. 76).

[Könighofer, 2022] Bettina Könighofer, Julian Rudolf, Alexander Palmisano, Martin Tappler et Roderick Bloem. « Online shielding for reinforcement learning ». *Innovations in Systems and Software Engineering* (2022), p. 1-16 (cf. p. 5).

[Kothawade, 2021] Suraj Kothawade, Vinaya Khandelwal, Kinjal Basu, Huaduo Wang et Gopal Gupta. « AUTO-DISCERN : autonomous driving using common sense reasoning ». *arXiv preprint arXiv :2110.13606* (2021) (cf. p. 73, 74).

[Kovacova, 2022] Maria Kovacova, Judit Oláh, József Popp et Elvira Nica. « The Algorithmic Governance of Autonomous Driving Behaviors : Multi-Sensor Data Fusion, Spatial Computing Technologies, and Movement Tracking Tools ». *Contemporary Readings in Law and Social Justice* 14.2 (2022), p. 27-45 (cf. p. 4).

[Krasowski, 2022] Hanna Krasowski, Yinqiang Zhang et Matthias Althoff. « Safe Reinforcement Learning for Urban Driving using Invariably Safe Braking Sets ». *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2022, p. 2407-2414 (cf. p. 74).

[Lazaric, 2008] Alessandro Lazaric, Marcello Restelli et Andrea Bonarini. « Transfer of samples in batch reinforcement learning ». *Proceedings of the 25th international conference on Machine learning*. 2008, p. 544-551 (cf. p. 109).

[LeCun, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio et Patrick Haffner. « Gradient-based learning applied to document recognition ». *Proceedings of the IEEE* 86.11 (1998), p. 2278-2324 (cf. p. 20, 21).

[Lee, 2018] Hee Seok Lee et Kang Kim. « Simultaneous traffic sign detection and boundary estimation using convolutional neural network ». *IEEE Transactions on Intelligent Transportation Systems* 19.5 (2018), p. 1652-1663 (cf. p. 54).

[Leurent, 2018] Edouard Leurent et al. « An environment for autonomous driving decision-making » (2018) (cf. p. 77).

[Leurent, 2020] Edouard Leurent. « Safe and efficient reinforcement learning for behavioural planning in autonomous driving ». Thèse de doct. Université de Lille, 2020 (cf. p. 74).

[Li, 2017] Bo Li. « 3d fully convolutional network for vehicle detection in point cloud ». *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, p. 1513-1518 (cf. p. 54).

[Li, 2022a] Guofa Li, Shenglong Li, Shen Li et Xingda Qu. « Continuous decision-making for autonomous driving at intersections using deep deterministic policy gradient ». *IET Intelligent Transport Systems* 16.12 (2022), p. 1669-1681 (cf. p. 4).

[Li, 2022b] Guofa Li, Yifan Yang, Shen Li, Xingda Qu, Nengchao Lyu et Shengbo Eben Li. « Decision making of autonomous vehicles in lane change scenarios : Deep reinforcement learning approaches with risk awareness ». *Transportation research part C : emerging technologies* 134 (2022), p. 103452 (cf. p. 5).

[Li, 2019] Guofa Li, Yifan Yang et Xingda Qu. « Deep learning approaches on pedestrian detection in hazy weather ». *IEEE Transactions on Industrial Electronics* 67.10 (2019), p. 8889-8899 (cf. p. 66, 78).

[Li, 2021a] Jinning Li, Liting Sun, Jianyu Chen, Masayoshi Tomizuka et Wei Zhan. « A safe hierarchical planning framework for complex driving scenarios based on reinforcement learning ». *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, p. 2660-2666 (cf. p. 74).

[Li, 2018] Zhihao Li, Toshiyuki Motoyoshi, Kazuma Sasaki, Tetsuya Ogata et Shigeki Sugano. « Rethinking self-driving : Multi-task knowledge for better generalization and accident explanation ability ». *arXiv preprint arXiv :1809.11100* (2018) (cf. p. 63, 78).

[Li, 2021b] Zhong Li, Minxue Pan, Tian Zhang et Xuandong Li. « Testing dnn-based autonomous driving systems under critical environmental conditions ». *International Conference on Machine Learning*. PMLR. 2021, p. 6471-6482 (cf. p. 74).

[Liang, 2018] Xiaodan Liang, Tairui Wang, Luona Yang et Eric Xing. « Cirl : Controllable imitative reinforcement learning for vision-based self-driving ». *Proceedings of the European conference on computer vision (ECCV)*. 2018, p. 584-599 (cf. p. 68, 78).

[Lillicrap, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa et al. « Continuous control with deep reinforcement learning ». *arXiv preprint arXiv :1509.02971* (2015) (cf. p. 35, 36, 70).

[Lin, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan et al. « Microsoft coco : Common objects in context ». *Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, p. 740-755 (cf. p. 54).

[Lin, 2022] Yuanfei Lin et Matthias Althoff. « Rule-compliant trajectory repairing using satisfiability modulo theories ». *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2022, p. 449-456 (cf. p. 73, 74).

[Lini, 2011] G. Lini, A. Piazzi et L. Consolini. « Multi-optimization of n 3-splines for autonomous parking ». *2011 50th IEEE Conference On Decision And Control And European Control Conference* (2011), p. 6367-6372 (cf. p. 12, 71, 103).

[Linja, 2022] Anne Linja, Tauseef Ibne Mamun et Shane T Mueller. « When Self-Driving Fails : Evaluating Social Media Posts Regarding Problems and Misconceptions about Tesla's FSD Mode ». *Multimodal Technologies and Interaction* 6.10 (2022), p. 86 (cf. p. 54).

[Liu, 2021] Ze Liu, Yingfeng Cai, Hai Wang, Long Chen, Hongbo Gao, Yunyi Jia et al. « Robust target recognition and tracking of self-driving cars with radar and camera information fusion under severe weather conditions ». *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2021), p. 6640-6653 (cf. p. 4, 81).

[Luo, 2017] Hengliang Luo, Yi Yang, Bei Tong, Fuchao Wu et Bin Fan. « Traffic sign recognition using a multi-task convolutional neural network ». *IEEE Transactions on Intelligent Transportation Systems* 19.4 (2017), p. 1100-1111 (cf. p. 54).

[Luo, 2011] R. Luo, C. Chang et C. Lai. « Multisensor fusion and integration : Theories, applications, and its perspectives ». *IEEE Sensors Journal* 11 (2011), p. 3122-3138 (cf. p. 55).

[Maddern, 2017] Will Maddern, Geoffrey Pascoe, Chris Linegar et Paul Newman. « 1 year, 1000 km : The Oxford RobotCar dataset ». *The International Journal of Robotics Research* 36.1 (2017), p. 3-15 (cf. p. 76).

[Madyastha, 2011] Venkatesh Madyastha, Vishal Ravindra, Srinath Mallikarjunan et Anup Goyal. « Extended Kalman filter vs. error state Kalman filter for aircraft attitude estimation ». *AIAA Guidance, Navigation, and Control Conference*. 2011, p. 6615 (cf. p. 56, 84).

[Mahmassani, 2016] Hani S Mahmassani. « 50th anniversary invited article—Autonomous vehicles and connected vehicle systems : Flow and operations considerations ». *Transportation Science* 50.4 (2016), p. 1140-1162 (cf. p. 53).

[Marek, 1999] Victor W Marek et Miroslaw Truszczyński. « Stable models and an alternative logic programming paradigm ». *The Logic Programming Paradigm : a 25-Year Perspective*. Springer, 1999, p. 375-398 (cf. p. 47).

[Marković, 2022] Lovro Marković, Marin Kovač, Robert Milijas, Marko Car et Stjepan Bogdan. « Error state extended Kalman filter multi-sensor fusion for unmanned aerial vehicle localization in GPS and magnetometer denied indoor environments ». *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2022, p. 184-190 (cf. p. 81).

[McCormac, 2016] John McCormac, Ankur Handa, Stefan Leutenegger et Andrew J Davison. « Scenenet rgb-d : 5m photorealistic images of synthetic indoor trajectories with ground truth ». *arXiv preprint arXiv :1612.05079* (2016) (cf. p. 75).

[Mehta, 2018] Ashish Mehta, Adithya Subramanian et Anbumani Subramanian. « Learning end-to-end autonomous driving using guided auxiliary supervision ». *Proceedings of the 11th Indian Conference on Computer Vision, Graphics and Image Processing*. 2018, p. 1-8 (cf. p. 63, 78).

[Memarian, 2021] Farzan Memarian, Wonjoon Goo, Rudolf Lioutikov, Scott Niekum et Ufuk Topcu. « Self-supervised online reward shaping in sparse-reward environments ». *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, p. 2369-2375 (cf. p. 5).

[Menze, 2015] Moritz Menze et Andreas Geiger. « Object scene flow for autonomous vehicles ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, p. 3061-3070 (cf. p. 76).

[Michels, 2005] Jeff Michels, Ashutosh Saxena et Andrew Y Ng. « High speed obstacle avoidance using monocular vision and reinforcement learning ». *Proceedings of the 22nd international conference on Machine learning*. 2005, p. 593-600 (cf. p. 69, 70).

[Mirchevska, 2018] Branka Mirchevska, Christian Pek, Moritz Werling, Matthias Althoff et Joschka Boedecker. « High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning ». *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, p. 2156-2162 (cf. p. 68).

[Mirowski, 2016] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino et al. « Learning to navigate in complex environments ». *arXiv preprint arXiv :1611.03673* (2016) (cf. p. 78).

[Mnih, 2015] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare et al. « control through deep reinforcement learning ». *Nature* 518 (2015), p. 529-533 (cf. p. 29, 30, 35, 103).

[Mnih, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley et al. « Asynchronous methods for deep reinforcement learning ». *International conference on machine learning*. PMLR. 2016, p. 1928-1937 (cf. p. 38).

[Mnih, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra et al. « Playing atari with deep reinforcement learning ». *arXiv preprint arXiv :1312.5602* (2013) (cf. p. 28-30).

[Montemerlo, 2008] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger et al. « Junior : The stanford entry in the urban challenge ». *Journal of field Robotics* 25.9 (2008), p. 569-597 (cf. p. 73).

[Mukhtar, 2015] Amir Mukhtar, Likun Xia et Tong Boon Tang. « Vehicle detection techniques for collision avoidance systems : A review ». *IEEE transactions on intelligent transportation systems* 16.5 (2015), p. 2318-2338 (cf. p. 53).

[Muller, 2005] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp et Yann Cun. « Off-road obstacle avoidance through end-to-end learning ». *Advances in neural information processing systems* 18 (2005) (cf. p. 58).

[Müller, 2018] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem et Vladlen Koltun. « Driving policy transfer via modularity and abstraction ». *arXiv preprint arXiv :1804.09364* (2018) (cf. p. 13, 60, 61).

[Nachum, 2019] Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee et Sergey Levine. « Why does hierarchy (sometimes) work so well in reinforcement learning ? » *arXiv preprint arXiv :1909.10618* (2019) (cf. p. 41).

[Narvekar, 2020] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor et Peter Stone. « Curriculum learning for reinforcement learning domains : A framework and survey ». *The Journal of Machine Learning Research* 21.1 (2020), p. 7382-7431 (cf. p. 116).

[Neumann, 2016] Peter G Neumann. « Risks of automation : A cautionary total-system perspective of our cyberfuture ». *Communications of the ACM* 59.10 (2016), p. 26-30 (cf. p. 54).

[Niemelä, 1999] Ilkka Niemelä. « Logic programs with stable model semantics as a constraint programming paradigm ». *Annals of mathematics and Artificial Intelligence* 25 (1999), p. 241-273 (cf. p. 47, 48).

[Nosrati, 2018] Masoud S Nosrati, Elmira Amirloo Abolfathi, Mohammed Elmahgiubi, Peyman Yadmellat, Jun Luo, Yunfei Zhang et al. « Towards practical hierarchical reinforcement learning for multi-lane autonomous driving » (2018) (cf. p. 68).

[Notomista, 2017] Gennaro Notomista et Michael Botsch. « A machine learning approach for the segmentation of driving maneuvers and its application in autonomous parking ». *Journal of Artificial Intelligence and Soft Computing Research* 7.4 (2017), p. 243-255 (cf. p. 54).

[Nweke, 2019] H. Nweke, Y. Teh, G. Mujtaba et M. Al-Garadi. « Data fusion and multiple classifier systems for human activity detection and health monitoring : Review and open research directions ». *Information Fusion* 46 (2019), p. 147-170 (cf. p. 4, 11, 81).

[Ohn-Bar, 2020] Eshed Ohn-Bar, Aditya Prakash, Aseem Behl, Kashyap Chitta et Andreas Geiger. « Learning situational driving ». *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, p. 11296-11305 (cf. p. 59).

[Osband, 2016] Ian Osband, Charles Blundell, Alexander Pritzel et Benjamin Van Roy. « Deep exploration via bootstrapped DQN ». *Advances in neural information processing systems* 29 (2016) (cf. p. 30).

[Osiński, 2020] Błażej Osiński, Piotr Miłoś, Adam Jakubowski, Paweł Zięcina, Michał Martyniak, Christopher Galias et al. « CARLA Real Traffic Scenarios–novel training ground and benchmark for autonomous driving ». *arXiv preprint arXiv :2012.11329* (2020) (cf. p. 79).

[Osório, 2019] A. Osório et A. Information Pinto. « uncertainty and the manipulability of artificial intelligence autonomous vehicles systems ». *International Journal Of Human-Computer Studies* 130 (2019), p. 40-46 (cf. p. 11, 81).

[Pan, 2017a] Xinlei Pan, Yurong You, Ziyan Wang et Cewu Lu. « Virtual to real reinforcement learning for autonomous driving ». *arXiv preprint arXiv :1704.03952* (2017) (cf. p. 70).

[Pan, 2017b] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou et al. « Agile autonomous driving using end-to-end deep imitation learning ». *arXiv preprint arXiv :1709.07174* (2017) (cf. p. 78).

[Park, 2018] H. Park, K. Ahn, M. Park et S. Lee. « Study on robust lateral controller for differential GPS-based autonomous vehicles ». *International Journal Of Precision Engineering And Manufacturing* 19 (2018), p. 367-376 (cf. p. 12).

[Parkinson, 2017] Simon Parkinson, Paul Ward, Kyle Wilson et Jonathan Miller. « Cyber threats facing autonomous and connected vehicles : Future challenges ». *IEEE transactions on intelligent transportation systems* 18.11 (2017), p. 2898-2915 (cf. p. 54).

[Pateria, 2021] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan et Chai Quek. « Hierarchical reinforcement learning : A comprehensive survey ». *ACM Computing Surveys (CSUR)* 54.5 (2021), p. 1-35 (cf. p. 41, 42).

[Pathak, 2017] Deepak Pathak, Pulkit Agrawal, Alexei A Efros et Trevor Darrell. « Curiosity-driven exploration by self-supervised prediction ». *International conference on machine learning*. PMLR. 2017, p. 2778-2787 (cf. p. 41, 106).

[Perot, 2017] Etienne Perot, Maximilian Jaritz, Marin Toromanoff et Raoul De Charette. « End-to-end driving in a realistic racing game with deep reinforcement learning ». *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, p. 3-4 (cf. p. 66).

[Pham, 2018] Tu-Hoa Pham, Giovanni De Magistris et Ryuki Tachibana. « Optlayer-practical constrained optimization for deep reinforcement learning in the real world ». *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, p. 6236-6243 (cf. p. 5).

[Phan-Minh, 2022] Tung Phan-Minh, Forbes Howington, Ting-Sheng Chu, Sang Uk Lee, Momchil S Tomov, Nanxiang Li et al. « Driving in real life with inverse reinforcement learning ». *arXiv preprint arXiv :2206.03004* (2022) (cf. p. 74).

[Pires, 2016] I. Pires, N. Garcia, N. Pombo et F. Flórez-Revuelta. « From data acquisition to data fusion : a comprehensive review and a roadmap for the identification of activities of daily living using mobile devices ». *Sensors* 16 (2016), p. 184 (cf. p. 55).

[Qian, 1999] Ning Qian. « On the momentum term in gradient descent learning algorithms ». *Neural networks* 12.1 (1999), p. 145-151 (cf. p. 21).

[Al-Qizwini, 2017] Mohammed Al-Qizwini, Iman Barjasteh, Hothaifa Al-Qassab et Hayder Radha. « Deep learning algorithm for autonomous driving using googlenet ». *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, p. 89-96 (cf. p. 62, 78).

[Radlak, 2020] Krystian Radlak, Michał Szczepankiewicz, Tim Jones et Piotr Serwa. « Organization of machine learning based product development as per ISO 26262 and ISO/PAS 21448 ». *2020 IEEE 25th Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE. 2020, p. 110-119 (cf. p. 73, 74).

[Rausch, 2017] Viktor Rausch, Andreas Hansen, Eugen Solowjow, Chang Liu, Edwin Kreuzer et J Karl Hedrick. « Learning a deep neural net policy for end-to-end control of autonomous vehicles ». *2017 American Control Conference (ACC)*. IEEE. 2017, p. 4914-4919 (cf. p. 58).

[Reid, 2019] Tyler GR Reid, Sarah E Houts, Robert Cammarata, Graham Mills, Siddharth Agarwal, Ankit Vora et al. « Localization requirements for autonomous vehicles ». *arXiv preprint arXiv :1906.01061* (2019) (cf. p. 81).

[Reiter, 1980] Raymond Reiter. « A logic for default reasoning ». *Artificial intelligence* 13.1-2 (1980), p. 81-132 (cf. p. 50).

[Rhinehart, 2018] Nicholas Rhinehart, Rowan McAllister et Sergey Levine. « Deep imitative models for flexible inference, planning, and control ». *arXiv preprint arXiv :1810.06544* (2018) (cf. p. 69).

[Riedmiller, 2007] Martin Riedmiller, Mike Montemerlo et Hendrik Dahlkamp. « Learning to drive a real car in 20 minutes ». *2007 Frontiers in the Convergence of Bioscience and Information Technologies*. IEEE. 2007, p. 645-650 (cf. p. 69, 70).

[Riegel, 2020] Ryan Riegel, Alexander Gray, Francois Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya et al. « Logical neural networks ». *arXiv preprint arXiv :2006.13155* (2020) (cf. p. 74).

[Ros, 2016] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez et Antonio M Lopez. « The synthia dataset : A large collection of synthetic images for semantic segmentation of urban scenes ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 3234-3243 (cf. p. 76).

[Rosenblatt, 1962] Frank Rosenblatt et al. *Principles of neurodynamics : Perceptrons and the theory of brain mechanisms*. T. 55. Spartan books Washington, DC, 1962 (cf. p. 19).

[Sallab, 2017] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot et Senthil Yogamani. « Deep reinforcement learning framework for autonomous driving ». *arXiv preprint arXiv :1704.02532* (2017) (cf. p. 66).

[Sallab, 2016] Ahmad El Sallab, Mohammed Abdou, Etienne Perot et Senthil Yogamani. « End-to-end deep reinforcement learning for lane keeping assist ». *arXiv preprint arXiv :1612.04340* (2016) (cf. p. 66).

[Sanders, 2016] A. Sanders. « An introduction to Unreal engine 4. (CRC Press ». 2016 (2016) (cf. p. 88).

[Sauer, 2018] Axel Sauer, Nikolay Savinov et Andreas Geiger. « Conditional affordance learning for driving in urban environments ». *Conference on robot learning*. PMLR. 2018, p. 237-252 (cf. p. 63, 78).

[Savva, 2017] Manolis Savva, Angel X Chang, Alexey Dosovitskiy, Thomas Funkhouser et Vladlen Koltun. « MINOS : Multimodal indoor simulator for navigation in complex environments ». *arXiv preprint arXiv :1712.03931* (2017) (cf. p. 75).

[Schaul, 2015] Tom Schaul, John Quan, Ioannis Antonoglou et David Silver. « Prioritized experience replay ». *arXiv preprint arXiv :1511.05952* (2015) (cf. p. 30, 31).

[Schneider, 2011] Frank E Schneider et Dennis Wildermuth. « Results of the european land robot trial and their usability for benchmarking outdoor robot systems ». *Towards Autonomous Robotic Systems : 12th Annual Conference, TAROS 2011, Sheffield, UK, August 31–September 2, 2011. Proceedings 12*. Springer. 2011, p. 408-409 (cf. p. 53).

[Schrittwieser, 2020] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt et al. « go, chess and shogi by planning with a learned model ». *Nature* 588 (2020), p. 604-609 (cf. p. 103).

[Schulman, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan et Philipp Moritz. « Trust region policy optimization ». *International conference on machine learning*. PMLR. 2015, p. 1889-1897 (cf. p. 38).

[Schulman, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford et Oleg Klimov. « Proximal policy optimization algorithms ». *arXiv preprint arXiv :1707.06347* (2017) (cf. p. 12, 31, 39, 73, 103, 106, 126).

[Seuwou, 2020] Patrice Seuwou, Ebad Banissi et George Ubakanma. « The future of mobility with connected and autonomous vehicles in smart cities ». *Digital twin technologies and smart cities* (2020), p. 37-52 (cf. p. 2).

[Shah, 2018] Shital Shah, Debadeepta Dey, Chris Lovett et Ashish Kapoor. « Airsim : High-fidelity visual and physical simulation for autonomous vehicles ». *Field and Service Robotics : Results of the 11th International Conference*. Springer. 2018, p. 621-635 (cf. p. 77).

[Shahian Jahromi, 2019] Babak Shahian Jahromi, Theja Tulabandhula et Sabri Cetin. « Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles ». *Sensors* 19.20 (2019), p. 4357 (cf. p. 46).

[Shalev-Shwartz, 2017] Shai Shalev-Shwartz, Shaked Shammah et Amnon Shashua. « On a formal model of safe and scalable self-driving cars ». *arXiv preprint arXiv :1708.06374* (2017) (cf. p. 127, 130).

[Shao, 2023] Hao Shao, Letian Wang, Ruobing Chen, Hongsheng Li et Yu Liu. « Safety-enhanced autonomous driving using interpretable sensor fusion transformer ». *Conference on Robot Learning*. PMLR. 2023, p. 726-737 (cf. p. 4).

[Al-Sharman, 2023] Mohammad Al-Sharman, Rowan Dempster, Mohamed A Daoud, Mahmoud Nasr, Derek Rayside et William Melek. « Self-learned autonomous driving at unsignalized intersections : A hierarchical reinforced learning approach for feasible decision-making ». *IEEE Transactions on Intelligent Transportation Systems* (2023) (cf. p. 67).

[Shaukat, 2021] Nabil Shaukat, Ahmed Ali, Muhammad Javed Iqbal, Muhammad Moinuddin et Pablo Otero. « Multi-sensor fusion for underwater vehicle localization by augmentation of rbf neural network and error-state kalman filter ». *Sensors* 21.4 (2021), p. 1149 (cf. p. 11, 81).

[Simonyan, 2014] Karen Simonyan et Andrew Zisserman. « Very deep convolutional networks for large-scale image recognition ». *arXiv preprint arXiv :1409.1556* (2014) (cf. p. 21).

[Singh, 2020] P. Singh et A. Manure. « Introduction to tensorflow 2.0 ». *Learn TensorFlow 2.0* (2020), p. 1-24 (cf. p. 111).

[Sola, 2017] Joan Sola. « Quaternion kinematics for the error-state Kalman filter ». *arXiv preprint arXiv :1711.02508* (2017) (cf. p. 85).

[Song, 2017] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva et Thomas Funkhouser. « Semantic scene completion from a single depth image ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 1746-1754 (cf. p. 75).

[Soni, 2006] V. Soni et S. Singh. « Using homomorphisms to transfer options across continuous reinforcement learning domains ». *AAAI* 6 (2006), p. 494-499 (cf. p. 109).

[Speranza, 2018] M Grazia Speranza. « Trends in transportation and logistics ». *European Journal of Operational Research* 264.3 (2018), p. 830-836 (cf. p. 53).

[Standardization, 2017a] I Standardization. « Intelligent transport systems—Assisted Parking System (APS)—Performance requirements and test procedures ». *Advances In Neural Information Processing Systems* (2017) (cf. p. ii).

[Standardization, 2017b] I. Standardization. « Intelligent transport systems — Assisted Parking System (APS) — Performance requirements and test procedures ». *Advances In Neural Information Processing Systems* 2017 (2017) (cf. p. 103).

[Subrahmanian, 1995] VS Subrahmanian et Carlo Zaniolo. « Relating Stable Models and AI Planning Domains. » *ICLP*. T. 95. 1995, p. 233-247 (cf. p. 47).

[Sun, 2020] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui et al. « Scalability in perception for autonomous driving : Waymo open dataset ». *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, p. 2446-2454 (cf. p. 76).

[Sun, 2023] Qian Sun, Le Zhang, Huan Yu, Weijia Zhang, Yu Mei et Hui Xiong. « Hierarchical reinforcement learning for dynamic autonomous vehicle navigation at intelligent intersections ». *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, p. 4852-4861 (cf. p. 67).

[Sutton, 2018] Richard S Sutton et Andrew G Barto. *Reinforcement learning : An introduction*. MIT press, 2018 (cf. p. 5, 13, 23, 27, 28, 103).

[Sutton, 1999] Richard S Sutton, Doina Precup et Satinder Singh. « Between MDPs and semi-MDPs : A framework for temporal abstraction in reinforcement learning ». *Artificial intelligence* 112.1-2 (1999), p. 181-211 (cf. p. 41, 42).

[Syrjänen, 2001] Tommi Syrjänen et Ilkka Niemelä. « The smodels system ». *International Conference on Logic Programming and NonMonotonic Reasoning*. Springer. 2001, p. 434-438 (cf. p. 47).

[Szegedy, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov et al. « Going deeper with convolutions ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, p. 1-9 (cf. p. 21).

[Taylor, 2005] M. Taylor et P. Stone. « Behavior transfer for value-function-based reinforcement learning ». *Proceedings Of The Fourth International Joint Conference On Autonomous Agents And Multiagent Systems* (2005), p. 53-59 (cf. p. 109).

[Thrun, 2006] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel et al. « Stanley : The robot that won the DARPA Grand Challenge ». *Journal of field Robotics* 23.9 (2006), p. 661-692 (cf. p. 12, 71).

[Thunyapoo, 2020] B. Thunyapoo, C. Ratchadakorntham, P. Siricharoen et W. Susutti. « Self-Parking Car Simulation using Reinforcement Learning Approach for Moderate Complexity Parking Scenario ». *2020 17th International Conference On Electrical Engineering/Electronics, Computer, Telecommunications And Information Technology (ECTI-CON)* (2020), p. 576-579 (cf. p. 72).

[Tiihonen, 2003] Juha Tiihonen, Timo Soininen, Ilkka Niemelä et Reijo Sulonen. « A practical tool for mass-customising configurable products ». *DS 31 : Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*. 2003 (cf. p. 47).

[Todorov, 2012] Emanuel Todorov, Tom Erez et Yuval Tassa. « Mujoco : A physics engine for model-based control ». *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, p. 5026-5033 (cf. p. 30, 40).

[Torabi, 2018] Faraz Torabi, Garrett Warnell et Peter Stone. « Behavioral cloning from observation ». *arXiv preprint arXiv :1805.01954* (2018) (cf. p. 109).

[Toromanoff, 2020] Marin Toromanoff, Emilie Wirbel et Fabien Moutarde. « End-to-end model-free reinforcement learning for urban driving using implicit affordances ». *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, p. 7153-7162 (cf. p. 65, 69, 78, 123, 132).

[Vergara, 2019] Marcus Loo Vergara. « Accelerating training of deep reinforcement learning-based autonomous driving agents through comparative study of agent and environment designs ». Mém. de mast. NTNU, 2019 (cf. p. 124).

[Vezhnevets, 2017] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver et al. « Feudal networks for hierarchical reinforcement learning ». *International conference on machine learning*. PMLR. 2017, p. 3540-3549 (cf. p. 41).

[Vitale, 2022] Christian Vitale, Panayiotis Kolios et Georgios Ellinas. « Autonomous intersection crossing with vehicle location uncertainty ». *IEEE Transactions on Intelligent Transportation Systems* 23.10 (2022), p. 17546-17561 (cf. p. 55).

[Wan, 2001] E. Wan et Van Der Merwe. « R. & Haykin, S. The unscented Kalman filter ». *Kalman Filtering And Neural Networks* 5 (2001), p. 221-280 (cf. p. 4, 56).

[Wang, 2024] Dawei Wang, Weizi Li, Lei Zhu et Jia Pan. « Learning to change : Choreographing mixed traffic through lateral control and hierarchical reinforcement learning ». *arXiv preprint arXiv :2403.14879* (2024) (cf. p. 67).

[Wang, 2020a] Jun Wang, Li Zhang, Yanjun Huang, Jian Zhao et Francesco Bella. « Safety of autonomous vehicles ». *Journal of advanced transportation* 2020 (2020), p. 1-13 (cf. p. 2, 3).

[Wang, 2016a] Shenlong Wang, Min Bai, Gellert Mattyus, Hang Chu, Wenjie Luo, Bin Yang et al. « Torontocity : Seeing the world with a million eyes ». *arXiv preprint arXiv :1612.00423* (2016) (cf. p. 76).

[Wang, 2020b] Tinghan Wang, Yugong Luo, Jinxin Liu, Rui Chen et Keqiang Li. « End-to-end self-driving approach independent of irrelevant roadside objects with auto-encoder ». *IEEE transactions on intelligent transportation systems* 23.1 (2020), p. 641-650 (cf. p. 122).

[Wang, 2016b] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu et al. « Sample efficient actor-critic with experience replay ». *arXiv preprint arXiv :1611.01224* (2016) (cf. p. 37).

[Wang, 2016c] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot et Nando Freitas. « Dueling network architectures for deep reinforcement learning ». *International conference on machine learning*. PMLR. 2016, p. 1995-2003 (cf. p. 30).

[Waymo, 2017] LLC Waymo. *Waymo safety report : On the road to fully self-driving*. 2017 (cf. p. 54).

[Wei, 2018] Ermo Wei, Drew Wicke, David Freelan et Sean Luke. « Multiagent soft q-learning ». *arXiv preprint arXiv :1804.09817* (2018) (cf. p. 40).

[Weinstein, 2010] Alejandro J Weinstein et Kevin L Moore. « Pose estimation of Ackerman steering vehicles for outdoors autonomous navigation ». *2010 IEEE International Conference on Industrial Technology*. IEEE. 2010, p. 579-584 (cf. p. 12, 103).

[Wolf, 2017] Peter Wolf, Christian Hubschneider, Michael Weber, André Bauer, Jonathan Härtl, Fabian Dürr et al. « Learning how to drive in a real world simulation with deep q-networks ». *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, p. 244-250 (cf. p. 13, 66, 119).

[Wu, 2018] Yi Wu, Yuxin Wu, Georgia Gkioxari et Yuandong Tian. « Building generalizable agents with a realistic and rich 3d environment ». *arXiv preprint arXiv :1801.02209* (2018) (cf. p. 75).

[Wu, 2017] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao et Jimmy Ba. « Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation ». *Advances in neural information processing systems* 30 (2017) (cf. p. 39).

[Wydmuch, 2018] Marek Wydmuch, Michał Kempka et Wojciech Jaśkowski. « Vizdoom competitions : Playing doom from pixels ». *IEEE Transactions on Games* 11.3 (2018), p. 248-259 (cf. p. 75).

[Wymann, 2000] Bernhard Wymann, Eric Espié, Christophe Guionneau, Christos Dimitrakakis, Rémi Coulom et Andrew Sumner. « Torcs, the open racing car simulator ». *Software available at http ://torcs. sourceforge. net* 4.6 (2000), p. 2 (cf. p. 62, 77).

[Xiao, 2021] Wei Xiao, Noushin Mehdipour, Anne Collin, Amitai Y Bin-Nun, Emilio Frazzoli, Radboud Duintjer Tebbens et al. « Rule-based optimal control for autonomous driving ». *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*. 2021, p. 143-154 (cf. p. 73, 74).

[Xin, 2014] J Xin, C Wang, Z Zhang et N Zheng. « China future challenge : Beyond the intelligent vehicle ». *IEEE Intell. Transp. Syst. Soc. Newslett* 16.2 (2014), p. 8-10 (cf. p. 53).

[Xiong, 2022] Huaqing Xiong, Tengyu Xu, Lin Zhao, Yingbin Liang et Wei Zhang. « Deterministic policy gradient : Convergence analysis ». *Uncertainty in Artificial Intelligence*. PMLR. 2022, p. 2159-2169 (cf. p. 35).

[Xu, 2022] Chejian Xu, Wenhao Ding, Weijie Lyu, Zuxin Liu, Shuai Wang, Yihan He et al. « Safebench : A benchmarking platform for safety evaluation of autonomous vehicles ». *Advances in Neural Information Processing Systems* 35 (2022), p. 25667-25682 (cf. p. 3).

[Xu, 2023] Cheng Xu, Zuoming Zhang, Fengjie Fu, Wenbin Yao, Hongyang Su, Youwei Hu et al. « Analysis of Spatiotemporal Factors Affecting Traffic Safety Based on Multisource Data Fusion ». *Journal of Transportation Engineering, Part A : Systems* 149.10 (2023), p. 04023098 (cf. p. 4).

[Xu, 2017] Huazhe Xu, Yang Gao, Fisher Yu et Trevor Darrell. « End-to-end learning of driving models from large-scale video datasets ». *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 2174-2182 (cf. p. 13, 58, 119).

[Xu, 2018] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang et Guy Broeck. « A semantic loss function for deep learning with symbolic knowledge ». *International conference on machine learning*. PMLR. 2018, p. 5502-5511 (cf. p. 5, 13).

[Yang, 2023] Wen-Chi Yang, Giuseppe Marra, Gavin Rens et Luc De Raedt. « Safe Reinforcement Learning via Probabilistic Logic Shields ». *arXiv preprint arXiv :2303.03226* (2023) (cf. p. 5, 13, 74).

[Yaqoob, 2019] I. Yaqoob, L. Khan, S. Kazmi, M. Imran, N. Guizani et C. Hong. « Autonomous driving cars in smart cities : Recent advances, requirements, and challenges ». *IEEE Network* 34 (2019), p. 174-181 (cf. p. 81).

[Yeong, 2021] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry et Joseph Walsh. « Sensor and sensor fusion technology in autonomous vehicles : A review ». *Sensors* 21.6 (2021), p. 2140 (cf. p. 4).

[Yu, 2020] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu et al. « Bdd100k : A diverse driving dataset for heterogeneous multitask learning ». *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, p. 2636-2645 (cf. p. 77).

[Zhang, 2020] J. Zhang, H. Chen, S. Song et F. Hu. « Reinforcement Learning-Based Motion Planning for Automatic Parking System ». *IEEE Access* 8 (2020), p. 54485-15450 (cf. p. 72).

[Zhang, 2019] P. Zhang, L. Xiong, Z. Yu, P. Fang, S. Yan, J. Yao et al. « Reinforcement learning-based end-to-end parking for automatic parking system ». *Sensors* 19 (2019), p. 3996 (cf. p. 72).

[Zhang, 2017] Shanshan Zhang, Rodrigo Benenson, Mohamed Omran, Jan Hosang et Bernt Schiele. « Towards reaching human performance in pedestrian detection ». *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), p. 973-986 (cf. p. 54).

[Zhang, 2023] Yuxiao Zhang, Alexander Carballo, Hanting Yang et Kazuya Takeda. « Perception and sensing for autonomous vehicles under adverse weather conditions : A survey ». *ISPRS Journal of Photogrammetry and Remote Sensing* 196 (2023), p. 146-177 (cf. p. 3).

[Zhao, 2022] Yinuo Zhao, Kun Wu, Zhiyuan Xu, Zhengping Che, Qi Lu, Jian Tang et al. « Cadre : A cascade deep reinforcement learning framework for vision-based autonomous urban driving ». *Proceedings of the AAAI Conference on Artificial Intelligence*. T. 36. 3. 2022, p. 3481-3489 (cf. p. 69, 131, 132).

[Zheng, 2018] K. Zheng et S. Rrt Liu. « based path planning for autonomous parking of vehicle ». *2018 IEEE 7th Data Driven Control And Learning Systems Conference (DDCLS)* (2018), p. 627-632 (cf. p. 12, 71).

[Zhu, 2017] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei et al. « Target-driven visual navigation in indoor scenes using deep reinforcement learning ». *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, p. 3357-3364 (cf. p. 76).

[Zhuang, 2018] Y. Zhuang, Q. Gu, B. Wang, J. Luo, H. Zhang et W. Robust Auto-parking Liu. Reinforcement Learning based Real-time Planning Approach with Domain Template, 2018 (cf. p. 72).

[Złotowski, 2017] Jakub Złotowski, Kumar Yogeeswaran et Christoph Bartneck. « Can we control it ? Autonomous robots threaten human identity, uniqueness, safety, and resources ». *International Journal of Human-Computer Studies* 100 (2017), p. 48-54 (cf. p. 54).

Titre : Apprentissage par renforcement profond neuro-symbolique pour une conduite urbaine sûre à l'aide de capteurs à faible coût.

Mots clés : Apprentissage par Renforcement Hiérarchique, Voiture Autonome, Conduite Urbaine

Résumé : La recherche effectuée dans cette thèse concerne le domaine de la conduite urbaine sûre, en utilisant des méthodes de fusion de capteurs et d'apprentissage par renforcement pour la perception et le contrôle des véhicules autonomes (VA). L'évolution généralisée des technologies d'apprentissage automatique ont principalement propulsé la prolifération des véhicules autonomes ces dernières années. Cependant, des progrès substantiels sont nécessaires avant d'atteindre une adoption généralisée par le grand public. Pour accomplir son automatisation, les véhicules autonomes nécessitent l'intégration d'une série de capteurs coûteux (e.g. caméras, radars, LiDAR et capteurs à ultrasons). En plus de leur fardeau financier, ces capteurs présentent une sensibilité aux variations telles que la météo, une limitation non partagée par les conducteurs humains qui peuvent naviguer dans des conditions diverses en se fiant à une vision frontale simple. Par ailleurs, l'avènement des algorithmes neuronaux de prise de décision constitue l'intelligence fondamentale des véhicules autonomes. Les solutions d'apprentissage profond par renforcement, facilitant l'apprentissage de la politique du conducteur de bout en bout, ont trouvé application dans des scénarios de conduite élémentaires, englobant des tâches telles que le maintien dans la voie, le contrôle de la direction et la gestion de l'accélération. Cependant, il s'avère que ces algorithmes sont coûteux en temps d'exécution et nécessitent de large ensembles de données pour un entraînement efficace. De plus, la sécurité doit être prise en compte tout au long des phases de développement et de déploiement des véhicules autonomes. La première contribution de cette thèse améliore la localisation des véhicules en fusionnant les mesures des capteurs GPS et IMU avec une adaptation d'un filtre de Kalman, ES-EKF, et une réduction du bruit des mesures IMU. L'algorithme est déployé et testé en utilisant des données de vérité terrain sur un microcontrôleur embarqué, le STM32 Nucleo, et a atteint un niveau de précision de 92 % sur une route en conditions réelles avec un temps d'exécution d'environ 20 μs. Ce pourcentage représente la proportion de points de données où la position estimée (X, Y) par l'ES-EKF se trouve dans un intervalle de confiance de 0,5 mètre par rapport à la position réelle correspondante. La deuxième contribution propose l'algorithme DPPO-IL (Dynamic Proximal Policy Optimization with Imitation Learning), conçu pour faciliter le stationnement automatisé en accordant une attention toute particulière à la sécurité. Cet algorithme apprend à exécuter des manœuvres de stationnement optimales tout en naviguant entre des d'obstacles statiques et dynamiques grâce à un entraînement complet intégrant des données simulées et réelles. La troisième contribution est un framework de conduite urbaine de bout en bout appelé Guided Hierarchical Reinforcement Learning (GHRL). Il intègre des données de vision et de localisation ainsi que des démonstrations d'experts exprimées avec des règles ASP (Answer Set Programming) pour guider la politique d'exploration de l'apprentissage par renforcement hiérarchique et accélérer la convergence de l'algorithme. Lorsqu'une situation critique se produit, le système s'appuie également sur des règles liées à la sécurité pour faire des choix judicieux dans des conditions imprévisibles ou dangereuses. GHRL est évalué sur le jeu de données NoCrash du simulateur Carla et les résultats montrent qu'en incorporant des règles logiques, GHRL obtient de meilleures performances que les algorithmes de l'état de l'art.

Title : Neuro-symbolic Deep Reinforcement Learning For Safe Urban Driving Using Low-Cost Sensors.

Keywords : Hierarchical Reinforcement Learning, Self Driving Car, Safe Urban Driving

Abstract : The research conducted in this thesis is centered on the domain of safe urban driving, employing sensor fusion and reinforcement learning methodologies for the perception and control of autonomous vehicles (AV). The evolution and widespread integration of machine learning technologies have primarily propelled the proliferation of autonomous vehicles in recent years. However, substantial progress is requisite before achieving widespread adoption by the general populace. To accomplish its automation, autonomous vehicles necessitate the integration of an array of costly sensors, including cameras, radars, LiDARs, and ultrasonic sensors. In addition to their financial burden, these sensors exhibit susceptibility to environmental variables such as weather, a limitation not shared by human drivers who can navigate diverse conditions with a reliance on simple frontal vision. Moreover, the advent of decision-making neural network algorithms constitutes the core intelligence of autonomous vehicles. Deep Reinforcement Learning solutions, facilitating end-to-end driver policy learning, have found application in elementary driving scenarios, encompassing tasks like lane-keeping, steering control, and acceleration management. However, these algorithms demand substantial time and extensive datasets for effective training. In addition, safety must be considered throughout the development and deployment phases of autonomous vehicles. The first contribution of this thesis improves vehicle localization by fusing data from GPS and IMU sensors with an adaptation of a Kalman filter, ES-EKF, and a reduction of noise in IMU measurements. This method excels in urban environments marked by signal obstructions and elevated noise levels, effectively mitigating the adverse impact of noise in IMU sensor measurements, thereby maintaining localization accuracy and robustness. The algorithm is deployed and tested employing ground truth data on an embedded microcontroller, the STM32 Nucleo, and has achieved an accuracy level of 92% on a real-world road and a swift execution time of around 20 μs. This percentage represents the proportion of data points where the estimated position (X, Y) from the ES-EKF falls within a confidence level of a 0.5m distance threshold of the corresponding ground truth position. The second contribution introduces the DPPO-IL (Dynamic Proximal Policy Optimization with Imitation Learning) algorithm, designed to facilitate end-to-end automated parking while maintaining a steadfast focus on safety. This algorithm acquires proficiency in executing optimal parking maneuvers while navigating static and dynamic obstacles through exhaustive training incorporating simulated and real-world data. The third contribution is an end-to-end urban driving framework called GHRL. It incorporates vision and localization data and expert demonstrations expressed in the Answer Set Programming (ASP) rules to guide the hierarchical reinforcement learning (HRL) exploration policy and speed up the learning algorithm's convergence. When a critical situation occurs, the system relies on safety rules, which empower it to make prudent choices amidst unpredictable or hazardous conditions. GHRL is evaluated on the Carla NoCrash benchmark, and the results show that by incorporating logical rules, GHRL achieved better performance over state-of-the-art algorithms.

Institut Polytechnique de Paris
91120 Palaiseau, France