Karlsruhe Institute of Technology

Reinforcement Learning for Controlled Traffic Rule Exceptions

Master Thesis

Jing Qin

Department of Electrical Engineering and Information Technology
Institut fuer Technik der Informationsverarbeitung
and
FZI Research Center for Information Technology

Reviewer:	Prof. Dr.–Ing. Eric Sax
Second reviewer:	Prof. Dr.–Ing. J. Marius Zöllner
Advisor:	Daniel Bogdoll, M.Sc.

Research Period: 19.Dezember 2022 – 19.Juni 2023

Affirmation

I declare truthfully that I have written this thesis independently, that I have indicated all the aids used completely and accurately, and that I have indicated everything that has been taken from the work of others, either unchanged or with modifications.

Karlsruhe, *Jing Qin*
im Juni 2023

Abstract

Autonomous vehicles have the potential to revolutionize modern transportation systems. However, ensuring the safe and efficient operation of autonomous vehicles in complex traffic environments, especially those in traffic rule exception scenarios, is still a challenge. This thesis presents a novel approach to enhance the motion planning of autonomous vehicles in anomaly traffic scenarios through the integration of Deep Reinforcement Learning(DRL) with a structured rulebook.

The research begins by identifying the challenges faced by autonomous vehicles in coping with traffic rule exception scenarios, where traffic rules may differ from standard conditions. It then proceeds to present an in-depth literature review to gain insights into the current methods and traffic scenarios in motion planning and their limitations.

A method is then proposed, which leverages DreamerV3, a state-of-the-art DRL algorithm, to train an autonomous vehicle's driving policy. The method integrates trajectory generation as DRL output and a structured Rulebook as part of the reward function of the DRL algorithm. The structured rulebook aims to encode the rules in a way that reflects the priority between rules, while its integration aims to improve the ability of agents to comply with these rules, while also allowing for transient rule violations in traffic rule exception scenarios by reflecting the priority between rules.

The proposed method was rigorously tested using the CARLA simulation environment. Multiple training scenarios with varying complexity were designed to evaluate the effectiveness and robustness of the method in handling different anomaly traffic rule exception scenarios. The experimental results demonstrate that the proposed method outperforms the traditional control command-based methods and DRL methods without rulebook integration, in terms of both the learning curve and the final performance.

Though the research shows promising results, it is acknowledged that there are limitations regarding the scalability of the rulebook integration and the effectiveness of the method at higher speeds. Suggestions for future work include refining the rulebook integration process, investigating its effectiveness in high-speed scenarios, and exploring ways to automate the tuning of custom coefficients in the reward function.

In summary, this master thesis contributes to the field of autonomous driving by proposing a method that combines DRL with structured Rulebook, thereby improving the performance of autonomous vehicles in traffic rule exception scenarios. The research provides valuable insights and sets a foundation for future work in the development of more robust and efficient autonomous driving systems.

Contents

1 Introduction

The emergence of autonomous vehicles represents a key development in the field of transportation. These vehicles, equipped with cutting-edge technologies, have the potential to significantly reduce traffic accidents, thereby enhancing the safety of all vehicles and individuals on the road. However, the path to this safer future is fraught with complexities. One of the major challenges is how to integrate these autonomous vehicles into the existing traffic rule framework, which involves both regular traffic rule scenarios and traffic rule exception scenarios. This implies a higher level of complexity in the decision-making process of autonomous vehicles. The unpredictable nature of real-world traffic scenarios often demands a flexible approach to established traffic rules. In certain cases, deviations from these rules may be unpreventable to ensure safety, similar to the decisions often made by experienced human drivers. This thesis delves into this issue and investigates an approach to improve the motion planning of autonomous vehicles in traffic rule exception scenarios through deep reinforcement learning.

1.1 Motivation

Traffic rules, usually expressed in legal or regulatory language, are not easily comprehensible for machines. This poses a significant challenge for the development and implementation of autonomous vehicles, as they must be able to understand and adhere to these rules to ensure safe and orderly traffic. Several strategies have been proposed to translate these rules into a language that machines can understand, for use in autonomous driving, however, these methods often overlook the importance of prioritizing different rules, a factor that is critical in real-world traffic scenarios [40], and the prioritization of rules often becomes a crucial factor affecting vehicle decisions when dealing with traffic rule exception scenarios [8]. Additionally, in current DRL applications for autonomous vehicles, the design of reward functions primarily focuses on the state parameters of the ego vehicle, often without considering the inclusion of traffic rules as part of the reward function [25].

In this context, it is observed that most DRL research applications in autonomous driving consider environments involving regular traffic scenarios, such as highways or typical urban traffic [2], where traffic rules are often set as constraints to regulate vehicle behavior. Traffic rule exception scenarios, on the other hand, have not been extensively studied, and in such scenarios, the prioritization of traffic rules should be emphasized [8].

Moreover, since DRL has been applied to autonomous vehicles, it has demonstrated reliable performance at the behavior planning level [28], for example, Xu et al. [46] studied using DRL to train autonomous vehicles for overtaking decisions on highways, and achieved good performance. On the other hand, some research has been designed using an end-to-end approach with DRL, where this scheme takes raw sensor inputs and directly outputs vehicle control commands, thereby accomplishing vehicle decision-making and motion planning in traffic environments [42]. However, little research has explored the use of DRL to directly generate driving trajectories and use controllers to follow them [28].

Given these limitations, it clearly makes sense to explore a DRL-based approach that takes into account hierarchical and scalable traffic rules, generates interpretable motion trajectories, and tracks them using a controller to train autonomous vehicles to perform proper motion planning under traffic rule exceptions.

1.2 Contribution

This thesis contributes to the field by applying deep reinforcement learning, specifically based on the DreamerV3 model [17], to train self-driving vehicles and enhance their performance and safety in scenarios involving traffic rule exceptions. The primary contributions of this thesis are twofold.

Firstly, we formalize traffic rules using a "Rulebook", a machine-understandable language that assigns precedence to different rules. This Rulebook is scalable and is designed to be readily incorporated into a reward function in reinforcement learning.

Second, we use DreamerV3 to learn the motion trajectory of the vehicle that will be tracked by a controller in real time. Instead of behavior planning or direct outputting control commands as most approaches do, we choose to generate proper trajectories to guide the vehicle's motion in real time. This approach is expected to accomplish more appropriate motion planning and improve the overall performance of the self-driving vehicle.

In summary, this thesis presents a novel approach to enhancing the motion planning capabilities of self-driving vehicles in the face of traffic rule exception scenarios. By combining deep reinforcement learning with a scalable and machine-understandable Rulebook and trajectory generator, we aim to make some progress in the safe and efficient integration of self-driving cars into existing traffic systems.

In the following chapters, the content will be organized in this way: Chapter 2 Background explained some basic theoretical background and necessary prerequisite knowledge on which this thesis is based. Chapter 3 Literature Review introduced the state-of-the-art research and studies related to this thesis. Chapter 4 Method listed the framework and details of the method used in this thesis. Chapters 5 and 6 respectively completed the design of the experiment setup and the analysis of the experimental results. Finally, in Chapter 7, the achievements and shortcomings of this thesis are summarized, and possible future work is proposed.

2 Background

In this chapter, the necessary background information that forms the foundation of this thesis will be presented. First, a brief introduction to the basic concepts and theories of Reinforcement Learning (RL) is provided, which is a type of machine learning focusing on how agents should take actions in an environment to maximize a certain cumulative reward. Following that, Deep Reinforcement Learning is introduced, an extension of Reinforcement Learning; it combines neural networks with reinforcement learning and has shown promise in addressing autonomous driving problems. Next, the chapter shifts its focus to Linear Temporal Logic (LTL), discussing the syntax and semantics of LTL and its application in formulating traffic rules. The Frenet Frame is then introduced as a coordinate system that is useful in path planning for autonomous vehicles. Additionally, this chapter delves into the PID Controller, a fundamental control strategy that helps adjust control inputs and is widely used in autonomous driving. Lastly, the chapter concludes with a discussion on exceptions to traffic rules, an important concept to consider as it entails situations where vehicles may have to adapt to unconventional traffic conditions. This background sets the stage for in-depth discussions and developments in the subsequent chapters.

2.1 Reinforcement Learning

Reinforcement Learning is a class of machine learning algorithms where an agent learns to make decisions by interacting with an environment. Unlike supervised learning, RL does not require a dataset of labeled examples, and the learning is guided by feedback signals in terms of rewards or penalties [30].

In reinforcement learning, the interaction between the agent and the environment is typically modeled as a Markov Decision Process (MDP). An MDP is a mathematical framework used to describe an environment for RL. The core components of an MDP are states, actions, rewards, state transitions, and policies. The relationship between state, action, and reward can be represented in Figure **??**. The state space is a set of all possible states in which the environment can exist. At each time step t, the environment is in some state $s_t \in S$. The action space is the set of all actions that the agent can take. An action $a_t \in A$ is taken by the agent based on the current state s_t. After taking an action, the agent receives a scalar reward r_{t+1} from the environment. The reward is a numerical value that represents the immediate benefit of taking action a_t in state s_t. The environment transitions from the current state s_t to a new state s_{t+1} according to a transition probability distribution, $P(s_{t+1}|s_t, a_t)$. A policy is a strategy that the agent employs to determine the next action based on the current state. It can be deterministic or stochastic [32].

The agent's goal is to learn a policy $\pi(a|s)$ that maximizes the expected cumulative reward over time, often referred to as the return. The return G_t is a sum of the rewards obtained after a time

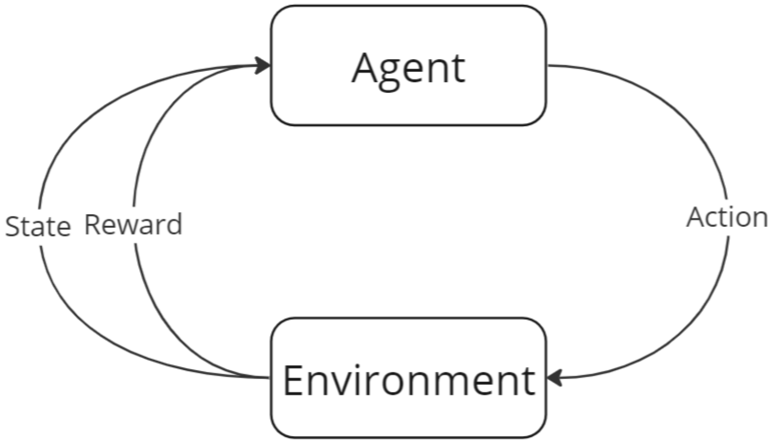


Figure 2.1: The interaction of an agent with its environment based on MDP. The agent observes the current state of the environment. Based on the state, the agent takes an action. The environment responds by updating its state and giving the agent a reward. The agent uses this reward to evaluate how good the action was and to make better decisions in the future.

step t, and it is mathematically represented as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad [2.1]$$

where γ is a discount factor between 0 and 1 that trades off the importance of immediate and future rewards.

Value functions estimate how good it is for an agent to be in a certain state or to perform a certain action in a state, given a policy. The state-value function $V^{\pi}(s)$ of an MDP is the expected return starting from state s, and subsequently following policy π:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G_t | s_t = s\right] \quad [2.2]$$

The action-value function $Q^{\pi}(s,a)$ is the expected return starting from state s, taking action a, and subsequently following policy π:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[G_t | s_t = s, a_t = a\right] \quad [2.3]$$

Q-learning is an off-policy algorithm that seeks to find the optimal action-value function. The update rule is based on the Bellman equation:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right] \quad [2.4]$$

Policy gradient methods aim to learn a policy directly by optimizing its parameters through gradient ascent on the expected return. A common approach is the REINFORCE algorithm, which utilizes the log-likelihood of the chosen action and the received reward.

RL approaches can be categorized into two main classes: model-free and model-based methods.

The key distinction lies in the employment of a model of the interactions between the robot and the environment. Model-free methods do not rely on a model and derive rewards and optimal actions through trial-and-error with the physical system. On the other hand, model-based methods utilize a model of the transition dynamics to derive rewards and optimal actions. Policies are optimized based on the model and then applied to the physical system.

In the context of reinforcement learning (RL), model-based approaches offer different perspectives compared to model-free methods by utilizing a learned model of the environment dynamics. This allows for reduced reliance on direct interactions with the real environment, which can be limited due to safety concerns or cost considerations. By learning a model for environment dynamics, agents can perform exploration and make action selections based on the learned transition function T and reward function R.

Model-based RL methods, such as DynaQ[39] and R-max[6], aim to acquire knowledge of the environment dynamics to inform decision-making. The transition function T and reward function R are learned to facilitate action selection. By maintaining a model approximation of the environment, agents can store knowledge of its dynamics, leading to fewer interactions with the environment, which can sometimes be costly. In contrast, model-free RL approaches do not require explicit knowledge of the transition function or reward function. Instead, these methods directly sample the underlying MDP to estimate the value function.

While model-free methods have garnered significant scientific interest, they have the drawback of requiring trajectory sampling to derive the optimal policy, which can be a disadvantage when applied to real robots. In contrast, model-based approaches offer an alternative by deriving the optimal policy based on internal simulations using a learned forward model representing the robot's dynamics. This characteristic significantly reduces the physical interactions between the robot and its environment, resulting in reduced mechanical wear. However, a major challenge for model-based RL algorithms is the accurate representation of the transition dynamics by the model. The advantages and disadvantages of the two categories of RL algorithms are outlined in Table2.1.

In summary, model-based RL approaches provide a means to reduce the reliance on direct interactions with the real environment by learning a model of the environment dynamics. These methods offer advantages such as reduced mechanical wear but depend heavily on the accuracy of the learned model in representing the transition dynamics.

RL Methods	Advantages	Disadvantages
Model-based RL	- Reduced interactions between the robot and the environment - Faster convergence to optimal solutions	- Significant impact of model accuracy on learning tasks - Dependence on transition models
Model-free RL	- Ease of implementation - No requirement for prior knowledge of transitions	- Higher risk of damage - Increased wear and tear of the robot - Slower learning convergence

Table 2.1: Advantages and disadvantages of model-based and model-free RL methods [21]

2.2 Deep Reinforcement Learning

DRL is an advanced branch of reinforcement learning, where deep learning techniques are utilized to learn complex patterns from high-dimensional data and assist the agent in making decisions. In DRL, neural networks are often used as function approximators to represent the policy or the value function, making it possible to deal with continuous state and action spaces which are typical in real-world applications, such as autonomous driving.

The core idea behind DRL is still to maximize the expected cumulative reward. The Q-function in DRL is often approximated by a neural network, which takes the state and action as input and outputs the expected return. The objective is to minimize the loss between the predicted Q-value and the target Q-value:

$$L(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta)\right)^2\right] \qquad [2.5]$$

where θ and θ' are the parameters of the current and target Q-networks respectively, s and a are the current state and action, s' is the next state, r is the reward, and γ is the discount factor.

In addition to Q-learning based methods, policy optimization methods are also prevalent in DRL. These methods aim to directly optimize the policy function $\pi(a|s;\theta)$ to maximize the expected return. For example, the Proximal Policy Optimization (PPO) algorithm is popular due to its sample efficiency and ease of implementation[37].

Model-based DRL involves the use of learned models of the environment to improve the efficiency of learning. Instead of learning solely from interactions with the environment, which can be costly, a model of the environment is learned and used to simulate transitions. This allows the agent to plan ahead by considering the consequences of actions.

When it comes to autonomous driving, DRL can be an essential component. The high-dimensional input space, including camera images, Lidar data, and sensor readings, can be handled efficiently using deep neural networks. Moreover, DRL can be used to learn driving policies that are capable of handling complex traffic scenarios. For instance, DRL can be employed to train autonomous vehicles in tasks such as lane keeping, overtaking, and navigating through traffic intersections. The end-to-end approach, which maps raw sensor inputs to control commands, is a promising direction in using DRL for autonomous driving, as it eliminates the need for hand-crafted features and rule-based decision-making.

In summary, Deep Reinforcement Learning has shown the potential to tackle complex decision-making tasks by combining the strengths of reinforcement learning with deep neural networks. Its application in autonomous driving is a promising avenue for developing intelligent and adaptive control policies for vehicles.

2.3 Linear Temporal Logic

Linear Temporal Logic is a mathematical language often used to describe the behavior of systems over time. The term "Linear" in LTL reflects the conception of time as a linear progression of discrete moments or states. The “Temporal” aspect emphasizes the logic’s focus on the temporal properties of systems, enabling expressions that encompass time-based behaviors and conditions.

Linear Temporal Logic Syntax

Let **P** be a set of atomic propositions. An atomic proposition is a primitive, indivisible unit of propositional logic that can assume either a true or false value. The LTL syntax is defined as follows[4]:

- If $p \in \mathbf{P}$, then p is an LTL formula.
- If φ is an LTL formula, then $\neg\varphi$ is an LTL formula (Negation).
- If φ_1 and φ_2 are LTL formulas, then $(\varphi_1 \wedge \varphi_2)$ is an LTL formula (Conjunction).
- If φ_1 and φ_2 are LTL formulas, then $(\varphi_1 \vee \varphi_2)$ is an LTL formula (Disjunction).
- If φ_1 and φ_2 are LTL formulas, then $(\varphi_1 \rightarrow \varphi_2)$ is an LTL formula (Implication).
- If φ is an LTL formula, then $\mathbf{X}\varphi$ is an LTL formula (Next).
- If φ is an LTL formula, then $\mathbf{G}\varphi$ is an LTL formula (Globally).
- If φ is an LTL formula, then $\mathbf{F}\varphi$ is an LTL formula (Eventually).
- If φ_1 and φ_2 are LTL formulas, then $(\varphi_1\mathbf{U}\varphi_2)$ is an LTL formula (Until).

In the syntax delineated above, the logical connectives are standard: $\neg$ stands for negation, $\wedge$ for conjunction, $\vee$ for disjunction, and → for implication. The temporal operators include:

- **Next** (**X**): $\mathbf{X}\varphi$ asserts that φ is true at the next state.
- **Globally** (**G**): $\mathbf{G}\varphi$ asserts that φ is true at all future states.
- **Eventually** (**F**): $\mathbf{F}\varphi$ asserts that there exists a future state where φ is true.
- **Until** (**U**): $\varphi_1\mathbf{U}\varphi_2$ asserts that φ_1 holds until φ_2 holds.

Linear Temporal Logic Semantics

The semantics of LTL define how to interpret LTL formulas over sequences of states (or models) that typically represent executions of a system. A sequence of states is an infinite sequence $s_0, s_1, s_2, ...$ where each s_i represents the state of the system at time i.

An LTL formula is evaluated over such sequences and the evaluation defines whether the sequence satisfies the formula, often written as $(s_0, s_1, s_2, ...) \models \varphi$, where φ is the LTL formula.

The semantics for atomic propositions $p \in \mathbf{P}$ is defined as $(s_0, s_1, s_2, ...) \models p$ if and only if p holds in state s_0.

The semantics for logical connectives in LTL is similar to classical propositional logic.

- $(s_0, s_1, s_2, ...) \models \neg\varphi$ if and only if $(s_0, s_1, s_2, ...) \not\models \varphi$.
- $(s_0, s_1, s_2, ...) \models \varphi_1 \wedge \varphi_2$ if and only if $(s_0, s_1, s_2, ...) \models \varphi_1$ and $(s_0, s_1, s_2, ...) \models \varphi_2$.
- $(s_0, s_1, s_2, ...) \models \varphi_1 \vee \varphi_2$ if and only if $(s_0, s_1, s_2, ...) \models \varphi_1$ or $(s_0, s_1, s_2, ...) \models \varphi_2$.
- $(s_0, s_1, s_2, ...) \models \varphi_1 \rightarrow \varphi_2$ if and only if $(s_0, s_1, s_2, ...) \not\models \varphi_1$ or $(s_0, s_1, s_2, ...) \models \varphi_2$.

The semantics for temporal operators involve evaluations over multiple states in the sequence.

- $(s_0, s_1, s_2, ...) \models \mathbf{X}\varphi$ if and only if $(s_1, s_2, s_3, ...) \models \varphi$.
- $(s_0, s_1, s_2, ...) \models \mathbf{G}\varphi$ if and only if for all $i \geqslant 0$, $(s_i, s_{i+1}, s_{i+2}, ...) \models \varphi$.
- $(s_0, s_1, s_2, ...) \models \mathbf{F}\varphi$ if and only if there exists $i \geqslant 0$ such that $(s_i, s_{i+1}, s_{i+2}, ...) \models \varphi$.
- $(s_0, s_1, s_2, ...) \models \varphi_1 \mathbf{U} \varphi_2$ if and only if there exists $i \geqslant 0$ such that $(s_i, s_{i+1}, s_{i+2}, ...) \models \varphi_2$ and for all $0 \leqslant j < i$, $(s_j, s_{j+1}, s_{j+2}, ...) \models \varphi_1$.

These semantics essentially define how the temporal operators are interpreted over sequences of states. They are central to how LTL formulas are used to specify and reason about the temporal properties of systems, particularly in the verification of hardware and software systems.

Traffic Rules in Linear Temporal Logic

In the domain of autonomous driving, the formalization of traffic rules using Linear Temporal Logic (LTL) provides a rigorous mathematical framework to ensure safe and legal driving behaviors. By representing traffic rules as LTL formulas, we can capture both temporal and logical aspects of the rules. For instance, the “keep lane” rule can be formalized by employing the "Globally" operator, **G**, such that:

$$\mathbf{G}(\text{in\_lane}) \tag{2.6}$$

This mandates that the vehicle must remain in its lane at all times. Similarly, the rule for avoiding collisions can be translated into LTL as indicating that at all points in time, the vehicle must not be in a state where a collision occurs:

$$\mathbf{G}(\neg\text{collision}) \tag{2.7}$$

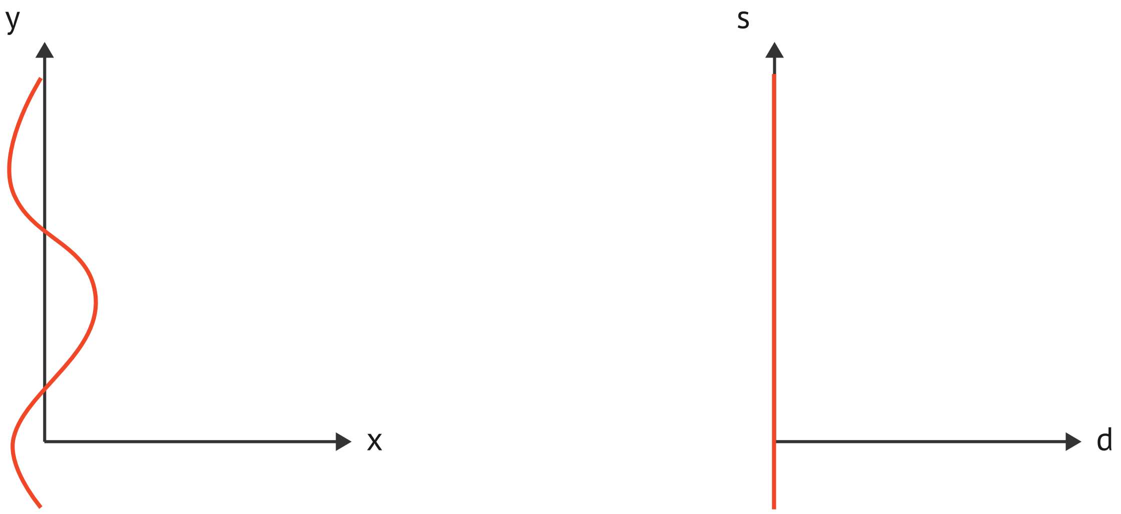
More complex rules can be represented using a combination of LTL operators. For instance, to ensure that a vehicle remains at a safe distance from other vehicles until it reaches its destination, the "Until" operator, **U**, can be employed:

$$\mathbf{G}(\text{safe\_distance}\mathbf{U}\text{destination\_reached}) \tag{2.8}$$

Such formalization enables the validation and verification of autonomous driving systems by ensuring that the behaviors of the system do not violate the stipulated traffic rules under any temporal sequence of states. Additionally, employing LTL for traffic rules paves the way for automated synthesis of control strategies that conform to these rules, and for the execution of rigorous simulation-based testing and formal verification methods.

2.4 Frenet Frame

Frenet frame is a concept that is commonly employed in autonomous driving and mobile robotics to represent the position and motion of objects on a path in a manner that is more intuitive compared to Cartesian coordinates, the comparison of two kinds of coordinates can be seen in Figure 2.2. In Frenet space[44], an object's coordinates are denoted using two primary components: the longitudinal position, often denoted as s, and the lateral offset from the path, typically represented as d. This coordinate frame moves along the path, which renders it particularly beneficial in situations where describing motion in terms of movement along and across a path, such as lane-keeping and overtaking maneuvers in autonomous driving, is more natural.



(a) Cartesian Coordinate

(b) Frenet Coordinate

Figure 2.2: Two different coordinates

The Frenet frame can be conceptualized as a moving coordinate system that travels along a reference path. At any point on the path, the s coordinate signifies the distance traversed along the path from a reference starting point, while the d coordinate measures the perpendicular distance from the path. This implies that for an object moving exactly along the reference path, d would be zero.

Frenet coordinates are especially advantageous when dealing with curved paths or when the behavior of an object relative to the path is of significance. In the traffic rule exception scenarios explored in this thesis, the reasonable behavior that needs to be achieved by the self-driving car is mainly achieved through lane changes and, that is, the trajectory that needs to be achieved by the vehicle has a strong dependence on the path. Generating local trajectories based on the Frenet frame in real-time and following them using a controller becomes a reasonable choice.

2.5 PID Controller

The PID controller is one of the most widely used control strategies in industrial control systems and various other applications due to its simplicity and effectiveness. PID stands for Proportional, Integral, and Derivative, which are the three terms that make up the control algorithm. The PID controller is designed to eliminate the error between a desired setpoint and the actual output of a system.

Mathematically, the output control signal, $u(t)$, of a PID controller can be expressed as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad [2.9]$$

Where,

- $u(t)$ is the control output at time t.
- $e(t)$ is the error signal at time t, defined as $e(t) = r(t) - y(t)$, where $r(t)$ is the reference or setpoint and $y(t)$ is the system's actual output.
- K_p, K_i, and K_d are the proportional, integral, and derivative gain constants respectively.
- The first term, $K_p e(t)$, is the Proportional term, which is proportional to the current error.
- The second term, $K_i \int_0^t e(\tau) d\tau$, is the Integral term, accounting for the accumulation of past errors.
- The third term, $K_d \frac{de(t)}{dt}$, is the Derivative term, considering the rate of change of the error.

The proportional term, $K_p e(t)$, provides a control action that is proportional to the error. The constant K_p determines how aggressively the controller responds to an error. However, a proportional controller alone cannot eliminate the steady-state error. The integral term is introduced to address the steady-state error. By considering the accumulation of past errors, it aims to bring the error to zero in the steady state. The constant K_i determines how aggressively the controller responds to accumulated errors. The derivative term predicts the future trend of the error by considering its rate of change. It can improve stability and reduce overshooting. The constant K_d determines how much the rate of change of the error affects the control action.

In practice, tuning the PID controller gains (K_p, K_i, K_d) is crucial for achieving the desired performance and stability of the system. Various methods are available for tuning these gains, including the Ziegler-Nichols method, Cohen-Coon method, and trial and error.

Due to the effectiveness and simplicity of PID controllers in handling various control problems, they have found widespread application in the field of autonomous vehicle control. Their ability to robustly minimize errors makes them particularly suitable for trajectory tracking tasks, where precision is critical. In this thesis, a PID controller is employed to control the steering and acceleration of the autonomous vehicle to accurately follow the trajectories generated by the motion planning algorithms.

2.6 Traffic Rule Exceptions

In the context of traffic regulations, there are instances where drivers may find themselves compelled to deviate from prescribed traffic rules due to certain situations. These exceptions arise when adherence to the rules could potentially lead to hazardous outcomes or endanger the safety of the individuals involved. Here we explore several examples of traffic rule exceptions and highlight the underlying rationale for their occurrence.

One common scenario giving rise to traffic rule exceptions involves the presence of emergency vehicles, such as ambulances or fire trucks, that require unimpeded access to their destination. When emergency vehicles approach with activated lights and sirens, drivers are obligated to yield the right of way and create a pathway for these vehicles to pass swiftly. In doing so, drivers may need to break traffic rules, such as stopping in a designated no-stopping zone or disregarding a red traffic light. The overriding concern in this situation is the preservation of life and the prompt delivery of emergency assistance, thus necessitating the violation of certain traffic regulations.

Another circumstance in which traffic rule exceptions occur is during road construction or maintenance activities. Temporary traffic rules and detours are often implemented to ensure the safety of both road workers and fellow drivers. Drivers must adapt to these modified regulations, which may involve deviating from their regular lanes or following alternative routes. By doing so, drivers contribute to the prevention of accidents and promote the smooth progress of construction activities.

Pedestrian safety represents another critical aspect of traffic rule exceptions. Drivers are expected to yield to pedestrians at designated crosswalks, providing them with a safe passage. However, unpredictable situations can arise, such as when pedestrians jaywalk or unexpectedly enter the road outside of designated crosswalks. In such cases, drivers may need to take evasive action to avoid a potential collision. This defensive maneuver might entail momentarily deviating from their lane or making sudden stops. By taking appropriate action to prevent harm, drivers prioritize the well-being of pedestrians over strict adherence to traffic rules.

Furthermore, the necessity to avoid collisions can lead to traffic rule exceptions. When confronted with a sudden swerve or braking of another vehicle, a driver may need to react swiftly to avert an imminent collision. This may involve making rapid decisions, such as changing lanes or applying forceful braking, which temporarily violate certain traffic regulations. The primary objective in such situations is to ensure the safety of all parties involved by mitigating the risk of a potential accident.

While traffic rules generally serve as a framework for maintaining order and ensuring road safety, these rule exceptions arise when adhering strictly to the regulations could lead to undesirable consequences. It is crucial for autonomous vehicles to understand and respond appropriately in these complex scenarios.

3 Literature Review

3.1 Deep Reinforcement Learning in Autonomous Driving

DRL has evolved into a potent technique for training agents to make intelligent decisions within complex environments. In this section, grounded on the principal subject of this thesis - training autonomous vehicles in motion planning through DRL under traffic rule exception scenarios, an analysis of the literature is conducted regarding the application of Deep Reinforcement Learning in autonomous driving, with a special focus on its utility in motion planning. Additionally, the section sheds light on the principal scenarios that have been under scrutiny in contemporary research. This evaluation is critical for understanding the current landscape of DRL applications in the field and envisioning the prospects of further innovations and advancements in the context of autonomous vehicles and their ability to adeptly navigate under anomalous traffic rule conditions.

3.1.1 Motion Planning

The motion planning of autonomous vehicles can be divided into several layers such as behavior planning, motion planning, control command, etc. [2]. In recent years, some progress has been made in the research of combining DRL with motion planning of autonomous driving.

The approach used by Fayjie et al. [12] focuses on rule-based systems for navigation and obstacle avoidance in urban environments for behavior-level motion planning. Ye et al. [47] propose an automatic lane-changing strategy using approximate policy optimization (PPO), a type of RL, that outperforms traditional rule-based approaches. Based on this, Hu et al.[19] extended it by considering the interaction between multiple agents and using multi-agent RL to negotiate the use of roads with other drivers. This approach is particularly important in merging scenarios, where interaction and negotiation with other drivers is crucial. outon et al. [5] demonstrate that their game-theoretic RL agent achieves more efficient merging compared to traditional RL training methods. This approach is particularly powerful in dense traffic scenarios. Wang et al. [42] proposed hierarchical behavior and motion planning, which combined classical motion planners with RL and significantly improved the performance of RL. Meanwhile, Cao et al. [7] addressed the challenge of highway exits by using a combination of RL and Monte Carlo tree search. This demonstrates the versatility of RL in adapting to different traffic scenarios. Similarly, Hoel et al. [18] applied deep RL to speed and lane change decisions for truck-trailer combinations, showing its applicability to different types of vehicles. Sun et al. [38] focus on semi-regular adaptive cruise control using a depth-deterministic policy gradient-based algorithm, however, this is also based on behavior-level planning. Guo et al. [14] investigate the integration of connected and autonomous vehicles (CAVs) in mixed traffic scenarios. Their approach, using cooperative lane-changing strategies for RL, demonstrates that the inclusion of CAVs can significantly improve traffic flow and speed.

All of the aforementioned methods combining DRL with motion planning for autonomous vehicles have been approached from the perspective of the behavior layer and have achieved commendable performance.

Aradi et al. [3] presented a study that revolves around the use of Policy Gradient reinforcement learning in autonomous highway driving. In their study, they focused on designing an end-to-end behavior control for a kinematic vehicle model in a simulated highway environment. Jaritz et al. [20]investigated end-to-end race driving using DRL with a focus on avoiding mediated perception, such as object recognition and scene understanding. They employed an Asynchronous Actor Critic (A3C) framework to learn car control using only RGB images from a forward-facing camera in a realistic rally game simulation. The research showed that the newly proposed reward and learning strategies led to faster convergence and more robust driving. Nageshrao et al. [31] approached autonomous highway driving with DRL considering the operational space diversity. The authors highlighted that traditional rule-based decision-making or a-priori cost functions may not be ideal in real-time scenarios for autonomous vehicles (AVs). To address this, they proposed a DRL-based method where the autonomous vehicle interacts with simulated traffic. They emphasized safety by incorporating a short horizon safety check to provide alternate safe actions in critical scenarios. This approach enhanced learning efficiency without compromising safety and optimality. Sallab et al. [36] explored end-to-end DRL for Lane Keeping Assist. Their research aimed to apply DRL to automotive applications, which at the time was a relatively new area of research. They proposed algorithms under two main categories: Discrete actions (Deep Q-Network Algorithm) and Continuous actions (Deep Deterministic Actor Critic Algorithm). The algorithms were tested on The Open Racing Car Simulator (TORCS) and demonstrated learning of autonomous maneuvering in complex road structures. Yu et al.[48] addressed intelligent land-vehicle model transfer trajectory planning through DRL. They aimed to solve model errors and tracking dependence in intelligent vehicle motion planning. They used an abstract model of the real environment and employed a Deep Deterministic Policy Gradient (DDPG) alongside a vehicle dynamic model for joint training. This approach allowed for obtaining effective control action sequences directly. The method improved the model's generalization performance and achieved continuous output, which also reduced lateral control errors.

In contrast, the literature mentioned above uses an end-to-end solution, using the raw sensor input directly as the DRL input and outputting control commands such as steer or acceleration. However, fewer studies have applied DRL to generate trajectories[2]. Feher et al. [13] utilized a Deep Deterministic Policy Gradient agent to create waypoints that a vehicle should adhere to. In the beginning, a waypoint list is formulated by the planner, which encompasses a set time duration and increment steps. Following this, DRL adjusts the lateral positions of these waypoints to carve an optimized path. A significant limitation of this method is its exclusive concentration on lateral planning while keeping the longitudinal route constant. This leads to the generation of a less than ideal trajectory that might not be well-suited for complex driving scenarios such as traffic rule exception scenarios.

3.1.2 Traffic Scenarios

As mentioned earlier, the majority of the literature focuses on regular traffic scenarios, such as highways or urban traffic environments. There is scant literature that takes into account traffic rule exception scenarios[2]. This represents a promising avenue for breakthroughs in research on autonomous vehicles in conjunction with DRL.

Talamini et al.[40] explore the use of DRL to train vehicle behaviour in a scenario where violation of traffic rules cannot be avoided, but their output for lateral motion stays at the behaviour planning level and does not incorporate traffic rules into the reward design in a structured way.

Therefore, the study of traffic rule exception scenarios is where the current research field is lacking and is the scenario that this thesis focuses on, and based on the results of the literature survey, a study based on this scenario is meaningful.

3.1.3 Dreamer

Dreamer is a model-based reinforcement learning algorithm developed by Google DeepMind [15], which has made significant strides in the world of artificial intelligence and machine learning. The Dreamer algorithm builds on the concept of model-based RL by incorporating learning from imagined trajectories. It features a unique approach, where it decouples the processes of model learning and policy learning, allowing it to optimize policies directly from predictions.

The core of Dreamer is its world model. The world model is a learned recurrent dynamics model which encodes the state of the environment in a latent space. This model predicts not just the immediate next state, but a whole sequence of future states, essentially "imagining" what would happen in the future given the current state and a sequence of actions. The world model is trained using backpropagation through time on sequences of states and actions collected by the policy.

Once the world model is trained, Dreamer proceeds to learn a policy. However, instead of learning from interaction with the environment, it learns entirely from the imagined trajectories generated by the world model. This is done by maximizing the expected return of the imagined trajectories using a variant of the actor-critic method. The actor, or policy, is a function that outputs actions given states, and the critic is a function that estimates the expected return given states and actions. The actor is trained to choose actions that maximize the critic’s estimates, while the critic is trained to accurately estimate the returns of the actor’s actions.

DreamerV3 is an improved version of the original Dreamer algorithm [17], it is a general and scalable algorithm for reinforcement learning that can master a wide range of domains with fixed hyperparameters, outperforming specialized algorithms. This outstanding feature makes it easier to train in unknown areas as well, without the need for extremely time-consuming hyperparameters tuning.

One significant feature of DreamerV3 is how it encodes inputs using symlog encoding. This technique is used in reconstructing inputs and also in making predictions for rewards. It’s a simplified, yet effective way to handle input data. Regarding the world model regularizer, DreamerV3 eliminates the need for fine-tuning a specific component called the KL regularizer. This was achieved by mixing two approaches, namely KL balancing and free bits. This combination removes the need to specify what a “good” error rate is, which can change depending on the domain. In the case of policy regularization, DreamerV3 smartly scales down large ranges to fit between 0 and 1, which helps in handling different kinds of rewards. It also filters out extreme cases to prevent skewing. This is especially useful in environments that have a random element.

Additionally, DreamerV3 uses a mix of uniform and neural network output for categorizing distributions in world model representations. This ensures that there is always a small chance for every category, keeping probabilities and divergences under control. In terms of structure, DreamerV3’s network is updated with layer normalization and SiLU activation functions. It’s somewhat similar to the previous version but has some upgrades that make it more robust. This robustness allows it to work with bigger networks, which helps in improving performance. For the critic EMA regularizer, DreamerV3 uses a faster network to calculate λ-returns. This approach saves time compared to an older method, making it more efficient.

Also, DreamerV3 modifies the replay buffer. Unlike DreamerV2 [16], which replays certain time steps, DreamerV3 samples from all the data it receives. This makes the training process faster because it can get feedback more quickly. Lastly, DreamerV3’s hyperparameters have been set to perform well in different environments. It has been tested in different domains like Atari, Control Suite, DMLab, and Minecraft, and showed that it can handle them well without additional modifications. This implies that it may also have potential in autonomous driving DRL applications based on image inputs.

In general, there has been large-scale research in the area of autonomous driving in terms of improving vehicle decision making and behavior planning using techniques such as RL and DRL. Research efforts have primarily concentrated on high-speed scenarios, such as following, overtaking, lane changing. Certainly, numerous researchers have also dedicated their efforts to studying urban traffic scenarios, such as behavior prediction and decision-making at intersections. While considerable advancements have been made in these areas, there remains a lack of comprehensive exploration and in-depth study of rule exception scenarios, which demand proper approaches to ensure the better performance of self-driving cars. This thesis aims to address this gap by investigating the application of reinforcement learning in handling these specific scenarios and proposing strategies to enhance the capabilities of autonomous vehicles in such situations.

3.2 Formalization of Traffic Rules

The formalization of traffic rules into a format that can be interpreted by autonomous vehicles is essential for ensuring road safety and regulatory compliance. Traditionally, traffic rules and associated laws have been documented in textual formats intended for human interpretation. For autonomous vehicles to operate safely and in accordance with traffic laws, it is necessary to convert these rules into a format that can be processed by computer algorithms. This conversion facilitates autonomous vehicles’ understanding of and adherence to traffic rules and enables them to navigate and coordinate effectively in traffic environments alongside human-driven vehicles. In literature, various methodologies have been proposed for formalizing traffic rules to ensure that autonomous vehicles can interpret and comply with these rules and associated driving ethics.

Censi et al. [8] developed a method termed "Rulebooks" to specify desired behaviors for autonomous vehicles. Rulebooks comprise pre-ordered sets of rules, where each rule acts as a violation metric based on outcomes or realizations. This structure imposes an ordering based on priority, allowing for the incorporation of legal, ethical, cultural, and various conflicting objectives into autonomous vehicle behavior.

An alternative approach is grounded in Temporal Logic, which facilitates the specification of time-based systems. Maierhofer et al.[26] utilized Temporal Logic to formalize traffic rules for interstates. They demonstrated that traffic rules could be mathematically expressed through Temporal Logic, enabling automatic and unambiguous validation of autonomous vehicles' adherence to these rules. Their work is centered on the German Road Traffic Regulation and the Vienna Convention on Road Traffic. Temporal Logic was also employed by Fainekos et al.[11] in robotic motion planning, proving its versatility.

Kloetzer and Belta [22] established an automated framework for controlling linear systems through linear temporal logic (LTL). By partitioning the state space and employing model checking, their approach identifies feedback control laws ensuring that closed-loop systems comply with LTL specifications. Zhang et al. [49] expanded on this concept by addressing challenges in designing controllers for linear systems influenced by disturbances.

Rizaldi et al.[35] proposed employing the Isabelle theorem prover to formalize and monitor autonomous vehicle behavior concerning traffic rules. They demonstrated the abstraction and codification of traffic rules and the concretization of each atomic proposition. In a subsequent work, Rizaldi and Althoff[34] explored the liability aspect in collisions involving autonomous vehicles by viewing traffic rules as requirements.

In cases where precision is difficult due to uncertainty, Morse et al. [29] presented a fuzzy approach for qualifying design exploration for autonomous systems. Through the use of fuzzy logic and model checking, they address vague and probabilistic requirements, respectively.

Esterle [10] presented an approach involving the identification and formalization of applicable traffic rules coupled with a simulation toolchain for evaluation. He also explored methods for interactive planning, ensuring traffic rule adherence.

Mehdipour et al. [27] offered a comprehensive review of formal methods applied to autonomous driving. They discussed the utilization of formal languages such as temporal logics to specify a wide range of driving behaviors, from safety to complex traffic rule compliance.

Xiao et al. [45] tackled the issue of optimizing complex specifications through optimal control strategies for autonomous vehicles. Their framework integrates Control Lyapunov Functions and Control Barrier Functions to converge to desired states and enforce safe interactions with other road users.

Aguilar et al. [1] addressed reward function shaping in neural network controllers for autonomous agents through reinforcement learning. They used Signal Temporal Logic (STL) rules and their quantitative semantics to systematically combine evaluations of multiple requirements into a single reward, taking into account priorities defined by the partial order.

Kress Gazit et al. [24] contributed to the field by focusing on the synthesis of robot behavior. They highlighted the importance of formal synthesis for robotics, which involves specifying tasks in a mathematically precise language and automatically generating robot controllers that are correct by construction. This approach not only minimizes implementation errors but also offers behavioral guarantees for the resulting controllers.

Van Den Hoven and Lokhorst [41] explored deontic logic as a means of supporting computer ethics, which can be extended to the ethical considerations necessary for autonomous vehicles. They proposed a logical model (DEAL) that combines deontic, epistemic, and action logic to create a structured moral discourse concerning issues in the field of information and communication technology. This approach might be of relevance in establishing ethical considerations in the behavior of autonomous vehicles.

In summary, various methods, such as rulebooks, temporal logic, linear temporal logic, and fuzzy logic, have been used to formalize and verify the traffic rules and behavior of autonomous systems. Each method has its merits and can be chosen according to the specific requirements and constraints in the design and operation of autonomous vehicles. However, in the studies explored above, some approaches consider the conversion of rules into a machine understandable language but do not consider the priority between different rules, some focus on the priority and hierarchical structure of the rules but not on the conversion of the language and the combination of reward functions, and some other approaches combine traffic rules into reward functions but the hierarchical structure of the different rules is ignored. Based on this, this thesis will attempt to express rules using suitable semantics and build rulebooks with hierarchical structures, based on which a scalable way to incorporate them into the design of reward functions for DRLs is proposed. The goal of this thesis is to give self-driving cars the necessary understanding of traffic rule exception scenarios scenarios and their corresponding rules so that they can make appropriate motion planning in such situations.

4 Method

In this chapter, the methodological architecture that forms the backbone of this master thesis is explored. This chapter is structured into four critical sections, each addressing an essential component of the proposed approach. The chapter begins with the Problem Statement section, which conveys the specific problem being addressed, highlighting its complexities and significance in the context of autonomous driving. Following this, the Rulebook Formalization section explores the process of translating traffic rules into a structured format that is conducive to computational interpretation. This section is further divided into two sub-sections - LTL Rulebook and From LTL Rulebook to Reward. The LTL Rulebook sub-section focuses on formalizing the rulebook using Linear Temporal Logic (LTL), while the From LTL Rulebook to Reward sub-section discusses the integration of the LTL Rulebook into reward design. The subsequent section, Frenet Frame Trajectory Generation, expounds on the methodology of generating vehicle trajectories in the Frenet frame, an essential aspect of motion planning in dynamic environments. Finally, the DreamerV3 section introduces the deep reinforcement learning model DreamerV3, shedding light on its capabilities and detailing how it is utilized in conjunction with the aforementioned components to achieve the objectives of this research. Through the combination of these components, Chapter 4 lays out a comprehensive methodological framework for autonomous driving in complex traffic scenarios.

4.1 Problem Statement

In autonomous driving systems, the ability to handle traffic rule exception scenarios wisely is essential. As discussed in Chapter 2, emergency vehicles, road construction, pedestrians suddenly entering the road, etc., are common traffic rule exception scenarios, in which momentarily breaking traffic rules is permitted, as this ensures compliance with higher-priority rules, such as vehicle or pedestrian safety. In the prior work of colleagues at FZI, a series of anomaly scenarios databases have been established, which express similar traffic rule exception scenarios under different environments and anomaly conditions: an autonomous vehicle driving in a lane, with an unforeseen obstacle, such as a stationary car, blocking its path, as shown in the Figure 4.1, which we refer to as an anomaly scenario. There is a conflict between the requirement to adhere to lane-keeping traffic rules and the need to avoid obstacles. If traffic rule exceptions are not permitted in this case, the vehicle may be prompted to stop, potentially obstructing the flow of traffic or even causing a collision, while an experienced human driver might choose to deviate from the lane and bypass the obstacle. This indicates that temporarily breaking a traffic rule is a sensible decision to pursue a more important objective, namely, to avoid the obstacle. This scenario is chosen by this thesis as the environment for autonomous driving DRL training and simulation, and this series of anomaly scenarios is used for the training and evaluation of the DRL agent. This thesis aims to explore the use of deep reinforcement learning as a tool to enable autonomous vehicles to understand and proficiently handle such situations. In addition, this thesis also investigates the generation of real-time trajectories using reinforcement learning, to facilitate more rational and optimized driving strategies for autonomous vehicles in traffic rule exception scenarios.

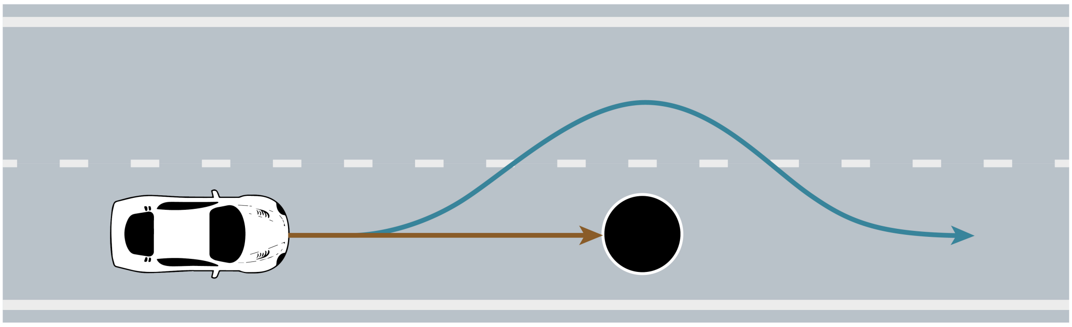


Figure 4.1: Anomaly traffic rule exception scenario. In the illustrated scenario, the self-driving car has to break the "lane keep" traffic rule in order to avoid obstacles.

Based on the problem described above, we model it as a Markov Decision Process. The state space consists of high-dimensional data representing images captured by the bird's eye view (BEV) camera fixed on the vehicle, with all data collected within the anomaly scenarios encompassed in the anomaly scenarios dataset. The action space comprises a set of trajectories generated based on the current state, and in this process, the action is to obtain a uniquely determined trajectory through a trajectory generator. When the vehicle takes an action (generates a trajectory), the transition to the next state is deterministic, as previously mentioned, with a transition probability of 1. The reward function integrates multiple considerations; it utilizes a rulebook to evaluate the alignment of the generated trajectory with traffic rules and assesses the vehicle's current states, such as collision state. DreamerV3 is the deep reinforcement learning agent used to learn an optimal policy that maps states to actions. Upon selecting a trajectory as an action, a PID controller is activated to guide the vehicle along this path, ensuring precise execution and movement to the next state. Through interaction with the anomaly scenarios and feedback from the reward function, the DRL agent optimizes the policy for effective motion planning within the given environment, this process can be shown by Figure 4.2. In the following sections, the establishment of the rulebook and reward function as well as the working mechanism of the trajectory generator will be discussed in detail; furthermore, the integration of DreamerV3 with the latter will also be explored.

4.2 Rulebook Formalization

In traffic environments, there are instances when not all traffic rules are fully satisfied, which are the rule exception scenarios mentioned earlier. In this context, it is meaningful to formalize the

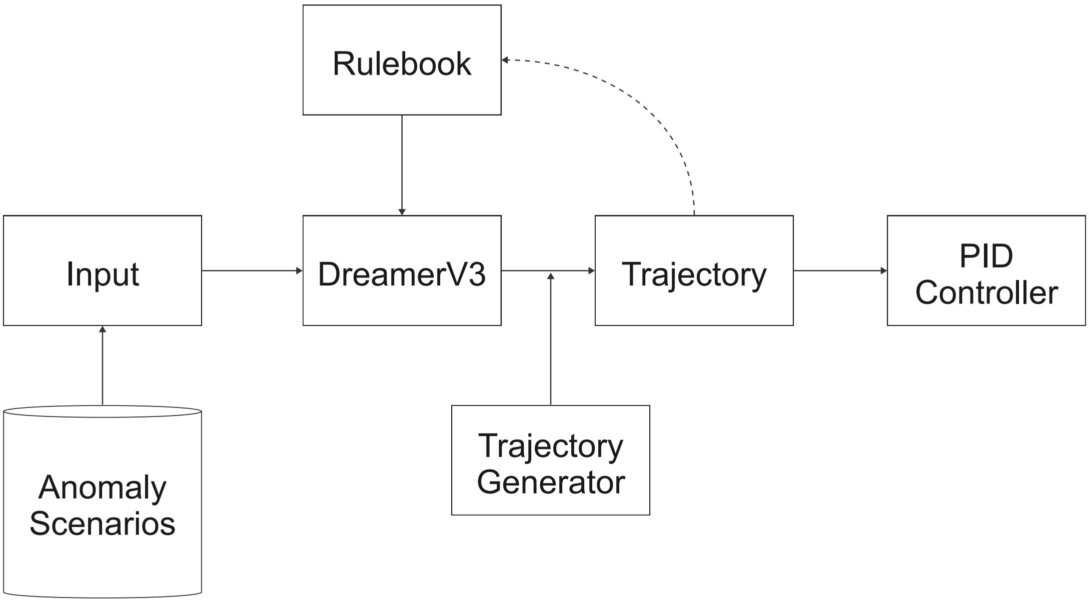


Figure 4.2: Methodology architecture. The camera sensor feeds the anomaly scenario into DreamerV3 as an observation. DreamerV3 then predicts actions to create a trajectory. A PID controller is used to track this trajectory in a constant velocity. In addition, the rulebook evaluates the trajectory and uses the results as part of the reward function.

rulebook that autonomous vehicles must adhere to. This is because such an approach reflects the hierarchical structure and priorities of the rules, a characteristic that, as previously mentioned, plays an important role in traffic rule exception scenarios. In Chapter 3, various methods of traffic rule formalization are explored and compared, and it is observed that in most studies, only each individual rule is translated from textual language to a language that a computer can understand, while the relationships between rules are not emphasized. However, the concept of the rulebook proposed by Censi et al. [8] addresses this issue well, as this formalization approach emphasizes the hierarchical structure of the rules and is beneficial for the environment being studied in this thesis. Building on this, this thesis will propose a further advanced scheme for constructing a rulebook, which on the one hand uses Linear Temporal Logic to describe the rules by evaluating the trajectory states, and on the other hand integrates the rulebook effectively into reward design through a sensible mathematical approach.

This formalization forms the basis upon which autonomous driving systems can establish the decision-making process of autonomous vehicles. The rulebook can include traffic rules and regulations as well as additional criteria that play a role in traffic scenarios. These criteria enable the system to rank the importance of various traffic rules and determine when it is wise to temporarily ignore lower-rung rules for the sake of higher-priority objectives, such as safety. This section will delve into how to create, construct, and implement such a rulebook in the context of deep reinforcement learning for autonomous vehicles. Additionally, it will explore methods to continuously update and expand this rulebook based on real-world experience and data.

4.2.1 LTL Rulebook

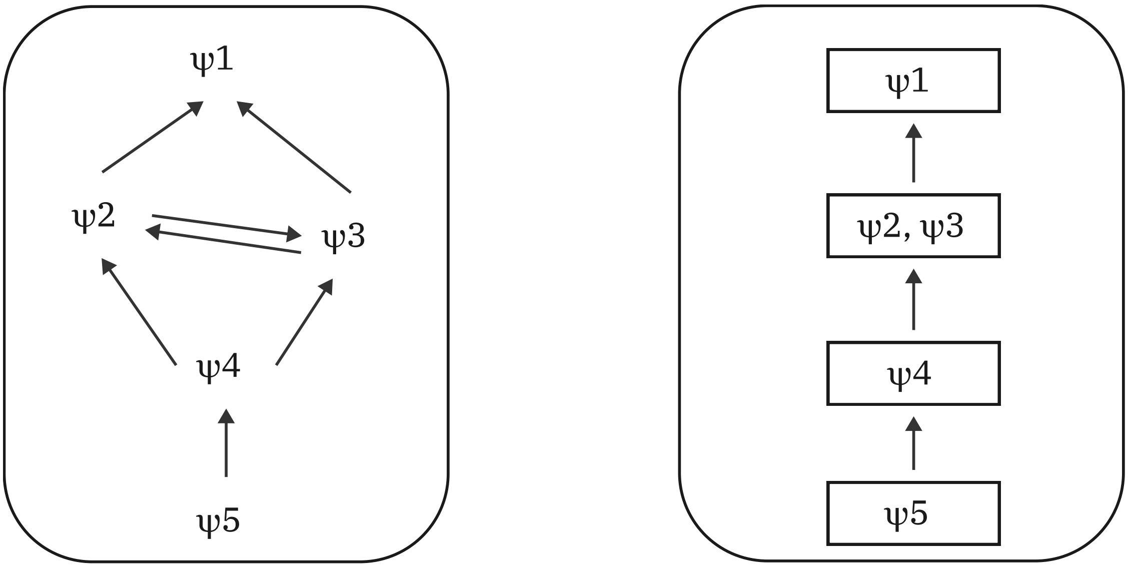
Censi et al.[8] put forward the notion of a rulebook as a formalism tailored for organizing an array of distinct rules. This framework proves to be valuable especially when the compliance of different rules is in contradiction, as the rulebook emphasizes the priority and hierarchical structure amongst these rules. In essence, it is defined as a set of rules wherein each rule maps the observed behavior to a real number, indicative of the degree of satisfaction or violation of the rule. The value of this formalism is demonstrated in the traffic rule exception scenarios, which works well to resolve the hierarchy and conflicting nature within the various rules.

Definition 4.2.1 (Rule)**.** Let $\mathbf{S}$ be a sequence of states. A rule $\psi : \mathbf{S} \rightarrow \mathbb{R}$ is a function that assigns to each state $s \in \mathbf{S}$ a real number $\psi(s)$. This value typically reflects the extent to which s complies with or violates the rule.

Definition 4.2.2 (Rulebook)**.** A rulebook R is defined as a tuple $(\Psi, \leq)$, where Ψ represents a finite set of rules and $\leq$ denotes a pre-order relation. Specifically, if $\psi \leq \psi'$, it implies that the rule ψ is of a lower priority compared to rule ψ'.

The rulebook's inherent hierarchical structure can be effectively visualized through graphical representation, wherein each rule within the rulebook is represented as a node on the graph. The edges connecting these nodes serve to indicate the priority relationships that exist between the respective rules. Specifically, an edge from one node to another would signify that the rule represented by the originating node is subordinate in priority to the rule represented by the terminating node. The example given in Figure 4.3a contains five rules. From the graph, it is clear that ψ_1 holds the highest priority. ψ_2 and ψ_3 are on the same level, as indicated by the two edges pointing to each other, which shows that they have equal priority. Next, ψ_4 is situated at the third level in the hierarchy. At the bottom, ψ_5 holds the least priority among the rules. The graph provides a straightforward representation of the hierarchical structure and the relative priorities of the rules.

In the rulebook employed in this study, the focus is on the hierarchical organization among various rules, as opposed to emphasizing the precedence relations among individual rules. Each level of the hierarchy can encompass one or more rules, with all the rules within a given level being treated as equivalent in terms of priority. This can be visually represented through a graph, as illustrated in Figure 4.3b. This structured format of the rulebook is sufficient for assessing rule conflicts in the majority of rule exception scenarios. For instance, a typical assessment criterion for vehicular behavior prioritizes safety over task completion, which in turn takes precedence over comfort, displaying a clear hierarchical order. The structured rulebook facilitates the incorporation of additional rules and the delineation of more hierarchical levels with relative ease, therefore, the scalability of the rulebook is guaranteed. Certainly, in a few complex scenarios, the hierarchical organization might not adequately capture the subtle relationships among various rules, necessitating a finer representation of relative priorities among individual rules. Censi et al. address this intricate situation by proposing advanced techniques such as priority refinement, rule aggregation, and rule augmentation[8] to achieve a more refined representation of the rulebook. Nonetheless, for the purpose of this thesis, the rulebook of hierarchical structure as depicted in Figure 4.3b is adopted as the rule formalization framework for traffic rule exception scenarios.



(a) Rulebook

(b) Hierarchical rulebook

Figure 4.3: The graph representation of rulebook

4.2.2 From LTL Rulebook to Reward

In this section, the process of consolidating a set of rules into a rulebook, which possesses a hierarchical structure, is explained. Furthermore, the section details the formulation of a reward function based on the rulebook, which could be used in reinforcement learning applications. The process can be systematically divided into four steps. Firstly, we need to articulate all the traffic rules relevant to specific rule exception scenarios and classify them into different hierarchies. Secondly, it is necessary to formalize each rule as an LTL formula. With this in place, these rules are then organized into a hierarchical rulebook based on the hierarchies that were established initially. Lastly, the construction of a reward function in alignment with the rulebook is carried out, which will serve as a vital component for reinforcement learning.

Suppose now there are five rules ψ_1 to ψ_5 in a specific traffic rule exception scenario. These rules are organized hierarchically and subsequently converted into distinct LTL formulas. For the first three rules, it is posited that the "Globally" operator, **G**, is employed, while the last two rules use the "Eventually" operator, **F**. The resulting formalized rules are listed in Table4.1. The symbols φ_i within the table denote adapted rules for LTL representation. For illustration, consider a traffic rule ψ_a stating "Vehicle should stay in the lane." This can be translated into the LTL formula $\mathbf{G}(\text{in\_lane})$. In this instance, φ_a here symbolizes the adapted rule (in_lane).

Notably, the semantics of LTL characterize the future, such as a particular condition that will eventually be satisfied. This implies that the evaluation of an LTL formula based on the state at a specific time instance might be contingent on future states. However, this characteristic does not apply to the design of reward functions in reinforcement learning algorithms. Taking inspiration

Rules	LTL Formula	Hierarchy
ψ_1	$\mathbf{G}\varphi_1$	1
ψ_2	$\mathbf{G}\varphi_2$	2
ψ_3	$\mathbf{G}\varphi_3$	2
ψ_4	$\mathbf{F}\varphi_4$	3
ψ_5	$\mathbf{F}\varphi_5$	3

Table 4.1: Formalized Rules in LTL

from Aguilar et al.[1], this thesis adopts the syntactic structure of LTL but the infinity-norm future-oriented semantics are not preserved. To more effectively tackle the challenges in the construction of reward functions for reinforcement learning, an intuitive semantic interpretation, denoted as f, is introduced.

As illustrated in Equation 4.1 and Equation 4.2, this thesis introduces intuitive functions to establish the reward function for distinct categories of rules, which correspond to different Linear Temporal Logic temporal operators. It is noteworthy that a substantial subset of traffic rules can be formulated using the *Globally* and *Eventually* temporal operators in LTL. In the context of the obstacle avoidance scenario examined in this thesis, all relevant rules can be formulated using expressions comprising these two operators. Consequently, this work specifically presents intuitive function designs tailored to these operators. For the *Globally* operator, denoted as **G**, the implication is that the vehicle is required to continually adhere to a given rule. As such, a penalty of -1 is assigned to states that break the rule, and a value of 0 is allocated in other instances. Conversely, for the *Eventually* operator, symbolized as **F**, the underlying rule should be rewarded once a particular condition is reached. Thus, a value of 1 is assigned when the rule is fulfilled, and 0 is assigned otherwise. It is worth mentioning that in the definition provided, $S_i \not\models \mathbf{G}\varphi$ and $S_i \models \mathbf{F}\varphi$ exhibit a subtle deviation from the conventional definition in LTL. Here, these expressions refer to a set of states in the currently generated trajectory that satisfy or do not satisfy the current rule. If there is a state in the trajectory that does not satisfy the current rule, the trajectory violates that rule. Additionally, depending on the specific scenario, different multiplication factors can be employed.

$$f(\mathrm{G}\varphi,\tau) = \begin{cases} -1, & \text{if } \tau \not\models \mathrm{G}\varphi \\ 0, & \text{Otherwise} \end{cases} \qquad [4.1]$$

$$f(\mathrm{F}\varphi,\tau) \;=\; \begin{cases} 1, & \text{if } \tau \models \mathrm{F}\varphi \\ 0, & \text{Otherwise} \end{cases} \qquad [4.2]$$

Subsequently, the hierarchical structure of the rulebook is integrated into the reward function design. For each hierarchy level, a hierarchy coefficient, ρ_j, is introduced as depicted in Figure 4.4, here j indicates the rulebook level. This coefficient is employed to lower of the reward or penalty associated with the rules at the given level. Crucially, this factor is engaged only when there is a conflict between rules at the current hierarchy and those at a higher hierarchy. Otherwise, the coefficient assumes a default value of 1 when confronted with rules at an equivalent or lower hierarchy. The detailed implementation is clarified in Equation 4.3.

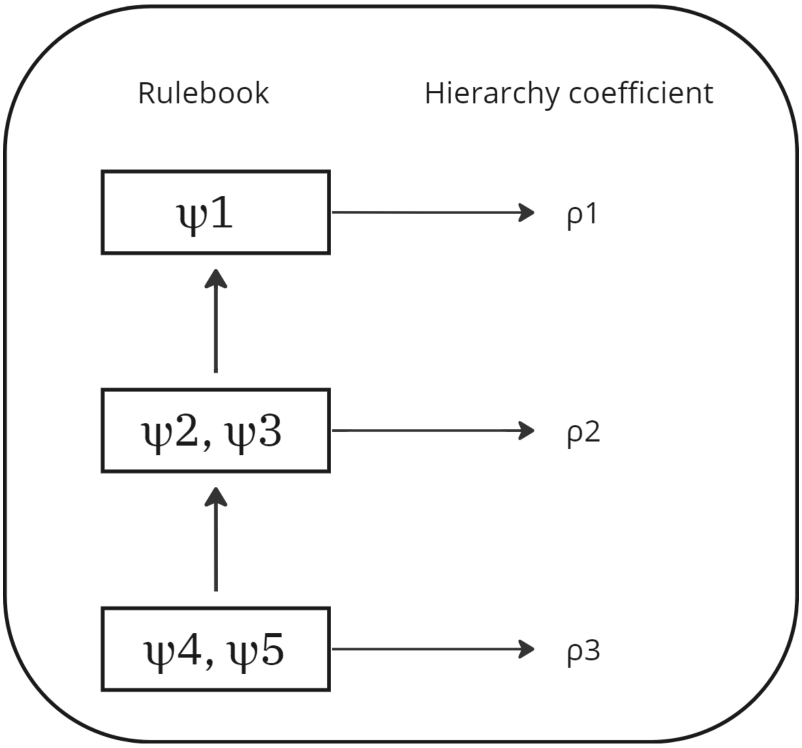


Figure 4.4: The rulebook and corresponding hierarchy coefficient

$$Reward(R,\tau) = \sum_{\psi \in \Psi} \left(\prod_{\psi' : \psi \leq \psi'} \rho_j \right) \cdot f(\psi,\tau) \cdot c_{\psi} \quad [4.3]$$

It is critical to underline that the hierarchy coefficient is activated solely in instances of rule conflicts, and only the coefficient corresponding to the rules in conflict is enabled; otherwise, the coefficients are set to 1. This guarantees that the rulebook is operational only during the necessary time frames. Nonetheless, the determination of when the rulebook is activated and deactivated is dependent on the specifics of the application scenarios.

4.3 Frenet Frame Trajectory Generation

Frenet frame trajectory generation is a comprehensive process. First, the trajectory's start state, denoted as $[x_1, x_2, \theta, \kappa, v, a]$, is determined. In cases where a prior trajectory has been calculated, this trajectory is evaluated at the prospective start state, a strategy known as low-level stabilization. Conversely, at system initialization or after reinitialization, the current vehicle position is employed, a method termed high-level stabilization.

Depending on the velocity, v, the algorithm selects a lateral mode. For higher velocities, a time-based mode, $d(t)$, is used, whereas for lower velocities an arc length-based mode, $d(s)$, is employed. The longitudinal start position, $s(0)$, is ascertained by projecting the start state onto the reference curve. Using Frenet transformation, the Frenet state vector is computed as:

$$[s, \dot{s}, \ddot{s}, d, \dot{d}, \ddot{d}]$$

Subsequently, lateral and longitudinal trajectories are generated in the Frenet space, taking into account their respective costs. Trajectories with high lateral accelerations are disregarded to optimize computational performance. The next phase involves combining these lateral and longitudinal trajectories. This is achieved by summing the partial costs of lateral and longitudinal components, represented as:

$$J(d(t), s(t)) = J_d(d(t)) + k_s \cdot J_s(s(t)) \qquad [4.4]$$

where k_s is a constant. Each longitudinal trajectory is combined with every lateral trajectory and converted back to world coordinates using the reference path. The algorithm verifies if the trajectories abide by physical driving constraints by evaluating curvature and acceleration. The trajectory sets are further scrutinized to check for any static or dynamic collisions, with a focus on trajectories with escalating total costs. The trajectory with the minimum cost that evades collisions is selected.

It is important to note that any trajectory λ created in the Frenet space is characterized by coordinates $[s(t), d(t)]$ and is uniquely defined by three terminal parameters: v_f, d_f, and t_f, as long as the vehicle itself is in a certain state. Building on this, these parameters can be used as the action space for a reinforcement learning agent. The agent is trained to derive a trajectory at each discrete step based on the current state.

Werling et al. [43] used an approach that employs varying terminal manifolds, denoted as $A\{v_f, d_f, t_f\}$, based on the vehicle's current state for generating a lattice of trajectories in Frenet space. The terminal manifolds are parameterized by:

- v_f: the target velocity, which dictates the desired speed at the end of the planned trajectory,
- d_f: the lateral displacement, representing the sideways movement relative to the reference path, and
- t_f: the arrival time, which is the time at which the vehicle is expected to reach the target state.

These parameters collectively define the goal state in Frenet space. The algorithm generates a large number of trajectories by varying these parameters and evaluates each trajectory based on a cost function, which is shown in Figure 4.5. The trajectory with the lowest cost is selected as the optimal trajectory. This process can be computationally intensive as it involves generating and evaluating a large number of trajectories.

In contrast, this thesis proposes an innovative approach that makes use of Deep Reinforcement Learning to efficiently generate trajectories. Instead of creating a superabundance of trajectories and conducting a linear search for the lowest cost trajectory, the terminal manifold $A\{v_f, d_f, t_f\}$ is employed as the action space in the DRL framework, which means that at each step, DRL agent can output a uniquely determined trajectory, like in Figure 4.6. Through training, the DRL agent learns to generate a trajectory directly. This trajectory is optimized over time, as the DRL agent receives feedback through a reward signal, which guides the agent in refining the trajectory generation process.

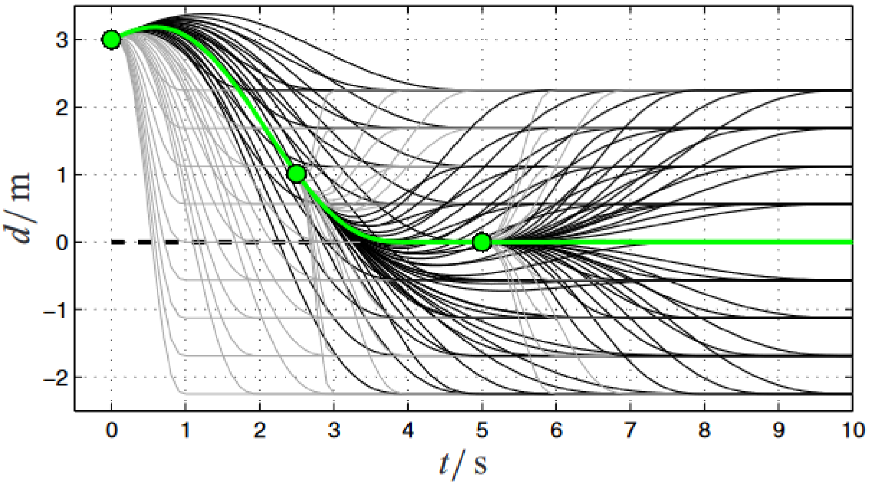


Figure 4.5: Use Frenet Planner to generate lattice and choose lowest cost trajectory[43]

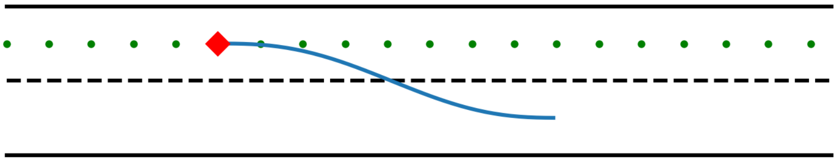


Figure 4.6: For each determined terminal manifold, DRL outputs a uniquely determined trajectory

4.4 DreamerV3

In light of the discussions in Chapter 3, the choice of utilizing DreamerV3 as the DRL model in this thesis is substantiated by several factors. Firstly, as one of the cutting-edge DRL models, DreamerV3 has demonstrated good performance across a diverse range of domains, indicating its robustness and versatility. Moreover, its capacity to train based on images is consistent with the experimental design of this thesis, where image data play a crucial role. Taking advantage of DreamerV3's inherent strengths in processing visual information can potentially lead to more subtle and effective motion planning by the autonomous vehicle.

In order to assess the efficacy of DreamerV3 within the CARLA environment, this thesis conducted a preliminary experiment. In this experiment, an end-to-end approach was employed, taking visual images from the camera sensor as inputs. The action space consisted of continuous steering angles and acceleration, serving as control commands for the vehicle's movement. The reward function was designed to impose penalties in instances of collision or low speed while granting rewards in other situations. The experiment was conducted within CARLA's TownOpt1 world. By contrasting the predictions DreamerV3 made concerning future states before and after training, and examining the vehicle's capability to follow the road, it was demonstrated that DreamerV3 can be effectively utilized within the CARLA environment. The results were promising, as DreamerV3 exhibited satisfactory performance within the CARLA environment. Figures 4.7 and Figure 4.8 clarify the evolution of DreamerV3 world model's capabilities in the CARLA environment from the inception of training to after 30,000 training iterations. In each figure, the first row displays the raw input procured from the RGB camera. The second row illustrates DreamerV3's internal simulation, or 'imagination,' which is constructed post-training, utilizing the World Model it has developed. and the third row illustrates a comparison between the camera input and DreamerV3's prediction after the data is processed through a decoder. It is evident that, with sufficient training, DreamerV3 successfully embodies the World Model of CARLA's environment and employs it to generate increasingly precise predictions.

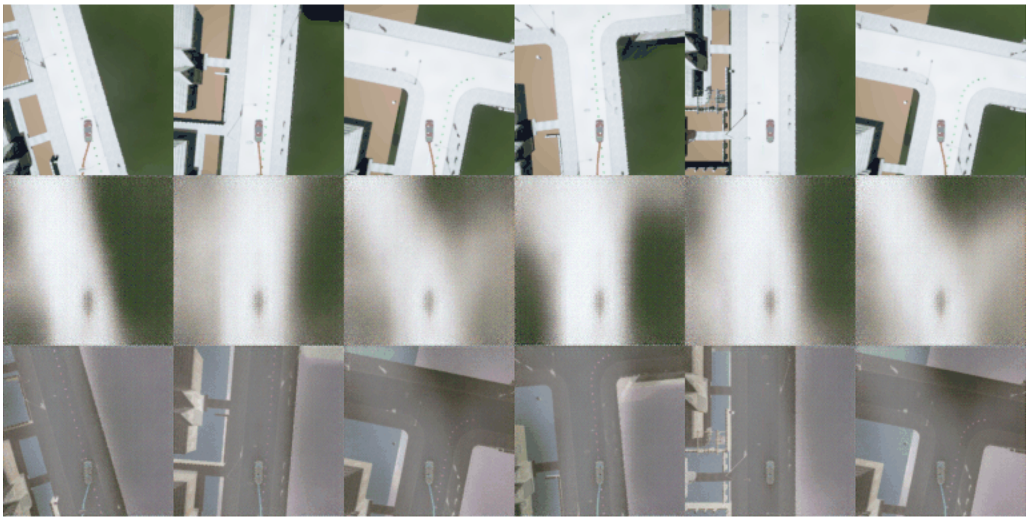


Figure 4.7: DreamerV3 prediction in CARLA environment at the train start. the first row the camera input, the second row is DreamerV3 prediction, the third one is a comparison between the above two after decoder.

Given DreamerV3's noteworthy performance in various domains and its limited need for hyperparameter adjustment, this thesis opts to use DreamerV3 as the reinforcement learning agent. In the proposed approach of this thesis, the above-mentioned Rulebook and the trajectory generator can be integrated with DreamerV3 to create a sophisticated DRL agent for autonomous driving. The Rulebook, which encompasses the formalized set of traffic rules and priorities, serves as an evaluative component for the generated trajectories. As DreamerV3 processes the images from the BEV camera, it interacts with the trajectory generator, which generates one trajectory based on the current state and the action given by DreamerV3. The rulebook is then invoked to evaluate whether this trajectory complies with traffic rules and regulations. DreamerV3, equipped with its capacity to learn and make predictions, receives feedback from the Rulebook in the form of rewards or penalties. This feedback is critical in training DreamerV3 to understand and prioritize trajectories that adhere to the rules and ensure safe navigation. Throughout this process, Dream-

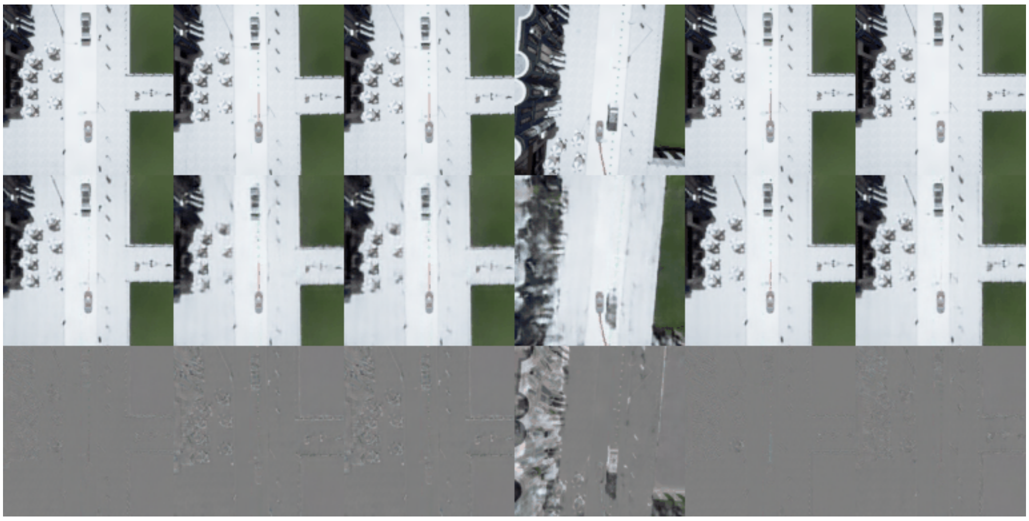


Figure 4.8: Dreamer prediction in CARLA after 30000 training iterations

erV3 optimizes its policy to generate increasingly safe and compliant trajectories over time. In summary, incorporating images as input for the reinforcement learning agent, a carefully designed Rulebook based reward function and action space will be employed. This design ensures that, at each step, the reinforcement learning agent generates a trajectory. Subsequently, the vehicle will be governed by a fundamental PID controller to accomplish trajectory adherence. It's important to note that, within this framework, the vehicle model will be simplified and represented as a basic bicycle model[33].

5 Experimental Setup

In the context of autonomous vehicles, it is essential to have a rigorous experimental setup to validate the effectiveness of algorithms and techniques proposed for handling anomalous traffic scenarios. This chapter is dedicated to describing the framework and parameters that were used in the experiments. The logical progression of this chapter aims to first set the stage by explaining the simulation environment and training scenarios where the experiments were conducted. Understanding the environment is crucial as it serves as the foundation upon which the experiments are built. Subsequently, this chapter delves into the training process which is the heart of the experimentation. Here, the various aspects of how the DRL model was trained are explained, including the action and state spaces, the reward function, and the training parameters. These elements are critical in understanding how the DRL model learns and adapts to perform the required tasks.

5.1 Simulation Environment

5.1.1 CARLA Environment

CARLA (Car Learning to Act) is an open-source simulation platform that is widely used in the research and development of autonomous driving systems. It is designed to simulate real-world driving scenarios in a highly customizable environment, enabling researchers and developers to test and validate autonomous driving algorithms under varied conditions. CARLA provides high-fidelity sensor models and realistic vehicle dynamics [9], making it a valuable tool for conducting experiments in a controlled setting before deploying autonomous vehicles in real-world scenarios. The platform supports integration with various programming languages and machine learning libraries, which facilitates the development of sophisticated autonomous driving systems. CARLA's versatility and realism make it an essential asset for researchers and practitioners in the field of autonomous driving.

In the conducted experiment, CARLA 0.9.13 serves as the simulation platform, specifically tailored for emulating an urban traffic environment. Two sensors, namely RGB camera and Collision Detector, are integrated into the ego vehicle. The RGB camera plays a crucial role in supplying the RL agent with observation inputs, which are essential for perception and decision-making. Concurrently, the Collision Detector is employed to ascertain instances of vehicular collisions.

5.1.2 Training Scenarios

In the previous research carried out by colleagues at FZI, a comprehensive database comprising a set of anomaly scenarios has been constructed. These scenarios are specifically designed to simulate traffic rule anomalies under various environmental conditions and encompass diverse anomalies. Based on the existing dataset, the primary focus of this thesis is invested in this kind of scenario in which a moving vehicle confronts an unforeseen obstacle in its path, compelling it to execute an evasive maneuver to prevent collision. In these contexts, the database is exceptionally rich, consisting of 1000 distinct anomaly scenarios. These are carefully crafted within the TownOpt1 world in CARLA, with Tesla Model 3 serving as the ego vehicle. These scenarios contain important elements such as the ego vehicle's spawning points, the types of anomalies, weather conditions, and, crucially, reference trajectories which play a pivotal role by acting as the s-axis in the Frenet framework and guiding the vehicle's motion. Setting up these scenarios entails a procedural approach where the initial point is arbitrarily selected on the road map in CARLA's world, and it is ensured that the lanes are meant for driving (carla.LaneType: Driving). The termination point is consistently placed 80 meters ahead in the direction of travel. Throughout this trajectory, anomalies are spawned randomly from the carla.BlueprintLibrary, which includes static objects and vehicles, while ensuring a minimum distance of 5 meters from both the initial and terminal points. This methodical setup yields a diverse range of scenarios that are instrumental in training and evaluating the performance of DRL-based autonomous driving systems in terms of motion planning under anomaly traffic rule exception scenarios within the experimental environment of this thesis.

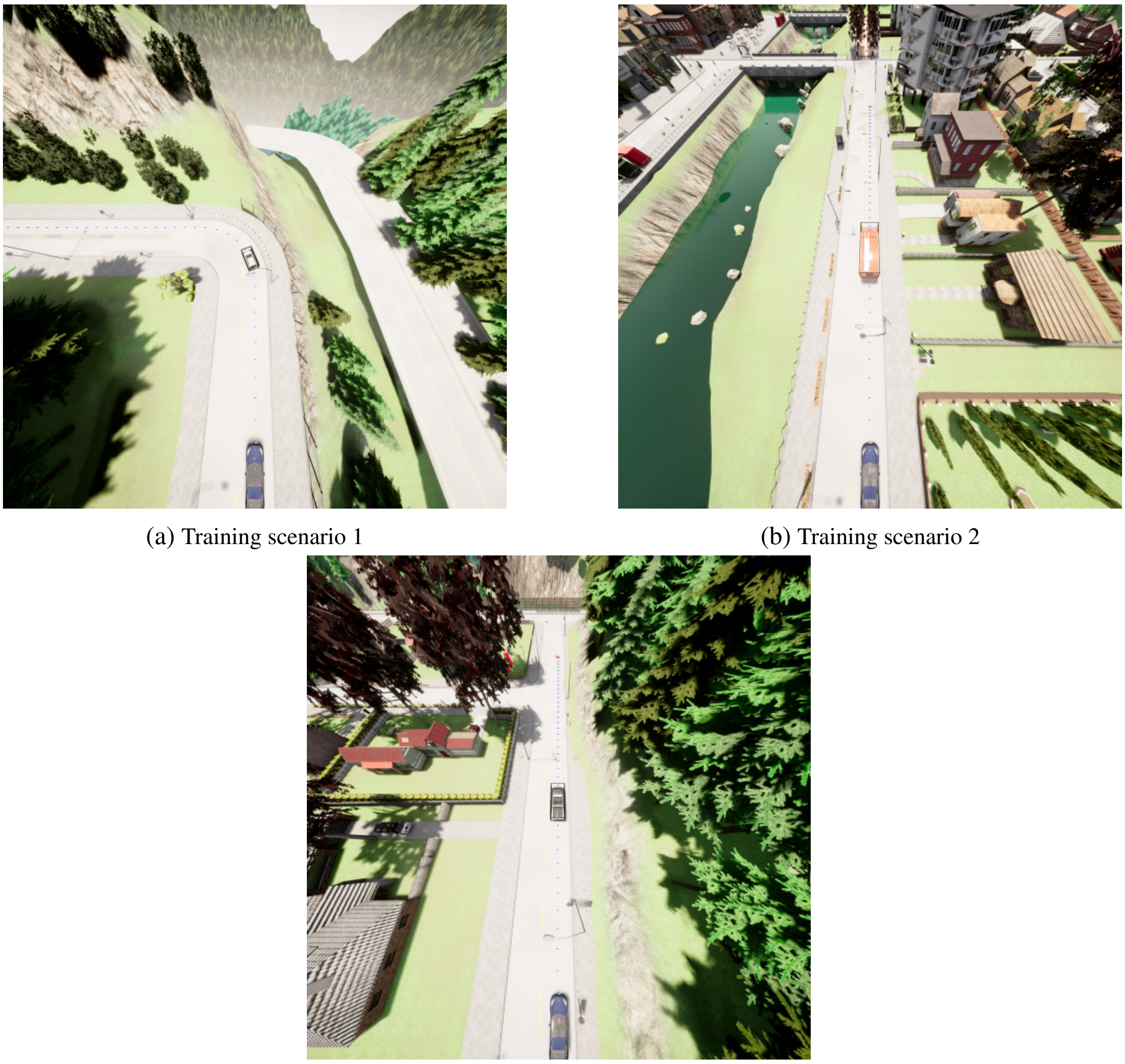
Some of the designed scenarios are shown in Figure 6.7. In the figure, a dashed line is noticeable, representing the reference path that the autonomous vehicle aims to follow. During the actual experimentation, the Foliage layer within the CARLA map was removed to ensure an open field of view for the camera sensor, thereby enhancing its perception capabilities.

In subsequent experiments, within each test scenario, the vehicle's primary objective will be set through the reward function to bypass obstacles and then return to the previous lane, ultimately succeeding in reaching the predetermined destination. The path traversed by the vehicle in the process of achieving this goal will serve as an important metric for evaluating the degree of successful accomplishment of the objective.

5.2 Training Process

The training of the vehicle begins in a simple environment withoud any anomalies, with a focus on training the ego vehicle to adhere to the urban roads on the same map, and this anomaly-free training will persist for 3,000 steps until the vehicle is able to effectively follow the urban roads. Subsequently, the training advances to scenarios involving anomalies, specifically the anomaly traffic rule exception scenarios that are the subject of this thesis. Such a training process aligns with an intuitive driving experience, in which a vehicle is capable of navigating according to traffic rules and, when encountering anomalies, gradually learns through training to execute appropriate actions under these circumstances.

At each step, the reinforcement learning agent provides a trajectory in Frenet space for the vehicle. A PID controller then takes over, guiding the vehicle in tracking the given trajectory at a relatively steady speed. This continues until a collision occurs, the vehicle reaches its target, or the maximum number of steps is reached.



(a) Training scenario 1

(b) Training scenario 2

(c) Training scenario 3

Figure 5.1: Training scenarios

Expanding on this, it is crucial to recognize that by initiating the training in a simplified setting devoid of anomalies, the model focuses on grasping the fundamental aspects of navigating through urban roads. This foundation is essential before introducing the complexity of anomalous situations, as it mirrors the natural progression in human driving experience where the basics are learned before dealing with complex scenarios. Once the model is adept in normal traffic conditions, introducing traffic rule exceptions permits the reinforcement learning agent to adapt and learn how to deal with these exceptions efficiently. This approach is more structured and may lead to a more robust autonomous driving system capable of handling a wider range of driving scenarios. Furthermore, the combination of the trajectories generated in Frenet space and the PID controller serves to ensure that the vehicle follows these trajectories accurately, which is key to successful navigation especially in environments with unexpected obstacles.

In the following sections, the action space, state space, reward function, and training parameters utilized by DRL agent in this experiment will be introduced and elaborated upon one by one. It is imperative to have a clear understanding of these components as they are pivotal in the training and performance of the DRL agent.

5.2.1 Action Space

As previously mentioned, in Frenet space, trajectories for the ego vehicle at any given state can be obtained by considering three primary terminal manifold parameters: velocity (v_f), final lateral position (d_f), and time to reach the target (t_f). In the specific approach studied in this thesis, d_f is crucial for avoiding obstacles, while v_f and t_f primarily influence the curvature and length of the trajectory. It is noteworthy that v_f and t_f have a minimal effect on the experimental outcomes when a PID controller is employed to track the trajectory using constant speed as smoothly as possible. Consequently, in the experiments, the values are set as $v_f = 10m/s$ and $t_f = 4s$, which were determined empirically to be reasonable figures for the scenarios and reference path lengths used in the experiments. This implies that the resulting trajectory is one where the vehicle aims to reach a lateral position d_f in 4 seconds at a target velocity of 10 meters per second.

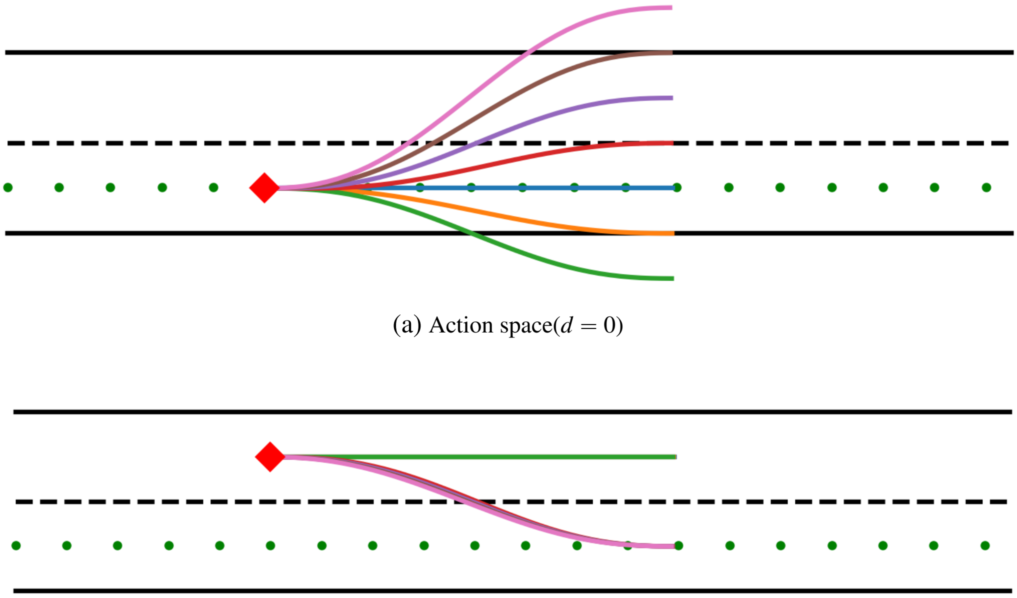
By keeping these two parameters constant, the action space is solely associated with d_f. The action space is defined as $Discrete(7)$, encompassing integers ranging from 0 to 6. Taking the practical driving environment into account, there is a significant distinction between the vehicle's state and feasible space when the vehicle maintains its lane ($d = 0$) and deviates from its lane ($d \neq 0$). Thus, the action space is correlated with different states of d_f. As depicted in Figure 5.2, when the vehicle is within its lane ($d = 0$), the seven actions from 0 to 6 correspond to seven distinct values of d_f, where the difference between adjacent d_f values is 1.75, which is equivalent to half the lane width in the test scenarios. Conversely, when the vehicle departs from its lane, actions 0 to 3 correspond to one value of d_f, while actions 4 to 6 correspond to another. These values represent, respectively, maintaining the current level of deviation and steering the vehicle back to the centerline of the lane. The specific correspondence between actions and d_f can be observed in Table 5.1.

action	0	1	2	3	4	5	6
$d = 0$	-3.5	-1.75	0	1.75	3.5	5.25	7
$d \neq 0$	d	d	d	d	0	0	0

Table 5.1: d_f value corresponding to action value

5.2.2 State Space

The state space in autonomous driving is vital as it conveys information about the current situation of the vehicle and its surrounding environment, which is instrumental for making informed decisions and taking appropriate actions. In this context, the state space primarily consists of image data obtained from an BEV RGB camera sensor, as depicted in Figure 5.3. This sensor is mounted on the vehicle in a fixed position, and its spatial configuration is represented as $x = 10$, $y = 0$, and $z = 15$ in Cartesian coordinates. Furthermore, the orientation of the sensor relative to the vehicle is characterized by pitch, yaw, and roll angles, which are set at 90, 0, and 0 degrees,



(a) Action space($d = 0$)

(b) Action space($d \neq 0$)

Figure 5.2: Action space

respectively. This specific setup ensures that the camera captures a clear and unobstructed view of the road ahead, which is critical for detecting and analyzing different elements within the driving environment.

The images captured by the RGB camera are of 128x128 pixel resolution, a size that strikes a balance between providing sufficient detail and computational efficiency. The state space incorporates several critical aspects of the driving environment, such as the road geometry, the position and movement of the ego vehicle, the presence of any anomalies or obstacles, and the reference path that the vehicle is intended to follow. The inclusion of these elements in the state space is crucial for understanding the dynamics of the environment and making predictions about future states, which is central to planning and decision-making in autonomous driving.

In the context of the Dreamer architecture, as shown in Figures 4.7 and 4.8, the image data captured by the RGB camera sensor is processed through a neural network that learns a world model. This world model is essentially an internal representation of the environment and is used by the Dreamer algorithm to predict future states based on current observations and actions. The world model aids in generating simulated experiences, enabling the Dreamer algorithm to learn an effective policy for motion planning and control by forecasting the consequences of different actions.

5.2.3 Reward Function

The design of the reward function contains two parts 5.1, one part is the reward function r_{ego} obtained based on the state quantity of ego vehicle, and the other part is the reward and punishment

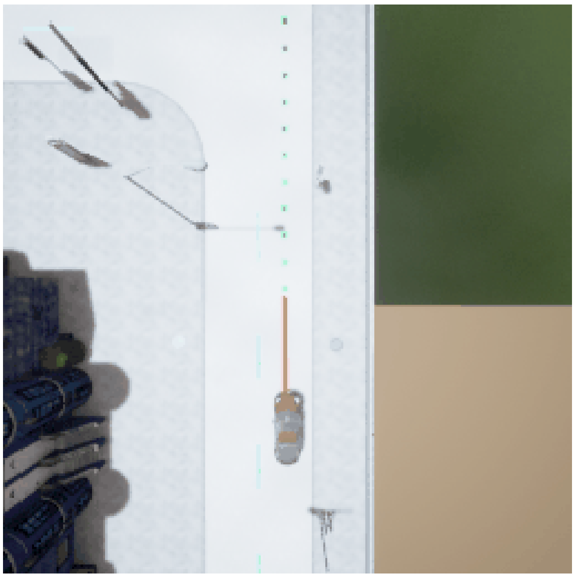


Figure 5.3: State space

$r_{rulebook}$ obtained based on Rulebook.

$$r = r_{ego} + r_{rulebook} \tag{5.1}$$

A total of three rules are involved in the traffic rule exception scenario studied in this thesis as follows:

- ψ_1: avoid collision
- ψ_2: the vehicle should keep its lane
- ψ_3: the vehicle shall not drive off the road

Using LTL logic semantics they can be expressed as

$$\psi_1 : \mathbf{G}(\text{no\_collision}) \tag{5.2}$$

$$\psi_2 : \mathbf{G}(\text{in\_lane}) \tag{5.3}$$

$$\psi_3 : \mathbf{G}(\text{no\_out\_road}) \tag{5.4}$$

Primarily from the point of view of safety and driving experience, we develop such a hierarchical relationship for these three rules and assign the corresponding hierarchical coefficients to each, as shown in Figure 5.4. From a safety and vehicle navigation perspective, keeping the vehicle on the road should generally take precedence over staying in the lane. However, the interrelation between the two rules is such that the violation of the ψ_3 inevitably implies the violation of the ψ_2. In essence, when the vehicle departs from the road, it concurrently deviates from its lane. Consequently, this relationship results in a violation of ψ_3 receiving a greater penalty than a violation of ψ_2, which to some extent offsets the effect of the equivalence hierarchy of the two rules. The selected rules, their LTL Formulas, and the corresponding Rulebook hierarchy are shown in detail in Table 5.2.

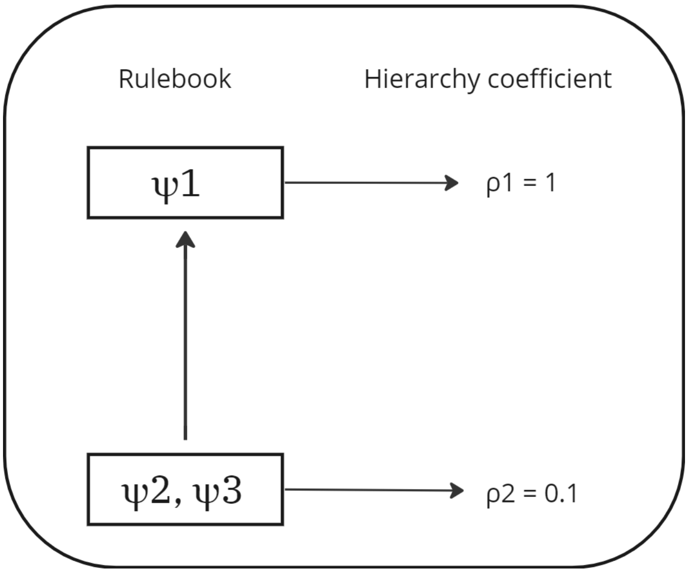


Figure 5.4: Rulebook for training scenario

Traffic Rules	LTL ID	LTL Formula	Hierachy
Avoid collision	ψ_1	G(no_collision)	1
The vehicle should keep its lane	ψ_2	G(in_lane)	2
The vehicle shall not drive off the road	ψ_3	G(no_out_road)	2

Table 5.2: Selection and formalization of traffic rules. Based on the traffic environment of the anomalies, these three traffic rules were selected for the rulebook

r_{ego} can be expressed as the following equation:

$$r_{ego} = r_{finish} + r_{speed} * l \qquad [5.5]$$

When the position of the vehicle is evaluated in Frenet space and s has reached s_{target} but d has not reached d_{target}, $r_{finish} = 10$, and if both have reached the target, $r_{finish} = 60$, otherwise, $r_{finish} = 0$. If the vehicle goes faster than $50km/h$ or slower than $10km/h$, $r_{speed} = -1$, otherwise 0, l is the length of the trajectory traveled in the past one step.

According to 5.1, $r_{rulebook}$ could be expressed as:

$$\begin{aligned} r_{rulebook} &= \sum_{\psi \in \Psi} \left(\prod_{\psi' : \psi \preceq \psi'} \rho_j \right) \cdot f(\psi, \tau) \cdot c_{\psi} \\ &= \rho_1 * r_{collision} * c_{col} + \rho_2 * r_{in\_lane} * l * current\_d + \rho_2 * r_{no\_out\_road} * l \end{aligned} \qquad [5.6]$$

According to 4.1 and 4.2, $r_{collision}$ is set to -1 if a collision occurs and 0 for the rest of the time. $r_{in\_lane} = -1$ if the vehicle deviates from its lane, $r_{no\_out\_road} = -1$ if the vehicle leave the road, otherwise, $r_{in\_lane} = r_{no\_out\_road} = 0$. It should be emphasized that ρ_2 is equal to 0.1 only if the rulebook is activated, and 1 the rest of the time. In the rule exception scenario studied in this thesis, the range of rulebook activation is set to when the end or start of the vehicle's current trajectory lies within $10m$ of the front and rear of the anomaly location, and this range is judged through the s value in the Frenet space. The design of the reward is based on the common strategy of reward design in DRL applications of self-driving cars[23] and adjusted after several experiments.

5.2.4 Training Parameters

We trained the vehicle in the Anomaly scenarios mentioned earlier by combining Frenet space trajectory generation, Rulebook, and DreamerV3. Among them, 800 scenarios were used for training and the remaining 200 scenarios were used for evaluation. The vehicle is equipped with an RGB camera and a collision detector, which are used to obtain observations and collision detection respectively. The evaluation is conducted at fixed intervals of 400 steps, with each evaluation including 20 episodes. Considering the design of the experimental setup in this paper, a method that accumulates multiple episodes and averages the metrics can more reasonably illustrate the training effect and performance. Table 5.3 shows the basic training parameters. The experiments are divided into two groups: Rulebook activated within a specified range and Rulebook not activated throughout the process. Activation within a specified range means that traffic rule exceptions are allowed for the ego vehicle within this range, meaning it can temporarily break rules without or with only little penalty. Not activated means that the hierarchical structure between rules is not considered.

In addition, we have also set up four control group experiments, in which all four experiments use discrete control commands ranging from -1 to 1 with intervals of 0.5 as the output of the DRL agent, corresponding to the values of steering and acceleration. Among them, two groups use the baseline DRL model Rainbow, and the other two groups also use DreamerV3 as the DRL model. The two experiments corresponding to each model include both situations where Rulebook is activated and not activated. The rest of the experimental setup is the same as the main experiment in this thesis.

Parameters	Value	Parameters	Value
steps	$4e4$	device	cuda:0
action repeat	1	obs_size	[128,128]
eval_every	400	batch_size	32
eval_episode_num	20	batch_length	64
actor_dist	onehot	discount	0.997

Table 5.3: Training parameters

6 Evaluation

In this section, the results of the experiments will be compared and analyzed. The training process and results of the method we used with baseline are compared in this section, and the results show that when dealing with anomaly traffic rule exception scenario, compared with using control command as the output of DRL agent, using DRL agent to generate a real-time Frenet trajectory and follow it with the controller has better learning efficiency and performance than using the control command as the output of RL agent, both for Baseline model and DreamerV3. In addition, using the rulebook as part of the reward function in the training makes the training faster and the final vehicle performs better in the anomaly traffic rule exception scenarios. Finally, the trajectories of the vehicles in each evaluation episode are tracked, and whether and to what extent the vehicles break a rule during the process is recorded. By comparing and analyzing the trajectory status and rule compliance of the vehicle in different training phases, it is proved that the vehicles can cope with the anomaly rule exception scenario well through training.

6.1 Evaluation Metrics

The selection of adequate evaluation indicators is important for the analysis of RL experimental results. In the scenario studied in this thesis, we want to evaluate the vehicle's performance in response to the anomaly traffic rule exception scenario. Good performance means that the vehicle needs to deviate from the lane at the right position to avoid the obstacle and, after completing the obstacle avoidance behavior, return to the original lane in the shortest possible time. On the one hand, when a vehicle cannot avoid an obstacle correctly, a collision will occur and thus it will not reach the target, so successfully completing the obstacle avoidance behavior will allow the vehicle to travel a longer road length. On the other hand, in the training scenario we set, a reference route is given in addition to the road environment and obstacles, and successfully reaching the end of the route can be considered as completing the obstacle avoidance task, so the degree of task completion can also be used to evaluate the training effect. In addition, the degree of compliance with traffic rules when avoiding obstacles is focused on in this thesis, which can be used to evaluate the role of the rulebook and the degree and duration of traffic rules being broken. Based on the above analysis, the following three metrics or plots were chosen to evaluate the training effects.

- *arrived_s*: The value of the vehicle on the s-axis in the Frenet coordinate at the end of each episode, i.e., the distance the vehicle traveled along the lane for each episode. It can also be interpreted as the length of the vehicle's trajectory projected onto the reference path.
- $finish\_score$: If the vehicle successfully reaches the end position at the end of the episode, the value is 1. If the vehicle reaches the end distance along the road longitudinally, but the

lateral distance is not reached, that is, it does not successfully return to the original lane, the value is 0.5, and the remaining case is 0, that is, the vehicle has not completed the obstacle avoidance task.

- *rule graph*: In each episode, the vehicle trajectory will be recorded, while the compliance of each rule of the whole process will also be monitored and presented in the form of a graph, as in Figure 6.6.

6.2 Effectiveness Evaluation

In this section, we evaluate the efficiency of different deep reinforcement learning (DRL) approaches in autonomous vehicle navigation. Efficiency, in this context, is interpreted as the vehicle's ability to effectively travel the required distance along the desired lane and successfully complete its journey across different scenarios.

6.2.1 Trajectory Generation Effectiveness

As depicted in Figure 6.1, the parameter $arrived_s$ represents the distance that the vehicle traveled along the lane in each episode, or more specifically, the value of the vehicle on the s-axis in the Frenet coordinate at the end of each episode. It is also interpretable as the length of the vehicle's trajectory projected onto the reference path.

The learning curve of the *arrived_s* parameter reflects the learning progression of the DRL model in optimizing the vehicle's ability to travel along the desired path. A higher value of *arrived_s* and a faster increase in the learning curve signify better performance.

From Figure 6.1, it is evident that the experiments utilizing DreamerV3 in conjunction with trajectory planning outperformed the other groups. The combination of DreamerV3 and trajectory planning led to a steeper learning curve, indicating that this approach quickly adapted to efficiently guide the vehicle along the desired path. On the other hand, the groups that relied on control commands as outputs, regardless of whether they used Rainbow or DreamerV3 as the DRL model, exhibited slower learning and poorer performance in this metric.

The $finish\_score$ parameter, displayed in Figure 6.2, is indicative of the success rate of the vehicle in completing its journey. A score of 1 represents that the vehicle successfully reached the end position. A score of 0.5 signifies that the vehicle managed to travel the required distance longitudinally but failed to successfully return to the original lane, and a score of 0 indicates that the vehicle did not complete the obstacle avoidance task.

An increasing trend in the $finish\_score$ learning curve and higher values indicate an improved ability of the vehicle to successfully navigate and complete its path.

Figure 6.2 shows that analogous to the *arrived_s* evaluation, the combination of DreamerV3 with trajectory planning resulted in a steeper learning curve and higher final $finish\_score$ values. In contrast, the experiments based on control commands remained stagnant close to 0, indicating a lack of successful completions.

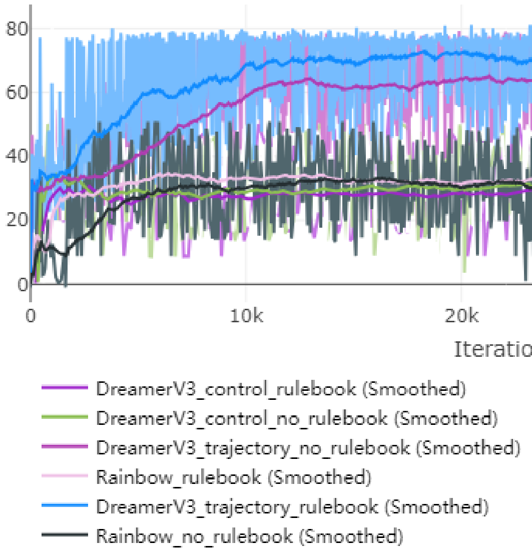


Figure 6.1: The *arrived_s* curves of six methods during the training process. *arrived_s* represents the farthest distance *s* reached by the vehicle in Frenet space in each episode. Three models - Rainbow control command, Dreamer control command, and Dreamer trajectory - are compared. For each model, there are two variations: one that includes Rulebook and another that does not include Rulebook.

The results of the efficiency evaluation suggest that the integration of trajectory generation with the DRL model significantly enhances the training efficiency of autonomous vehicles in traffic rule exception scenarios involving anomalies. The combination ensures not only that the vehicle travels efficiently along the desired lane but also completes its path more successfully compared to using control commands as outputs.

6.2.2 Rulebook Effectiveness

This section delves into the impact of integrating a hierarchically structured Rulebook into the reward function design when using DreamerV3 with trajectory generation. This is crucial for understanding how domain-specific knowledge can enhance the training of DRL models, particularly in complex or anomaly scenarios.

Figure 6.3 illustrates a side-by-side comparison of the learning curves representing the *arrived_s* parameter for the two experiments: DreamerV3 combined with trajectory generation with Rulebook activation and without Rulebook activation. From the figure, it is evident that the curve representing the experiments with Rulebook activation exhibits a steeper gradient and higher values in the later stages compared to the curve representing the experiments without Rulebook activation. This suggests that the vehicle was able to travel longer distances along the desired path in each episode when the Rulebook was integrated.

The enhancement in performance with Rulebook integration suggests that the Rulebook likely

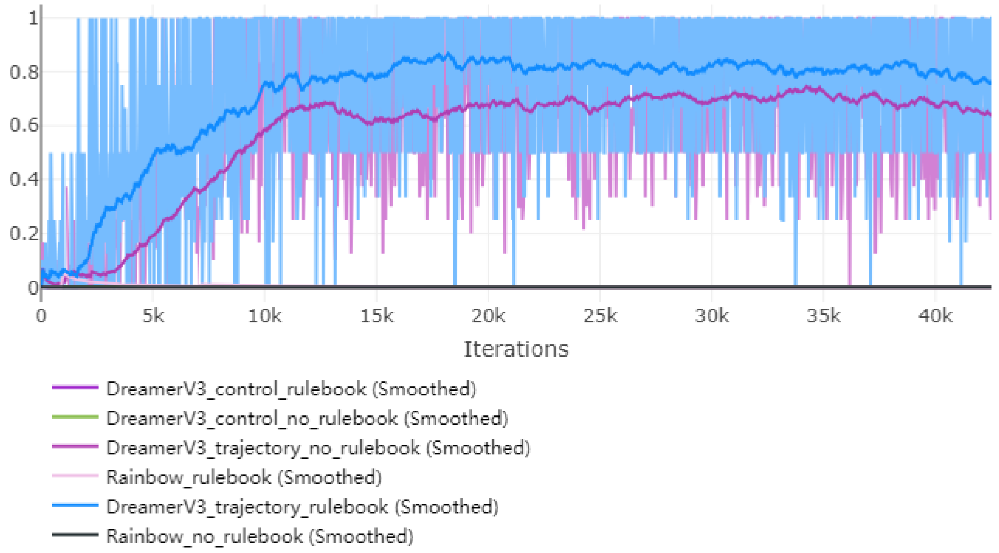


Figure 6.2: The *finish_score* curves of the same six methods during the training process. *finish_score* represents the extent to which the vehicle completes the task in each episode, with a score of 1 for successfully avoiding obstacles and reaching the destination, 0.5 for successfully avoiding obstacles but not reaching the destination, and 0 for not avoiding obstacles.

offers an additional layer of guidance, potentially reflecting real-world driving regulations and best practices. This likely enables the DRL model to quickly learn more effective navigation strategies, which, in turn, contributes to improved path-following efficiency. In anomaly scenarios, where unexpected or non-standard traffic situations may arise, this becomes even more critical as it guides the autonomous vehicle in adhering to structured rules, which can lead to safer and more predictable driving behavior.

Figure 6.4 provides a similar comparative analysis as Figure 6.3 but focuses on the *finish_score* parameter, which reflects the success rate of the vehicle in completing its journey. Similarly, Figure 6.4 shows that the experiments with Rulebook activation perform better. The learning curve with Rulebook activation is steeper and maintains higher values, indicating that the autonomous vehicle is more successful in completing its journeys as intended.

This improvement in $finish_score$ with Rulebook activation implies that not only does the Rulebook assist in guiding the vehicle along the desired path, but it also ensures that the journeys are completed more successfully and likely in a safer manner. This can be particularly vital in ensuring that the autonomous vehicle is capable of handling complex driving scenarios, especially those that might involve complex maneuvers, by adhering to a structured set of rules and guidelines.

In summary, the analysis of Figures 6.3 and 6.4 underscores the significance of integrating domain-specific knowledge, such as traffic regulations encapsulated in a Rulebook, into the reward function design for training DRL models in autonomous driving. The incorporation of a Rulebook not only enhances the ability of the DRL model to guide the autonomous vehicle more

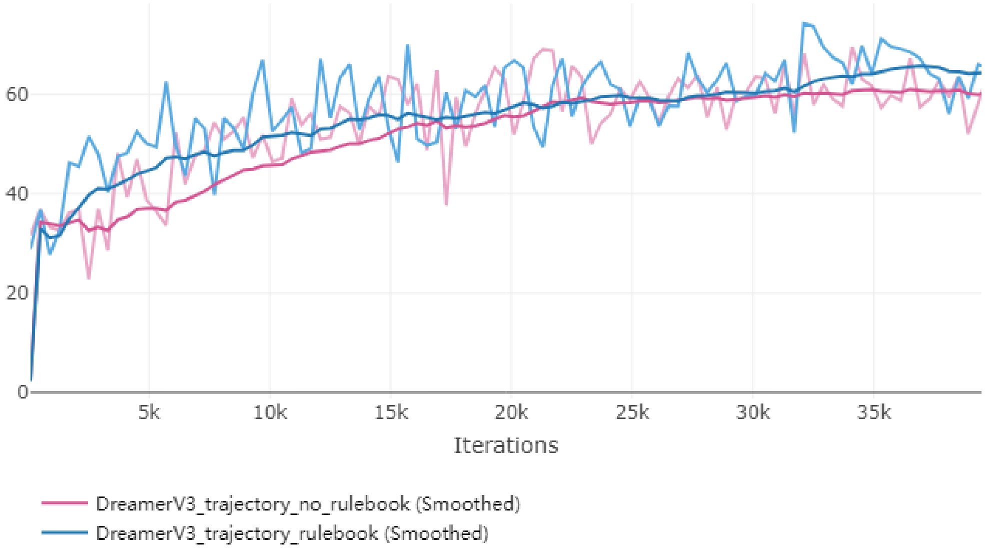


Figure 6.3: The *eval_arrived_s* curves of DreamerV3 with and without rulebook during the training process. *eval_arrived_s* is the average of the *arrives_s* obtained for every 20 evaluations.

efficiently along the desired path but also ensures a higher rate of successful and likely safer journey completions. This is particularly advantageous in anomaly traffic rule exception scenarios, where adherence to traffic rules and intelligent decision-making are critical.

6.3 Trajectory Performance Evaluation and Adherence to Traffic Rules

Although the method proposed in this thesis, which combines trajectory generation and Rulebook, performs well in terms of learning curve speed and final values, surpassing the methods based on control command output and those without Rulebook integration into the reward function, the actual performance of the vehicle in handling anomaly scenarios still requires further analysis and research. In the following analysis, the vehicle trajectory and degree of compliance with traffic rules achieved by the vehicle's motion will be examined by studying the driving trajectory of the vehicle in a certain episode in the evaluation, as well as the frequency, stages, and extent to which the vehicle breaks traffic rules during this process.

6.3.1 Successful Case Evaluation

After 35,000 training steps, as can be seen from Figures 6.3 and 6.4, the DRL agent has essentially entered a stable state, where the learning curve is no longer growing. It is reasonable to select representative cases from the evaluation episodes at this stage for vehicle trajectory performance analysis.

Figure 6.5 shows the vehicle's driving trajectory and the status at discrete moments in a complete

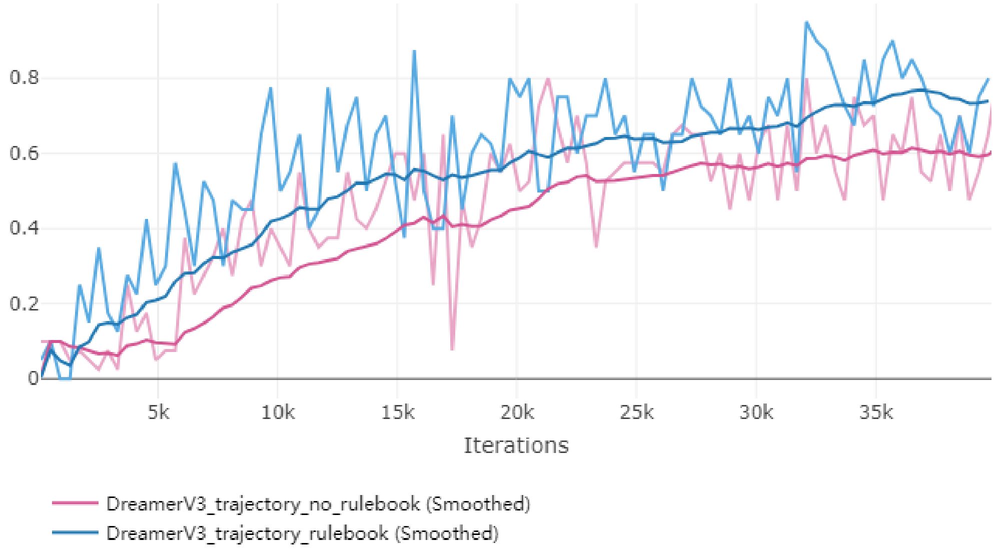


Figure 6.4: The *eval_finish_score* curves of DreamerV3 with and without rulebook during the training process. *eval_finish_score* is the average of the *finish_score* obtained for every 20 evaluations.

episode during this stage, where all images are raw images from the BEV camera. Observing this figure, it can be noticed that the vehicle changes lanes to avoid obstacles in advance before reaching them, and chooses to return to the original lane after passing the obstacles, ultimately successfully reaching the endpoint. Figure 6.6 shows the corresponding trajectory and traffic rule compliance diagram for this process. In this figure, the vehicle's trajectory is converted and plotted in Frenet coordinates so that regardless of whether the actual road serving as the reference path is a straight line or a curve, it appears as a straight line in the schematic, allowing for more intuitive observation of the vehicle's trajectory relative to the reference path. From the figure, it can be seen that the vehicle completes the obstacle avoidance behavior within the range where the Rulebook is activated and returns to the original lane. During this process, the portion of the "lane keep" rule that is violated is considered as traffic rule exceptions allowed by the Rulebook. It's worth noting that the definition of the Rulebook activation range targets the end of the trajectory that is closer to the anomaly. As shown in this figure, the beginning of the lane-changing trajectory is not within the Rulebook activation range, but the end of the trajectory is within the Rulebook activation range, and is thus considered as behavior that meets the requirements of the Rulebook. Combined with the above analysis, the episode corresponding to Figures 6.5 and 6.6 achieved a vehicle motion trajectory that is entirely in line with expectations. According to Figure 6.4, it is known that more than seventy percent of the episodes that successfully completed the task reached this in the evaluation episodes after sufficient training. This proves the effectiveness and good task completion rate of the method proposed in this thesis in anomaly scenarios.

Next, a more extensive showcase of successful scenarios is provided, as illustrated in Figure 6.7.

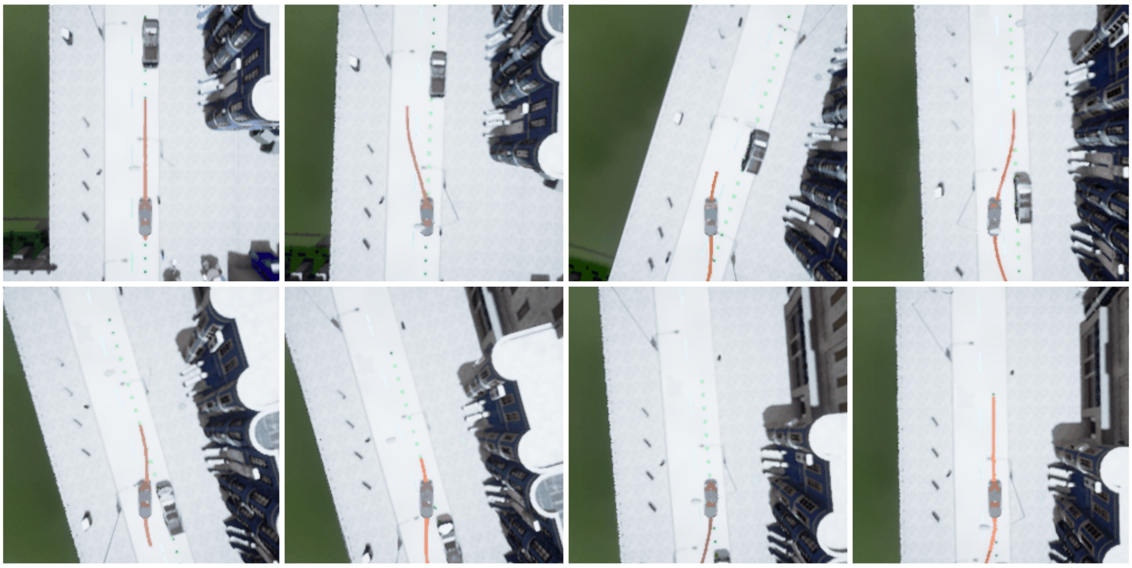


Figure 6.5: The camera view of the vehicle successfully completing the task on a straight road segment after 40,000 training steps.

In this figure, three distinct vehicle trajectories that have accomplished their goals successfully are depicted, highlighting the versatility and adaptive capabilities of the DRL method proposed in this thesis.

In Figure 6.7a, a particularly challenging scenario is presented. The vehicle is faced with an obstacle positioned early in its path. Despite the limited time and space for maneuvering, the vehicle adeptly avoids the obstacle, realigns itself with the lane, and proceeds to successfully reach its destination. This demonstrates the capacity of the proposed method to efficiently handle scenarios where swift reactions are paramount.

Figure 6.7b showcases a scenario where the anomaly is situated relatively close to the side of the road. Notably, the vehicle maneuvers around the obstacle and completes the task without infringing upon any traffic rules. This indicates the method's finesse in executing precise movements and adhering to traffic regulations. The integration of the Rulebook into the reward function becomes particularly significant here. While streamlining the training process, the Rulebook also ensures that the vehicle's trajectory respects the safety constraints and traffic rules, which is crucial for real-world implementation.

In Figure 6.7c, the anomaly is located near the center of the road, posing a different kind of challenge for autonomous vehicle. Instead of making intense lateral movements to avoid the obstacle, the vehicle takes a more measured approach. It adjusts its trajectory within a safe range to minimize the degree of rule violation. This illustrates the method's ability to balance between taking evasive action and maintaining adherence to traffic regulations.

Collectively, the scenarios exhibited in Figure 6.7 affirm the efficacy of the DRL method proposed in this thesis. It not only promotes training efficiency and task performance but also gives the autonomous vehicle the skillfulness necessary for responding to a wide spectrum of anomalous situations. The ability to adeptly navigate through diverse anomalies while ensuring compliance

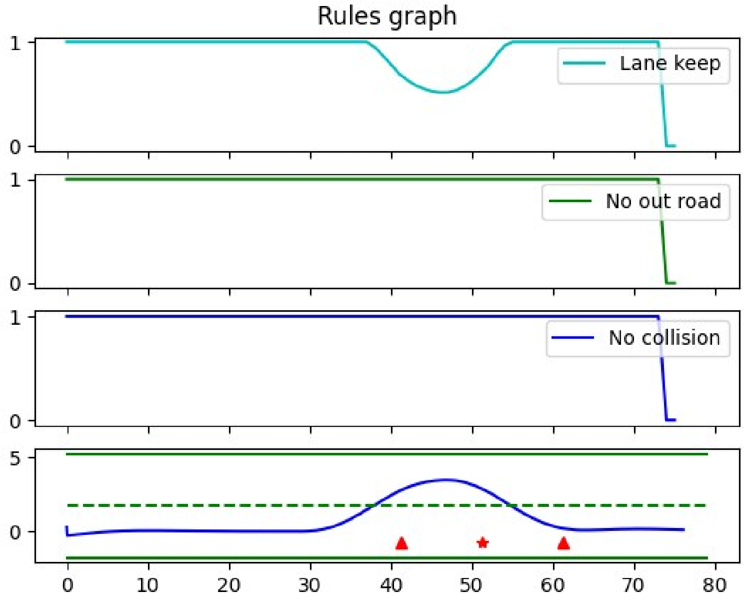


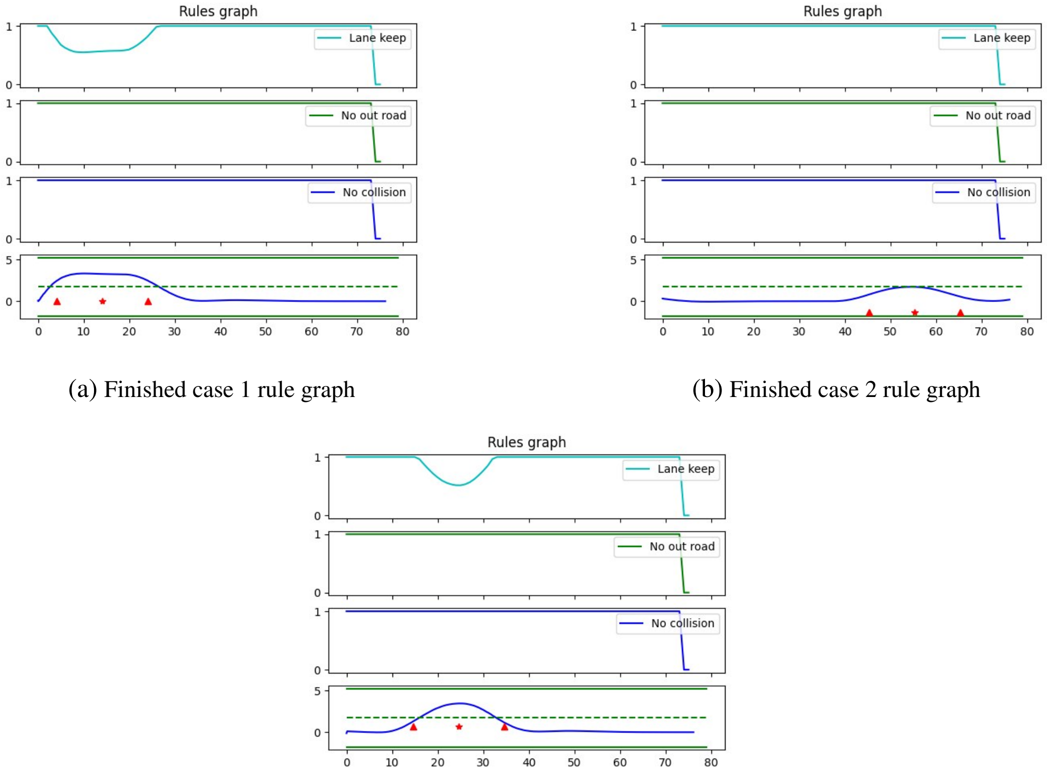
Figure 6.6: The vehicle's trajectory in Frenet coordinates and the degree of rule violation. The top three rows represent the degree of rule violation, with a value of 1 meaning that the rule is fully complied with, and closer to 0 indicating a greater degree of violation. The bottom row shows the vehicle's trajectory. The red ⋆ represents the position of the obstacle, and the red ▲ represents the activation range of the rulebook.

with traffic rules and safety standards is a critical step toward the successful deployment of autonomous vehicles in complex and dynamic real-world environments.

6.3.2 Failure Case Evaluation

Based on Figure 6.8, it is observed that the success rate of the autonomous vehicle in completing the task did not reach 100% after training. This implies that there were instances where the vehicle failed to accomplish the task. It is imperative to investigate and analyze these failure cases to understand the underlying reasons and facilitate the development of more robust solutions in the future. Through numerous experiments and assessments, several typical failure cases have been identified and are presented in Figure 6.8. In these scenarios, the vehicle collided with an obstacle in the vicinity of the anomaly and consequently failed to complete the task. However, it is crucial to note that the causes of these collisions vary.

In Figure 6.8a, the vehicle is faced with an obstacle on the road and decides to veer off the road to avoid a collision. However, in doing so, it ends up colliding with a building or object on the side of the road, which results in failure. This scenario highlights the challenge faced by autonomous vehicles when required to make split-second decisions. The vehicle's decision to leave the road suggests a prioritization to avoid the immediate obstacle, but this short-term reaction lacked consideration of other environmental factors like the objects adjacent to the road. In such scenarios, improving the model's environmental awareness and decision-making process



(a) Finished case 1 rule graph

(b) Finished case 2 rule graph

(c) Finished case 3 rule graph

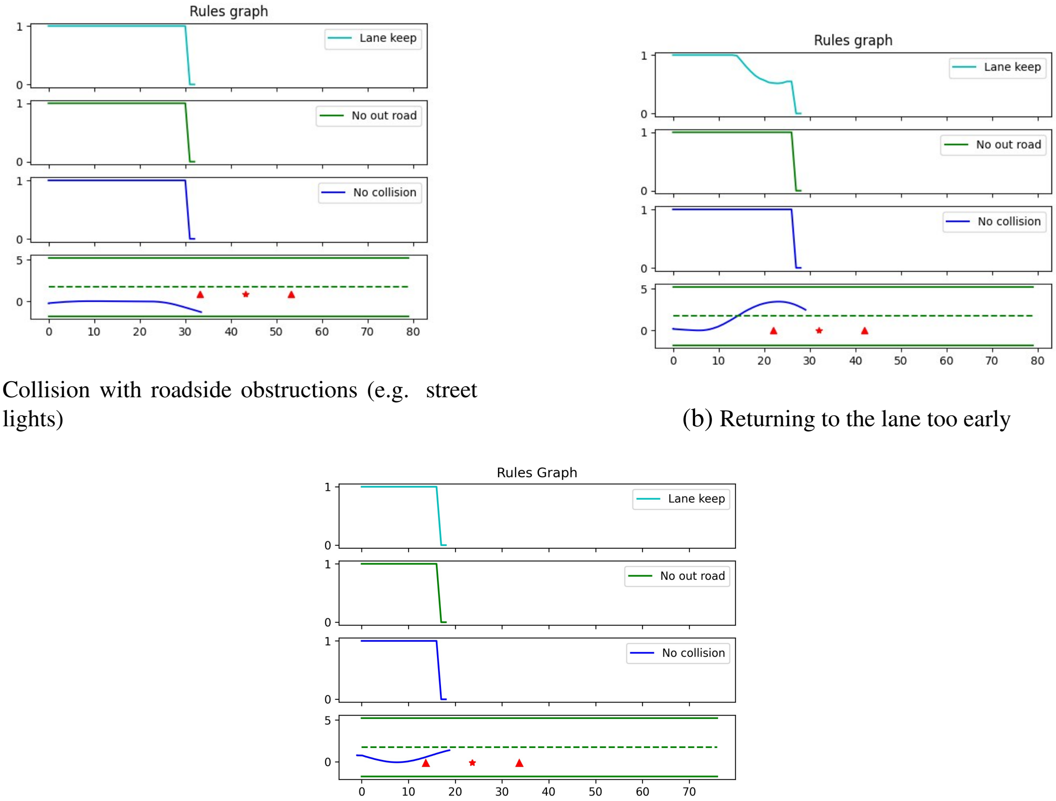
Figure 6.7: Finished rule graphs

to account for a wider range of factors could be beneficial.

In Figure 6.8b, the vehicle successfully changes lanes to avoid the obstacle but chooses to return to the original lane prematurely, at a point when it has not fully passed the obstacle. This results in a collision with the obstacle. This case underlines the importance of proper timing in the vehicle's decision-making process. It may be necessary to develop more refined criteria for the vehicle to ascertain when it is safe to return to the original lane, potentially by enhancing the model's ability to accurately gauge the positions and dimensions of obstacles.

Conversely, in Figure 6.8c, the vehicle delays its decision to change lanes until it is too close to the obstacle. At this juncture, there is not enough space for the vehicle to execute the lane change safely, leading to a collision. This scenario highlights the need for the vehicle to be more proactive in its decision-making, particularly in situations with limited space and time. Enhancing the model's predictive capabilities to anticipate the need for maneuvers well in advance could be instrumental in mitigating such issues.

In summary, these failure cases underline the complexities involved in autonomous vehicle navigation, especially in the presence of anomalies. Analyzing these cases sheds light on the areas where the current model can be improved, such as environmental awareness, timing in decision-making, and predictive capabilities. The insights gained through the examination of these failure cases will be invaluable in refining the model for better performance and safety in future iterations.



(a) Collision with roadside obstructions (e.g. street lights)

(b) Returning to the lane too early

(c) Changing lane too late

Figure 6.8: Failure rule graphs

6.4 Limitations and Challenges

Before delving into the details of future work that could enhance the research presented in this thesis, it is essential to acknowledge the limitations and challenges encountered in the current study. By recognizing these aspects, a clearer understanding of the scope of the research is achieved, and a foundation is laid for future enhancements and adjustments. The subsequent sections elaborate on the limitations and challenges associated with the methodology and evaluation of the autonomous vehicles' behavior in anomaly traffic rule scenarios using DreamerV3 and trajectory generation with Rulebook.

6.4.1 Limitations

This study focuses on autonomous vehicles operating at low speeds (20 kilometers - 30 kilometers/hour). Consequently, the effectiveness of the method at higher speeds cannot be guaranteed. Moreover, this research employs uniform speed tracking, while trajectory generation only takes into account a single terminal parameter, lateral offset (df). This may limit the generality of the conclusions drawn.

6.4.2 Challenges

The presence of various types of anomalies increases the difficulty for DreamerV3 to recognize obstacles and make predictions. This might have contributed to the inability to accurately assess the timing for lane changes and rejoining the original lane in some of the failure cases during the evaluation.

Additionally, the custom coefficients corresponding to the ’rule’ rewards are often based on intuition and require multiple experiments to fine-tune to optimal values. Furthermore, the neural network structure of DreamerV3 is relatively complex, and integrating it with the CARLA environment demands a substantial amount of time and effort.

7 Conclusion and Future Work

In this final chapter, a summary of the findings and contributions made through this thesis is presented. Additionally, limitations and potential areas for future work are discussed to build upon the insights and methods presented in this thesis.

7.1 Summary of Findings

The focus of this thesis was to develop an effective and efficient approach for training autonomous vehicles to navigate through traffic rule exception scenarios, with a particular emphasis on anomaly traffic rule exception scenarios. The method proposed was based on a combination of trajectory generation, a structured Rulebook, and DreamerV3 as the deep reinforcement learning model. The performance of the proposed method was evaluated through a series of experiments, and the findings were promising:

1. Enhanced Training Speed: The method achieved faster convergence in the learning process compared to the baseline. This is especially beneficial in practical implementations, where time is often a critical factor.

2. Superior Training Outcomes: In addition to a faster training process, the model exhibited superior performance in navigating through different anomaly traffic rule exception scenarios. This is indicative of the robustness and practicality of the approach.

3. Effective Integration of Domain Knowledge: By incorporating a structured Rulebook, domain knowledge was effectively integrated into the training process. This integration has been instrumental in enabling self-driving cars to comply with real-world traffic laws and cope with traffic rule exception scenarios.

Despite the promising results, it is important to acknowledge the limitations of this study, one notable limitation is that the definition of the Rulebook's activation range during its integration into the reward function was largely based on intuition and experience. This approach may not be universally applicable and could be a constraint when defining appropriate activation ranges for different rule exception scenarios.

7.2 Future Work

Given the limitations and the potential for further refinement of the proposed method, the following areas are recommended for future research and development:

1. Refinement of Rulebook Integration Process: Future work could focus on developing more systematic and data-driven approaches to defining the Rulebook's activation range. This might include utilizing optimization algorithms or learning-based methods to automatically tune the activation ranges based on the characteristics of the environment and the anomaly scenarios.

2. Trajectory Generation Under Varied Speed Scenarios: Another promising direction for research is to explore how the trajectory generation component of the method can be adapted for different speed scenarios. This could lead to the creation of more flexible and adaptive navigation strategies for autonomous vehicles, allowing them to operate efficiently under a wider range of conditions.

A List of Figures

B List of Tables

C Bibliography

[1] E. A. Aguilar, L. Berducci, A. Brunnbauer, R. Grosu, and D. Ničković. From STL rulebooks to rewards. 2021.

[2] S. Aradi. Survey of deep reinforcement learning for motion planning of autonomous vehicles. 23(2):740–759. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[3] S. Aradi, T. Becsi, and P. Gaspar. Policy Gradient Based Reinforcement Learning Approach for Autonomous Highway Driving. In *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pages 670–675, Aug. 2018.

[4] C. Baier and J.-P. Katoen. *Principles of model checking*. The MIT Press. OCLC: ocn171152628.

[5] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer. Reinforcement learning with iterative reasoning for merging in dense traffic. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[6] R. I. Brafman and M. Tennenholtz. R-max – a general polynomial time algorithm for near-optimal reinforcement learning.

[7] Z. Cao, D. Yang, S. Xu, H. Peng, B. Li, S. Feng, and D. Zhao. Highway exiting planner for automated vehicles using reinforcement learning. 22(2):990–1000. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[8] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator.

[10] K. Esterle. Formalizing and modeling traffic rules within interactive behavior planning. page 145.

[11] G. Fainekos, H. Kress-Gazit, and G. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2020–2025. IEEE.

[12] A. R. Fayjie, S. Hossain, D. Oualid, and D.-J. Lee. Driverless car: Autonomous driving using deep reinforcement learning in urban environment. In *2018 15th International Conference on Ubiquitous Robots (UR)*, pages 896–901.

[13] Fehér, S. Aradi, F. Hegedűs, T. Bécsi, and P. Gáspár. Hybrid DDPG approach for vehicle motion planning:. In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics*, pages 422–429. SCITEPRESS - Science and Technology Publications.

[14] J. Guo, S. Cheng, and Y. Liu. Merging and diverging impact on mixed traffic of regular and autonomous vehicles. 22(3):1639–1649. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[15] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination.

[16] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models.

[17] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models.

[18] C.-J. Hoel, K. Wolff, and L. Laine. Automated speed and lane change decision making using deep reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2148–2155. ISSN: 2153-0017.

[19] Y. Hu, A. Nakhaei, M. Tomizuka, and K. Fujimura. Interaction-aware decision making with adaptive strategies under merging scenarios.

[20] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi. End-to-End Race Driving with Deep Reinforcement Learning, Aug. 2018.

[21] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez. Deep reinforcement learning for autonomous driving: A survey. 23(6):4909–4926. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[22] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. 53(1):287–297. Conference Name: IEEE Transactions on Automatic Control.

[23] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone. Reward (mis)design for autonomous driving.

[24] H. Kress-Gazit, M. Lahijanian, and V. Raman. Synthesis for robots: Guarantees and feedback for robot behavior. 1(1):211–236. _eprint: https://doi.org/10.1146/annurev-control-060117-104838.

[25] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah. A survey of deep learning applications to autonomous vehicle control. 22(2):712–733. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[26] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff. Formalization of interstate traffic rules in temporal logic. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 752–759. ISSN: 2642-7214.

[27] N. Mehdipour, M. Althoff, R. D. Tebbens, and C. Belta. Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges. 152:110692.

[28] M. Moghadam, A. Alizadeh, E. Tekin, and G. H. Elkaim. An end-to-end deep reinforcement learning approach for the long-term short-term planning on the frenet space.

[29] J. Morse, D. Araiza-Illan, J. Lawry, A. Richards, and K. Eder. A fuzzy approach to qualification in design exploration for autonomous robots and systems. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6.

[30] S. S. Mousavi, M. Schukat, and E. Howley. Deep reinforcement learning: An overview. volume 16, pages 426–440.

[31] S. Nageshrao, E. Tseng, and D. Filev. Autonomous Highway Driving using Deep Reinforcement Learning, Mar. 2019.

[32] P. W. Ph.D. *Reinforcement Learning*. "O'Reilly Media, Inc.". Google-Books-ID: Sd-cHEAAAQBAJ.

[33] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818.

[34] A. Rizaldi and M. Althoff. Formalising traffic rules for accountability of autonomous vehicles. page 8.

[35] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow. Formalising and monitoring traffic rules for autonomous vehicles in isabelle/HOL. page 15.

[36] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani. End-to-End Deep Reinforcement Learning for Lane Keeping Assist, Dec. 2016.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, Aug. 2017.

[38] M. Sun, W. Zhao, G. Song, Z. Nie, X. Han, and Y. Liu. DDPG-based decision-making strategy of adaptive cruising for heavy vehicles considering stability. 8:59225–59246. Conference Name: IEEE Access.

[39] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In B. Porter and R. Mooney, editors, *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann.

[40] J. Talamini, A. Bartoli, A. De Lorenzo, and E. Medvet. On the impact of the rules on autonomous drive learning. 10(7):2394. Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

[41] J. Van Den Hoven and G.-J. Lokhorst. Deontic logic and computer-supported computer ethics. 33(3):376–386. Publisher: Wiley.

[42] J. Wang, Y. Wang, D. Zhang, Y. Yang, and R. Xiong. Learning hierarchical behavior and motion planning for autonomous driving.

[43] M. Werling, S. Kammel, J. Ziegler, and L. Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. 31(3):346–359. Publisher: SAGE Publications Sage UK: London, England.

[44] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993. ISSN: 1050-4729.

[45] W. Xiao, N. Mehdipour, A. Collin, A. Bin-Nun, E. Frazzoli, R. D. Tebbens, and C. Belta. Rule-based evaluation and optimal control for autonomous driving.

[46] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun. A Reinforcement Learning Approach to Autonomous Decision Making of Intelligent Vehicles on Highways. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(10):3884–3897, Oct. 2020.

[47] F. Ye, X. Cheng, P. Wang, C.-Y. Chan, and J. Zhang. Automated lane change strategy using proximal policy optimization-based deep reinforcement learning.

[48] L. Yu, X. Shao, Y. Wei, and K. Zhou. Intelligent Land-Vehicle Model Transfer Trajectory Planning Method Based on Deep Reinforcement Learning. *Sensors*, 18(9):2905, Sept. 2018.

[49] J. Zhang, Z. Zhu, and J. Yang. Linear time logic control of linear systems with disturbances.